An Approximation Approach for a Real–World Variant of Vehicle Routing Problem

Khoa Trinh, Nguyen Dang, and Tien Dinh

Faculty of Information Technology, University of Science, VNU-HCMC,
227 Nguyen Van Cu, Ho Chi Minh City, Vietnam
ttdkhoa@acm.org, {dttnguyen,dbtien}@fit.hcmus.edu.vn

Abstract. In this paper, we study a real-world variant of the Vehicle Routing Problem, arising from the work of distributing products of an industrial corporation. Given information about vehicles, depots, types of products, and the location of customers, we wish to determine the minimum number of vehicles required to pick up and deliver products to customers within a time limit. Using these vehicles, we are interested in a routing plan that minimizes the total travelling time. Our proposed solution is presented as follows: First, we introduce an exact algorithm using dynamic programming; Second, by relaxing dynamic programming formulation, we give a heuristic algorithm; Third, we provide our experimental results of the heuristic algorithm on real-world datasets provided by the corporation and highlight the effectiveness of the proposed algorithm.

Keywords: Vehicle routing and scheduling, transportation, dynamic programming, heuristics.

1 Introduction

The Vehicle Routing Problem (VRP), which is first proposed by Dantzig and Ramser [9] in 1950s, has proved to be practically useful. In particular, the VRP has a wide variety of applications in transportation and distribution industries [10]. The VRP is long known to be **NP**-hard [8]. In the last fifty years, many different methods have been developed to solve the VRP and its variants, including exact and heuristic algorithms. Laporte et al. published a detailed survey [12] of different approaches for VRP. Dynamic Programming (DP) is among the class of exact algorithms. The use of DP for the classic VRP can be dated back to the work by Eilon, Watson-Gandy and Christofides in [5]. In [3], the authors introduced a method called state-space relaxation which helps prune the state-space associated with the DP formulation, providing a lower bound on optimal solutions for the Capacitated Vehicle Routing Problem. More recently, in [1,4], DP is adopted to give approximate solution to the VRP with stochastic demands. The authors in [2] introduce a DP approach for a variant of Vehicle Routing Problem with Time Windows. However, in most of the real-world case studies,

N.T. Nguyen et al. (Eds.): New Challenges for Intelligent Information, SCI 351, pp. 87–96.
springerlink.com

heuristic methods are widely used due to their running time efficiency in solving large-scale problem instances while offering quite good approximate solutions.

In this paper, we consider a real-world variant of the VRP, motivated from the work of distributing dairy products by Vinamilk – the biggest dairy corporation in Vietnam. They have a fleet of 23 vehicles, in which each vehicle has its own capacity. There are two depots where dairy products will be loaded into vehicles. Vinamilk offers two main types of dairy products: milk and ice-cream. Each day, the corporation needs to serve between 100 and 1500 requests from customers in the city. In this problem, each vehicle starts from a garage, comes to depots several times to load products, and delivers them to customers until either all customers are served or the maximum operating time is reached. This means that a vehicle can take more than one route. The main objective is to minimize the required number of vehicles. If there are more than one routing plan using that number of vehicles, we want to minimize the total travelling time of all vehicles.

Moreover, due to the space limitation of the depots, there are only three available parking slots in each depot. In other words, the number of vehicles loading products simultaneously could not be more than three. This suggests a kind of scheduling problem, in which we need to find the order of vehicles entering the depots. A combination of the VRP and scheduling problem like this one often happens in practice. However, to the best of our knowledge, it is just mentioned only one time in [11], in which the number of depots, as well as the number of available parking slots, is fixed to one and the adopted method is the Ant Colony metaheuristics.

2 Problem Formulation

In this section, we will formulate the problem rigorously. Let $N = \{1, 2, \cdots, n\}$, $D = \{1, 2\}$ and $P = \{1, 2\}$ be the sets of customers, depots, and types of products respectively. The customer i requests a quantity q_i of a specific product p_i from some depot d_i. We are given a fleet $F = \{1, 2, \cdots, m\}$ of vehicles, in which the vehicle v can load the product p with the maximum capacity of $C_{v,p}$ (number of boxes.) All vehicles, having the same maximum operating time t_{max}, start at a garage g_0, deliver commodities to customers in several routes and finally go back to this garage. For simplicity of later implementation, we consider g_0 a special customer, requesting a zero quantity of every product type.

In each route, a vehicle needs to go to a depot, and picks up a quantity q of some type of product, which must not exceed its maximum capacity for that type. Note here that, as a constraint of the problem, each vehicle can only carry exactly one type of product in a route. The vehicle then delivers all of the product to appropriate customers before starting a new route. There is a fixed rate of loading and unloading commodity, namely r_{load} and r_{unload}. This means that it takes qr_{load} (or qr_{unload}) time unit to load (or unload) a quantity q of commodity. Besides, there are only n_s "slots," numbered from 1 to n_s, for every depot so that at most n_s vehicles can load products simultaneously at any depot.

We are also given a complete graph $G = (V, E)$, where $V = \{g_0\} \cup N \cup D$. For every pair of vertices $i, j \in V$, we know the travelling time $A_{i,j}$ from i to j. Our objective is to find the minimum number of given vehicles that can satisfy the demand of all customers, breaking tie by minimizing the total travelling time of all vehicles.

3 An Exact Algorithm Using Dynamic Programming

In this algorithm, we iterate through every subset $F' \subseteq F$ of given vehicles and try to find out the minimum total travelling time in order to satisfy all customers by using these vehicles, if possible. For each F', we then break the original problem into many subproblems, which can be characterized by a state. Each state is a 4-tuple $\langle U, l, t, T \rangle$ where

1. $U \subseteq N$ is the set of customers who are to be served.
2. l is a vector containing current locations of all vehicles.
3. t is a vector containing time used by all vehicles.
4. T is a $|D| \times n_s$ matrix, in which $T_{i,j}$ indicates the next free time point of slot j at depot i.

Let $f(U, l, t, T)$ be the minimum total travelling time for serving customers in the set U, given information about locations, time used by vehicles, and the status of slots in depots. Basically, given that l_0 contains g_0's only and there are no elements of t_0 greater than t_{max}, we have $f(\emptyset, l_0, t_0, T_0) = 0$, since there are no customers in this cases, and all vehicles are at garage.

When $U \neq \emptyset$, it is clear that an optimal solution to this subproblem consists of routes of all vehicles. In other words, this problem can be considered a multistage decision problem, in which, at any stage, we decide a route for a vehicle. To be more precise, let $\Re_{i,j} = \{r_{i_1}, r_{i_2}, \cdots\}$ be the set of routes which starts from the slot j at the depot i, in some optimal solution. Let us choose some $\Re_{i',j'} \neq \emptyset$ and a route $r_{opt} \in \Re_{i',j'}$ by some vehicle v with the earliest leaving time. Let X_{opt} be the set of customers who are in the route r_{opt}. According to Bellman's principle of optimality [7], the set remaining routes $\bigcup \Re_{i,j} \setminus r_{opt}$ must form an optimal solution for a smaller subproblem of serving $U \setminus X_{opt}$ customers with the current position of v being the last customer in r_0, the time used by v being the sum of t_v and time spent for the route r_{opt}, and the next free time point $T_{i',j'}$ being the time point when v leaves the depot.

Note that r_{opt} is indeed the shortest Hamiltonian path from some depot d to the last customer in the subgraph $G[X \cup \{d\}]$. Otherwise, we could find a shorter route, which reduces the total travelling time. Since we do not know X_{opt}, we iterate through every nonempty subset X of U, recursively solve the smaller subproblem on $U \setminus X$, combine the cost of this choice with the optimal value of the subproblem, and look for the minimum total cost.

In summary, the dynamic programming formulation is

$$f(U,l,t,T) = \min_{\substack{X\subseteq U, i\in X\cup\{g_0\}, v\in F' \\ d\in D, p\in P, 1\le s\le n_s}} \left\{ f(U\setminus X, l', t', T') + A_{l_v,d} + g(G[X\cup\{d\}], d, i) \right\}, \tag{1}$$

with the constraints:

$$\forall i \in X : d_i = d, p_i = p, \tag{2}$$

$$\sum_{j\in X} q_j \le C_{v,p}, \tag{3}$$

$$t'_v \le t_{max}. \tag{4}$$

where,

- The function $g(G[X\cup\{d\}], d, i)$ returns the shortest Hamiltonian path from d to i in the induced subgraph $G[X\cup\{d\}]$, where d is the depot that the vehicle started and i is the last customer has been served in the route. This is a classical NP-complete problem, which can also be solved by a slightly modified dynamic programming approach in [6].
- The vectors l', t' and T' take the old values of l, t and T respectively, except the following elements. The new location of v is i. The new time used of v is time point $\max\{A_{l_v,d} + t_v, T_{d,s}\}$, when it was ready load products, plus the total time spent for loading and unloading products, and the travelling time in the route. Similarly, the new next free time point of the slot s at depot d, when v leaves the depot, should be the sum of the time point $\max\{A_{l_v,d} + t_v, T_{d,s}\}$ and the time spent for loading products.

$$l'_v = i, \tag{5}$$

$$t'_v = \max\{A_{l_v,d} + t_v, T_{d,s}\} + (r_{load} + r_{unload}) \sum_{j\in X} q_j + g(G[X\cup\{d\}], s, i), \tag{6}$$

$$T'_{d,s} = \max\{A_{l_v,d} + t_v, T_{d,s}\} + r_{load} \sum_{j\in X} q_j. \tag{7}$$

Note here that the constraint (2) makes sure that every customer in X receives their desired commodity from the right depot. Also (3) and (4) are constraints about the maximum capacity and operating time of vehicles. The solution to the original problem, given a fleet F', is $f(N, 0_{|F'|\times 1}, 0_{|F'|\times 1}, 0_{|D|\times n_s})$.

Although this dynamic programming formulation provides optimal solution to the problem, it is not practical to implement this algorithm in practice since the number of states is exponential to the input size. Moreover, computing the function f for just one state by (1) also requires the access to an exponential number of other states. This phenomenon is known as the "curse of dimensionality." In the next section, we propose a heuristic algorithm based on relaxation of (1).

4 A Heuristic Algorithm

4.1 Description

Since the size of F may be quite large, considering every subset of F, as in the previous section, is not practical. Our approach is to gradually build up a fleet F', and try to satisfy as much customers' demand as possible using this fleet, which we will discuss in the following paragraphs. If F' cannot deliver all of the products, we then greedily add another available vehicle v into F' such that it is capable to carry the largest quantity of products requested by the remaining not served customers, and repeat the process.

Now, assuming that we are given a set of vehicles, we want to plan routes in order to minimize the total remaining demand from customers. Using similar idea as in the above exact algorithm, we start from the initial state $\langle N, 0_{|F'|\times 1}, 0_{|F'|\times 1}, 0_{|D|\times n_s}\rangle$, trying to add a new route in each iteration. Here, we are only interested in "potentially good" states whose ratio of total travelling time by all vehicles and total delivered quantity of products is as small as possible. Intuitively, this ratio is the average time to deliver a unit of product and it tells us how efficient all vehicles has deliver products to the customers so far.

For adding a new route, we first pick up from the current state $\langle U, l, t, T\rangle$ the vehicle v which has the minimum time used t_v, the product p and depot d which maximize the total remaining quantity $\min\{C_{v,p}, \sum_{i\in U, p_i=p, d_i=d} q_i\}$ that v can load, and the slot s at the depot d with the minimum time difference $|T_{d,s} - (t_v + A_{l_v,d})|$. Starting from the depot d with the vehicle v, we construct a new route by using the simple Breadth First Search on customers who requested the product p.

In the same manner as defining states, we characterize each tentative route by a set of customers R in the route, the current position i of the vehicle on the route, the total travelling time t so far and the total quantity q already delivered. Let us denote a tentative route by $\langle R, i, t, q\rangle$. We maintain a queue Q of such routes. We try to extend routes in Q, if possible, and push the new routes into Q'. Next, we assign $Q \leftarrow Q'$, and repeatedly extend routes in Q. Indeed, the size of Q would grow exponentially. Therefore, aiming at a "good" state, we prune Q and only keep at most α routes with the smallest ratio $\frac{t}{q}$. In this paper, we choose $\alpha = O(n)$, and we will provide experimental results on some specific values of α in the next section. For routes which cannot be extended anymore, we push them into a list called $Q_{candidates}$. Finally, we update the current state by adding the route in $Q_{candidates}$ such that the above ratio is minimum.

Fig. 1. illustrates the search tree, in which the root is the depot d. A tentative route is a path from the root to some node. In each iteration, the queue Q contains all nodes which have the same height. All white nodes are pruned and we continue searching from $\alpha = 4$ marked nodes, which indicate routes with the smallest ratio $\frac{t}{q}$.

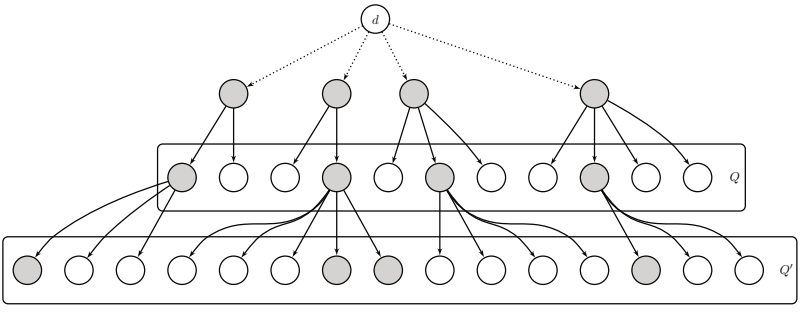

Fig. 1. Illustration of searching for new routes

Assuming that we already chose a vehicle v, the following procedure FIND–ROUTE takes input as the current state and returns the list of routes in $Q_{candidates}$ described above.

Procedure 1. FIND–ROUTE($\langle U, l, t, T\rangle$, v)

1: Choose the product p and the depot d which maximize the total remaining quantity $\min\{C_{v,p}, \sum_{i\in U, p_i=p, d_i=d} q_i\}$ that v can load
2: Choose the slot s at the depot d with the minimum time difference $|T_{d,s} - (t_v + A_{l_v,d})|$
3: Push an initial route $\langle \emptyset, d, A_{l_v,d}, 0\rangle$ into the queue Q
4: $Q_{candidates} \leftarrow \emptyset$
5: **while** $Q \neq \emptyset$ **do**
6: $\quad Q^{'} \leftarrow \emptyset$
7: $\quad$ **for all** $\langle R, i, t, q\rangle \in Q$ **do**
8: $\quad\quad Q^{''} \leftarrow \emptyset$
9: $\quad\quad$ **for all** $j \in U \setminus R$ such that $d_j = d, p_j = p$ and $q_j + q \leq C_{v,p}$ **do**
10: $\quad\quad\quad$ **if** $\max\{T_{d,s}, t_v + c_{l_v,d}\} + (q_j + q)(r_{load} + r_{unload}) + A_{i,j} + A_{j,g_0} \leq t_{max}$ **then**
11: $\quad\quad\quad\quad$ Push $\langle R \cup \{j\}, i, t + A_{i,j}, q + q_j\rangle$ into $Q^{''}$
12: $\quad\quad$ **if** $Q^{''} = \emptyset$ **then**
13: $\quad\quad\quad$ Push $\langle R, i, t, q\rangle$ into $Q_{candidates}$
14: $\quad\quad$ **else**
15: $\quad\quad\quad$ Push all tentative routes in $Q^{''}$ into $Q^{'}$
16: $\quad$ Sort all tentative routes in $Q^{'}$ increasing according to the ratio of travelling time used and total amount of commodity delivered.
17: $\quad$ Push the first α routes in $Q^{'}$ into Q
18: **return** $Q_{candidates}$

The pseudo-code of the heuristic algorithm is shown as follows.

Algorithm 2. HEURISTIC(α)

1: Input $\leftarrow N, D, P, F, G = (V, E), C, r_{load}, r_{unload}, g_0, t_{max}, n_s$
2: $F' = \emptyset$, $U \leftarrow N$
3: **repeat**
4: **if** $F = \emptyset$ **then**
5: **return** NO SOLUTION
6: Choose a vehicle $v \in F$ such that v is capable to carry the largest quantity of commodity from the remaining customers U
7: $F' \leftarrow F' \cup \{v\}$, $F \leftarrow F \setminus \{v\}$
8: Initialize the state $\langle U, l, t, T\rangle$: $U \leftarrow N$, every vehicles are now at the garage g_0 with $t \leftarrow 0_{|F'|\times 1}$, $T \leftarrow 0_{|D|\times n_s}$ and the cost $f \leftarrow 0$
9: **repeat**
10: Choose the vehicle $v \in F'$ with the minimum t_v
11: $Q_{candidates} \leftarrow$ FIND–ROUTE($\langle U, l, t, T\rangle$,$v$)
12: **if** $Q_{candidates} \neq \emptyset$ **then**
13: Choose the route $\langle R_0, i_0, t_0, q_0\rangle$ in $Q_{candidates}$ with the minimum ratio $\frac{t_0}{q_0}$
14: $f \leftarrow f + t_0$, $U \leftarrow U \setminus R_0$, $l_v \leftarrow i_0$
15: $t_v \leftarrow \max\{T_{d,s}, t_v + A_{l_v,d}\} + q_0(r_{load} + r_{unload}) + t_0$
16: $T_{d,s} \leftarrow \max\{T_{d,s}, t_v + A_{l_v,d}\} + q_0 r_{load}$
17: **until** $Q_{candidates} = \emptyset$
18: **until** $U = \emptyset$
19: **for all** vehicle $v \in F'$ **do**
20: $f \leftarrow f + A_{l_v,g_0}$
21: **return** $\langle F', f\rangle$

4.2 Running Time Analysis

In the procedure FIND–ROUTE, the queue Q contains at most $\alpha = O(n)$ routes so that the two nested for-do loops (lines 7 and 9) run in time $O(n^2)$. After that, sorting all routes in Q' costs $O(n^2 \log n)$ since the size of Q' could contain at most n^2 tentative routes. For each iteration of the while-do loop in line 5, the length of each route increases by one, and every route never contains any vertex more than once, making this loop iterate no more than n times. Therefore, the running time of the FIND–ROUTE subroutine is $O(n^3 \log n)$.

In the main algorithm, the loop in line 3 iterates at most m times since each time the cardinality of F decreases by one. In lines 9..17, after each iteration, we add a new route into our solution and call the procedure FIND–ROUTE. We have indeed the upper-bound of the number of routes is $O(n)$. In summary, the time complexity of the heuristic algorithm is $O(mn^4 \log n)$.

5 Experimental Results

In this section, we provide the experimental results of our heuristic algorithm on datasets provided by Vinamilk Corporation, which will be briefly described as

follows. Each dataset records a real-life instance of the problem in one business day, and the maximum operating time of all vehicles t_{max} is equal to 420 minutes (or 7 hours.) There are two types of products, and the number of requests ranges from around 100 to 1500. In addition, the corporation has a fixed fleet of exactly 23 vehicles with different capacity.

Our algorithm is implemented in C++ (compiler GCC 3.4.2) and tested on a personal computer with the Intel Quad-core 2.33GHz processor and 4GB RAM. In the experiment, we choose a specific value $\alpha = 100$ for all test cases. Generally, if α is too small, say less than 20, it may require a larger number of vehicles. On the other hand, if we choose $\alpha > n$, the running time increases, but the total travelling time could decrease. The detailed results are shown in Table 1.

Table 1. Detailed results of the heuristic algorithm

Instance	No. of Customers	Total Demand	No. of Vehicles	Travelling Time (mins)	Running Time (secs)	No. of Vehicles used by Vinamilk
FEB01	821	9861	13	2676	40	20
FEB02	573	7320	10	2200	12	19
FEB03	658	9977	11	2650	22	11
FEB04	582	8376	10	1941	16	20
FEB05	756	9891	12	2337	26	22
FEB06	602	8949	11	2325	18	21
FEB08	939	14009	17	3348	76	20
FEB09	542	11283	12	2499	18	19
FEB10	720	9854	12	2382	27	19
FEB11	482	13379	13	2740	8	16
FEB18	363	5583	7	1557	3	15
FEB19	819	11597	14	2837	69	21
FEB20	510	9561	10	2177	13	23
FEB22*	1513	16483	23	4655	312	23
FEB23	727	13827	17	3306	111	18
FEB24	726	10395	12	2434	52	19
FEB25	678	8464	11	2153	19	14
FEB26	760	10515	12	2341	75	23
FEB27	607	7018	9	1916	21	18

These instances are taken from the work of distributing dairy products of the corporation in February, 2010. We skip several instances which have relatively small number of requests from the customers. The second and third columns describe the size of the input. The next two columns show the output of our algorithm. The last column describes the number of vehicles used by Vinamilk. Currently, the corporation has a team of two people who solves them by hand using greedy techniques and some experienced–based heuristics. Note that the instance FEB22 is very large, in which the corporation had to rent extra vehicles. In most cases, our method can give a solution which uses much smaller number of vehicles. On average, our proposed algorithm helps reduce the required number of vehicles by 33.9%.

We already developed a system which solves and visualizes the planning routes. Fig. 2. shows four routes, which are part of the solution for FEB06, on the real map of Ho Chi Minh City. Customers are numbered increasingly according to the visited order in each route.

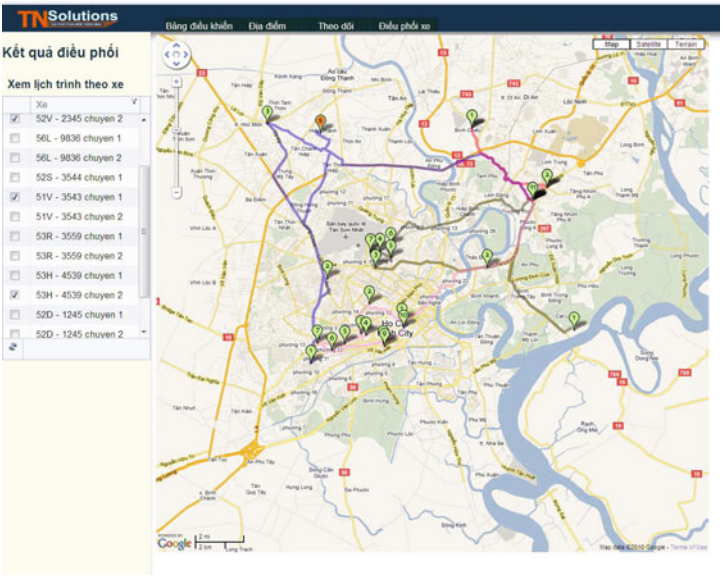

Fig. 2. Illustration of four routes (FEB06) on the real map

6 Conclusions and Future Work

In this work, we have presented a heuristic algorithm to solve a real-world variant of the VRP. In this algorithm, we restrict the state-space, and the solution is built up gradually by adding new routes. The algorithm is then validated experimentally on datasets provided by an industrial corporation. Our experiments show that the proposed method can produce a much smaller required number of vehicles than the current greedy method adopted by the above-mentioned corporation. The running time of the algorithm is also acceptable. Further investigation of metaheuristics on this problem is left open at this point. The literature suggests that metaheuristics such as local search is quite promising to approximate the minimum needed number of vehicles in such a combinatorial optimization problem.

Acknowledgements. We would like to acknowledge the support of Vinamilk Corporation in providing us real data in Feburary 2010 for experimentation and comparison.

This work is a part of the 217/HD-SKHCN project supported by the Department of Science and Technology, Ho Chi Minh City, 2010-2011.

References

1. Novoa, C., Storer, R.: An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. European Journal of Operational Research 196, 509–515 (2009)
2. Leendert, K., Manuel, M., Herbert, K., Marco, J.: Dynamic Programming Algorithm for the Vehicle Routing Problem with Time Windows and EC Social Legislation. Beta Research School for Operations Management and Logistics, University of Twente (2009)
3. Christofides, N., Mingozzi, A., Toth, P.: State-space relaxation procedures for the computation of bounds to routing problems. Networks 11(2), 145–164 (1981)
4. Secomandi, N.: Exact and heuristic dynamic programming algorithms for the vehicle routing problem with stochastic demands. Doctoral Thesis, University of Houston (1998)
5. Eilon, S., Watson-Gandy, C.D.T., Christofides, N., de Neufville, R.: Distribution Management-Mathematical Modelling and Practical Analysis. IEEE Transactions on Systems, Man and Cybernetics, 589–589 (1974)
6. Bellman, R.: Dynamic Programming Treatment of Travelling Salesman Problem. Journal of the ACM 9(1), 61–63 (1962)
7. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton (2003) (Republished)
8. Lenstra, J.K., Kan, A.H.G.R.: Complexity of vehicle routing and scheduling problems. Networks 11, 221–227 (1981)
9. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management Science 6(1), 80–91 (1959)
10. Golden, B.L., Assad, A.A., Wasil, E.A.: Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries. In: The Vehicle Routing Problem, Society for Industrial and Applied Mathematics. SIAM Monographs on Discrete Mathematics and Applications, pp. 245–286. SIAM, Philadelphia (2001)
11. Ortega, P., Oliva, C., Ferland, J., Cepeda, M.: Multiple ant colony system for a VRP with time windows and scheduled loading. Ingeniare, Revista chilena de ingenieria 17, 393–403 (2009)
12. Laporte, G.: Fifty years of vehicle routing. Transportation Science 43(4), 408–416 (2009)