Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2008, Article ID 697981, 17 pages
doi:10.1155/2008/697981

Research Article

Single-Commodity Vehicle Routing Problem with Pickup and Delivery Service

Goran Martinovic, Ivan Aleksi, and Alfonzo Baumgartner

Faculty of Electrical Engineering, Josip Juraj Strossmayer University of Osijek, Kneza Trpimira 2b, 31000 Osijek, Croatia

Correspondence should be addressed to Goran Martinovic, goran.martinovic@etfos.hr

Received 24 May 2008; Revised 6 October 2008; Accepted 19 December 2008

Recommended by Alois Steindl

We present a novel variation of the vehicle routing problem (VRP). Single commodity cargo with pickup and delivery service is considered. Customers are labeled as either cargo sink or cargo source, depending on their pickup or delivery demand. This problem is called a single commodity vehicle routing problem with pickup and delivery service (1-VRPPD). 1-VRPPD deals with multiple vehicles and is the same as the single-commodity traveling salesman problem (1-PDTSP) when the number of vehicles is equal to 1. Since 1-VRPPD specializes VRP, it is $\mathcal{NP}$ hard in the strong sense. Iterative modified simulated annealing (IMSA) is presented along with greedy random-based initial solution algorithm. IMSA provides a good approximation to the global optimum in a large search space. Experiment is done for the instances with different number of customers and their demands. With respect to average values of IMSA execution times, proposed method is appropriate for practical applications.

1. Introduction

In this paper, we consider a real-world problem with a loaded vehicle (the term capacitated vehicle is also used in the literature). The vehicle is available for pickup and delivery service. Single-commodity cargo, that is, same cargo type, is present at the customer sites. The vehicle is able to carry out a Pickup and delivery service of such cargo among the customers. By starting and ending at the origin depot, the vehicle follows a route without having a predefined sequence of Pickup and delivery services. This problem is called a *one-commodity vehicle routing problem* (1-VRPPD). The main feature of 1-VRPPD is that delivery customers can be served with a cargo gathered from Pickup customers. This problem was first defined in [1] as a *one-commodity Pickup and delivery traveling salesman problem* (1-PDTSP), which is the same as 1-VRPPD, when the number of vehicles is equal to 1. Although, 1-VRPPD deals with multiple vehicles, in this paper we consider a single vehicle. In order to solve this problem, the main task of our route planner is to calculate a single suboptimal vehicle route, which starts and ends at the origin depot and satisfies capacity constraints, minimizing the travel distance at the same time. For the first time iterative modified simulated annealing (IMSA) is used to solve this combinatorial optimization problem. In general, the VRP problem and its variations are $\mathcal{NP}$ hard. 1-VRPPD is $\mathcal{NP}$ hard problem in the strong sense since it specializes VRP. The main VRP feature that differs from traveling salesman problem (TSP) is vehicle capacity constraint. Experimental results given by IMSA confirm that complexity of VRP depends on the vehicle capacity. When the capacity tends to infinity, VRP tends to TSP.

The organization of this paper is as follows. In Section 2, we discuss several variations of VPR: the open VRP (OVRP), the *distance-constrained capacitated* VRP (DCVRP), the VRP with backhauling (VRPB), the VRP with simultaneous Pickup and delivery (VRPSPD), and several Hybrid variations of VRP, since they are most similar to 1-VRPPD. Section 3 deals with problem and model definition. Solution to the problem and its implementation are provided in Section 4. In Section 5 we describe our simulation environment, as well as its input and output parameters. Solution performance is illustrated in Section 6 with experimental results.

2. Related work

According to [2], over the last 30 years, less than 10% of the VRP research was devoted to problems where vehicles have two stages, that is, where vehicles are able to Pickup and deliver. The vehicle routing problem with Pickup and delivery is not explored nearly as much as the other variants of the vehicle routing problem.

Recently, [3] described a practical example of heuristics applied on VRP resulting in the transport network operational cost savings of 10% for Melbourne mail distribution at Australia Post. A hybrid genetic algorithm was used to solve VRP in [4] providing better solutions than those obtained using other genetic algorithms.

The *vehicle routing problem with backhauling* (VRPB) considers a vehicle servicing all delivery (*Linehaul*) customers with cargo loaded at the depot, followed by Pickup (*Backhaul*) customer services. Practical applications of this VRP variation are found in grocery industry. A memetic algorithm was used in [5] to solve VRPB. In [6], the authors present a new tabu search algorithm that starts from pseudo-lower bounds and is able to match almost all the best published solutions and find many new solutions.

The *vehicle routing problem with simultaneous Pickup and delivery* (VRPSPD) was introduced in 1989 by Min [7]. It considers both Pickup and delivery service at each customer, while each collected cargo must be returned to the origin depot. This problem is present in milk bottles transporting while empty ones must be returned to the origin depot. A tabu search algorithm, with and without maximum distance constraints, was recently developed in [8] to solve VRP-SPD.

The main difference between VRPB, VRPSPD, and 1-VRPPD is that in 1-VRPPD cargo picked up from Pickup customers can be delivered to delivery customers. In 1-VRPPD, the predefined sequence of servicing customers is not a constraint.

The *distance-constrained capacitated vehicle routing problem* (DCVRP) with a flexible assignment of start and end depots is proposed in [9] and solved by means of a heuristic algorithm on the network model.

The *open vehicle routing problem* (OVRP) considers solving a problem similar to VRP. The difference is that the OVRP route ends when the last customer is served, that is, after servicing customers the vehicle does not return to the depot. An overview of OVRP algorithms is provided in [10]. A heuristic algorithm applied in [11] is comparable in terms of solution quality to the best performing published heuristics. Practical applications of OVRP are in home delivery of packages and newspapers with third-party contractors who use their own vehicles and do not return to the depot.

Hybrid variations of VRP with Pickup and delivery service have also been present in literature. A hybrid variation of VRP is considered in [12] and it is used for blood transportation in health care, where the vehicle route consists of the following sequence: *delivery only, simultaneous Pickup and delivery, Pickup only*. The proposed genetic algorithm (GA) called CLOVES solves the problem in four steps.

The 1-PDTSP, defined in [1, 13], is solved with a branch-and-cut algorithm. A special case of the 1-PDTSP, when vehicle capacity is either 1 or ∞ and customers' demands are either +1 or –1, was defined in [14] and solved for a path and a tree graph topology.

Practical applications of 1-VRPPD are found in the transport of material of the same type where delivery customers correspond to cargo sinks and Pickup customers correspond to cargo sources. That is the case in transportation of gas, earth, sand, eggs, money, and so forth. In such a way, some customers produce goods while others demand those goods.

In this paper, we present a scenario with an employee utilizing our VRP application. In order to find a feasible suboptimal vehicle route, the user imports customer's coordinates and demands, and runs the IMSA algorithm. Via graphical user interface, described in Section 4, the user can select one part of the resulting route in order to find a shorter route.

In this paper, we explore the benefits of solving 1-VRPPD with simulated annealing and its variations.

3. Problem and model definition

Let G be a complete graph, $V = \{v_0, \dots, v_n\}$ a vertex set, $E = \{\{v_i, v_j\} : v_i, v_j \in V\}$ an edge set, $A = \{(v_i, v_j) : v_i, v_j \in V\}$ an arc set. G has dual, directed and undirected, representations, as Figure 1 illustrates. Vertices $v_i, i \in \{1, \dots, n\}$ correspond to the customers and vertex v_0 corresponds to the depot vertex. Arc $a \in A$ with start in v_i and end in v_j is denoted by (v_i, v_j). Non-negative cost c_{ij} is associated with each edge $e = \{v_i, v_j\} \in E$ that corresponds to the travel distance between two vertices v_i and v_j. Also, a binary number $x_e \in \{0, 1\}$ is assigned to e, where 1 represent visited edge and 0 represents unvisited edge. Thus, cost matrix $\mathbf{c}$ is symmetric, having the same cost in both directions, that is, $c_{ij} = c_{ji}\ \forall v_i, v_j \in V$. The use of loop edges is not allowed, that is, $c_{ii} = +\infty\ \forall v_i \in V$.

Customer demand $q_i > 0$ indicates cargo that needs to be delivered, that is, the customer requires delivery service. Customer demand $q_i \leq 0$ indicates cargo that needs to be collected, that is, the customer requires a Pickup service. When $q_i = 0$, Pickup service is considered in order to visit the costumer.

For a given vertex set $S \subseteq V$, let $q(S) = \sum_{i \in S} q_i$ denote the total demand of the set. Let $\delta(S)$ and $E(S)$ denote the set of edges $e \in E$ that have only one or both endpoints in S respectively.

Let $\delta^+(S) := \{(v_i, v_j) \in A : v_i \in S, v_j \notin S\}, \forall S \subset V$, denote the set of vertices j directly reachable from vertex i. Let $\delta^-(S) := \{(v_i, v_j) \in A : v_i \notin S, v_j \in S\}, \forall S \subset V$, denote the set of vertices j from which vertex i is directly reachable. Let $\delta(S) := \{\{v_i, v_j\} \in E : v_i \in S, v_j \notin S\}$, $\forall S \subset V$, denote the set of vertices j directly reachable from vertex i. $\delta^+(S)$ and $\delta^-(S)$ are given with respect to A, while $\delta(S)$ is given with respect to E. To ensure cargo conservation let the depot cargo

$$q_0 = -\sum_{i=1}^{i=n} q_i, \tag{3.1}$$

and let

$$K = \sum_{\forall v_i \in V : q_i > 0} q_i = -\sum_{\forall v_i \in V : q_i \leq 0} q_i. \tag{3.2}$$

A vehicle with a fixed positive vehicle capacity Q is available at the depot. To ensure feasibility, it is assumed that vehicle cargo $q_i \in [0, Q]$ is a positive whole number and never exceeds its maximal value. An example of this model is illustrated in Figure 2.

Bearing in mind the aforementioned, a mathematical formulation of 1-VRPPD can be defined as

$$\text{Minimize} \quad \sum_{e \in E} c_e x_e \tag{3.3}$$

$$\text{subject to} \quad \sum_{e \in \delta(v_i)} x_e = 2 \quad \forall v_i \in V, \tag{3.4}$$

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \subseteq V \setminus \{v_0\}, \tag{3.5}$$

$$\sum_{e \in \delta(v_0)} x_e \leq 2M, \tag{3.6}$$

$$\sum_{a \in \delta^+(v_i)} g_a - \sum_{a \in \delta^-(v_i)} g_a = q_i \quad \forall v_i \in V, \tag{3.7}$$

$$0 \leq g_a \leq \frac{Q}{2} x_e \quad \forall a \in A,\ \forall e \in E, \tag{3.8}$$

$$x_e \in \{0, 1\} \quad \forall e \in E. \tag{3.9}$$

Equation (3.4) ensures that Hamiltonian cycle exists. With (3.5) a vehicle is forced to visit and leave all customers along the route. (3.6) ensures that at least one of M vehicles visits and leaves depot vertex. (3.7) describes the flow of Pickup and delivery commodities, respectively. It ensures that cargo demands are appropriately cared for, that is, the cargo is gathered or delivered in its entirety. (3.8) ensures that vehicle capacity value is never less than zero and that it never exceeds its maximal value. It defines a feasible 1-VRPPD route, where g_a is a continuous variable that has mathematical meaning of a vehicle load going through an arc a. (3.9) represents x_e as a binary type variable assigned to e whose value is 1 if e is visited and 0 otherwise.

In this section, the 1-VRPPD model is presented as a combination of a generic VRP model described in [15] and added vehicle capacity constraints. References [1, 5, 13] describe similar VRP models with Pickup and delivery services.

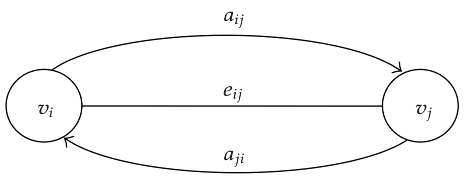


Figure 1: Dual, directed and undirected, graph representation.

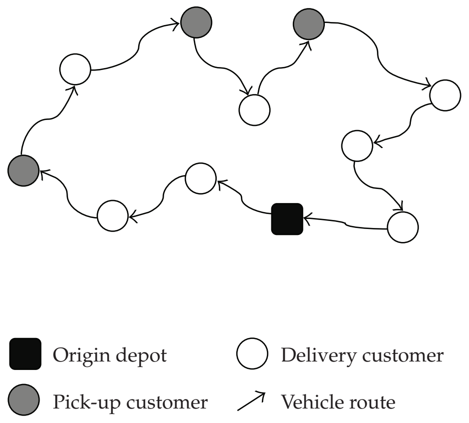


Figure 2: Vehicle routes of the one-commodity vehicle routing problem with Pickup and delivery (1-VRPPD).

4. Proposed solution and implementation

Iterative modified simulated annealing (IMSA) algorithm is a heuristic algorithm with iterative approach strategy used for solving combinatorial optimization problems. IMSA and exact permutation algorithm (EPA) are used to solve the 1-VRPPD $\mathcal{NP}$ hard problem.

In this section, we describe the implementation of IMSA algorithm. The number of nodes, that is, customers, is denoted by n. The number of vehicles M is 1.

4.1. Exact permutation algorithm (EPA)

For the purpose of improving the optimal route length for small graph instances in which the number of nodes is $n \leq 15$, we use Exact Permutation Algorithm (EPA). Instances with graphs, which have more than 15 nodes, are not practical and solving EPA for an arbitrary n is impossible. Its computational complexity is $O(n!)$, therefore, it cannot be enumerated in polynomial time. EPA represents the Exeter's permutation algorithm described in [16]. EPA is used for improving existing solution via interactive graphical user interface (GUI). Implementation of EPA is depicted at the end of Section 5.

4.2. Simulated annealing (SA)

Simulated annealing (SA) was first described in [17]. SA is a heuristic algorithm used for solving global optimization problems [13, 15]. In the beginning of the algorithm, at high temperature T, SA acts as a random algorithm. Subsequently, the current feasible

```
1. $Initialize$(nCounts$_{\max}$, nChanges$_{\max}$, $T_{\text{start}}$, $T_{\text{end}}$, $B, L, \alpha$)
2. $x$ = $InitialSolution$( );
3. while ($Feasible(x)$ == FALSE )
4.     $x$ = $InitialSoiution$( );
5. minDist = $Dist(x)$;
6. for (nCounts = nCounts$_{\max}$; nCounts > 0; nCounts ··)
7.     for (nChanges = nChanges$_{\max}$; nChanges > 0; nChanges ··)
8.         for ( $k = 0, T = T_{\text{start}}$; $((k < B)$ && $(T > T_{\text{end}}))$; $k$ ++, $T = T/\alpha$)
9.             for ($z = 0$; $z < L$; $z$ ++)
10.                $y$ = $RandomNeighborSequence(x$, nChanges);
11.                if ($Feasible(y)$== TRUE)
12.                    $\Delta$ = minDist − $Dist(y)$;
13.                    if (($p = \exp(\Delta/T)$) > 1.0)
14.                        $p$ = 1.0;
15.                    if ($RandomNumber$([0,1]) <= $p$)
16.                        if ($\Delta > 0$)
17.                            $x = y$;
18.                            minDist = $Dist(x)$;
```

Algorithm 1: (IMSA) algorithm.

solution X changes to another feasible neighbor solution Y in a very random way. In our implementation, neighbor feasible solution Y is created with *nSAChanges* random feasible substitutions made on current feasible solution X. In this work, we consider nSAChanges = $2\sqrt{n}$. As system anneals proportionally with the number of iteration ϑ, the algorithm becomes more deterministic. Criterion, which increases algorithm determinism, as ϑ growths and T falls, is the main part of the SA algorithm and is defined with temperature-probability function $f(\Delta)$ (4.1).

$$f(\Delta) = \begin{cases} 1, & \text{if } \Delta \geq 0, \\ e^{\Delta/T_{\vartheta}}, & \text{if } \Delta < 0, \end{cases} \tag{4.1}$$

where Δ = *RouteLength*(X)-*RouteLength*(Y) is the amplitude defined with the difference between route lengths of X and Y. Notation $T_{\vartheta} \in [T_{\text{start}}, T_{\text{end}}]$ represents the current temperature at iteration ϑ. Temperature T_{ϑ} falls linearly from T_{start} to T_{end} as ϑ growths. Temperature-probability function $f(\Delta)$ results with continuous probability $[0, 1]$ for rejecting new solutions. When $\Delta \geq 0$, the current feasible solution X is replaced with better feasible solution Y. Otherwise, random number $r \in [0, 1]$ is generated and compared with $f(\Delta)$. If $r \geq f(\Delta)$, new feasible solution Y is rejected and the current solution X remains unchanged. If $r < f(\Delta)$, then the current feasible solution X with shorter route length is replaced with feasible solution Y with longer route length. Occasional acceptance of a solution that leads to a longer route prevents the algorithm from becoming stuck in a local minimum.

4.3. Modified simulated annealing (MSA)

Modified simulated annealing (MSA) algorithm inherits features of SA algorithm with a difference in creating neighbor solutions Y. MSA deals with multiple feasible neighbor solutions Y made from the current feasible solution X. Neighbor solutions Y are created

Table 1: Parameters of implemented algorithms.

Method	SA	MSA	IMSA
nSAChanges	$2\sqrt{n}$	—	—
$\text{nChanges}_{\text{max}}$	—	$2\sqrt{n}$	$2\sqrt{n}$
$\text{nCounts}_{\text{max}}$	1	1	100

in MSA $nChanges_{\text{max}}$ inner iterations, at the same temperature T_ϑ, by making $nChanges_{\text{max}}$ random substitutions on X. At the current temperature T_ϑ, the first inner iteration provides Y made with 1 random substitution over X. The second inner iteration provides another Y made with 2 substitutions over X. The last, $nChanges_{\text{max}}$ iteration provides another Y created from X by making $nChanges_{\text{max}}$ substitutions over X. In accordance with SA algorithm, at the same temperature more neighbor solutions Y are created with MSA algorithm. The current solution X is changed accordingly with the SA algorithm, for all Y solutions.

4.4. Iterated modified simulated annealing (IMSA)

(IMSA) is a heuristic algorithm. IMSA algorithm inherits features of MSA algorithm. In order to find better solution, IMSA algorithm has outer loop in which MSA repeats $nCounts_{\text{max}}$ times. Chances for finding a better optimum are higher when the execution time needed for finding suboptimal solution is longer or the number of iterations is greater. Algorithm 1 illustrates IMSA algorithm.

Functions used in IMSA algorithm, shown in Algorithm 1, are:

Initialize: sets parameters to their default user defined values.

InitialSolution: returns an initial feasible route.

Feasible: returns TRUE if the route is feasible, otherwise returns FALSE.

Dist: returns route length.

RandomNeighborSequence: returns neighbor sequence Y that differs from the current sequence X on *nChanges* made.

RandomNumber: returns a real number for a given interval.

InitialSolution is based on Greedy-random sequence (GRS) algorithm. In order to create an initial solution, GRS starts from the start node, also denoted as current node. From current node, all unvisited neighbor nodes, denoted as next node, are considered with respect to their vehicle capacity Q constraint and the route length d. The feasible neighbors, which satisfy vehicle capacity constraint, are sorted with respect to travel length from the current node to the next node. The neighbour with the shortest path would be selected with pure greedy algorithm. However, GRS algorithm calculates initial solution from the graph. Starting with the current node, GRS selects feasible neighbor nodes with respect to vehicle cargo. Again, the nearest feasible neighbor would be selected with the pure greedy algorithm. But GRS selects the nearest feasible neighbor solution with 80% probability and the second feasible solution with 20% probability. This way, the greedy algorithm becomes random.

The values of parameters are the same for all variations of SA algorithm. Starting temperature T_{start} has value 1000, from which the system anneals until the temperature reaches final temperature $T_{\text{end}} = 10$. Constant L with default value 100 represents the number of neighbor solutions Y considers at the same temperature T. A bounded number of iterations

B is set to be 1000 in order to prevent a long execution time if the input temperature interval $[T_{\text{start}}, T_{\text{end}}]$ is too wide. Temperature reducing factor α equals 1.1 in order to decrease the temperature value by 10% in the next iteration, and is usually set to be a bit greater than 1.

In order to get the basic SA algorithm, the following changes need to be implemented in the algorithm from Algorithm 1. Line 6 does not exist in the SA algorithm, and it should be removed. Accordingly, parameter $Changes_{\max}$ in line 10 should be replaced with *nSAChanges*. Parameter $Counts_{\max}$, in line 7, should be set to value 1 or the line 7 should be removed, respectively. The MSA algorithm can be extracted from IMSA illustrated in Algorithm 1, by removing the line 6 of the proposed algorithm. Thus, implementation of the SA algorithm is simpler. It uses a fixed number of random substitutions *nSAChanges*, while MSA and IMSA have an iteratively changeable number of changes from 1 to $nChanges_{\max}$. MSA differs from IMSA in the number of $Counts_{\max}$ used to repeat the MSA algorithm. Finally, a small solution space is explored with SA, more is explored with MSA, and even more with IMSA algorithm. Parameters that are different for different methods are illustrated in Table 1.

5. Simulation environment

For the purpose of experimental analysis, we developed a simulation environment using *MSVisual C++* 6.0. Figure 3 depicts functional components and the usage of the simulation environment. During the *initialization* phase in step 1, the graph with or without cargo, vehicle capacity Q, and common simulation parameters are defined. In order to search for an optimal solution, the methods discussed in the previous section are available in the *search method* phase, simulation step 2. When a suboptimal solution is found, output parameters, route length d and execution time t, are evaluated in simulation step 3.

A solution improvement method can be applied by using an interactive GUI. An interactive GUI enables a user to improve the route performances in simulation step 4 (Figure 3). By selecting a subset of vertices $M \subseteq V$ on a route, the *search improvement method* provides some local improvements of selected subroutes. For example, let $M = \{a, b, c, d, e, f, g\}$ be a set of user-selected vertices. Such set M represents an input parameter for the solution improvement method that combines selected vertices in the resulting route. In that way, neighbor solutions Y differ from solution X only by vertices from set M, as illustrated in Figure 4.

In this paper, EPA that is defined in Section 4, is used to enumerate the routes with improved performance, if such routes exist. After the *suboptimal solution* phase in simulation step 3 (Figure 3), a suboptimal route can be improved by selecting vertex set M and starting the next simulation phase. Vertices $v \in M$ from the suboptimal route are then permuted in the *solution improvement method* phase in step 5 (Figure 3). If the new route is feasible and shorter than the previous one, then it is set to be the route with improved performance.

6. Experimental results

We experimented with the simulation environment on a PC with Intel T7200 processor under Windows XP. We chose several instances from [13] with positive demand at the depot. In this way, we considered real world problems with the vehicle starting loaded or empty at the depot. We compare the results with those from [13].

Figures 5–7 illustrate results for the graph instance with 25 vertices and vehicle capacities 10, 15, and 20. Circles represent vertices, and arrows represent the route. Numbers in vertices represent the amount of product $q_i > 0$ that needs to be delivered, and $q_i \leq 0$

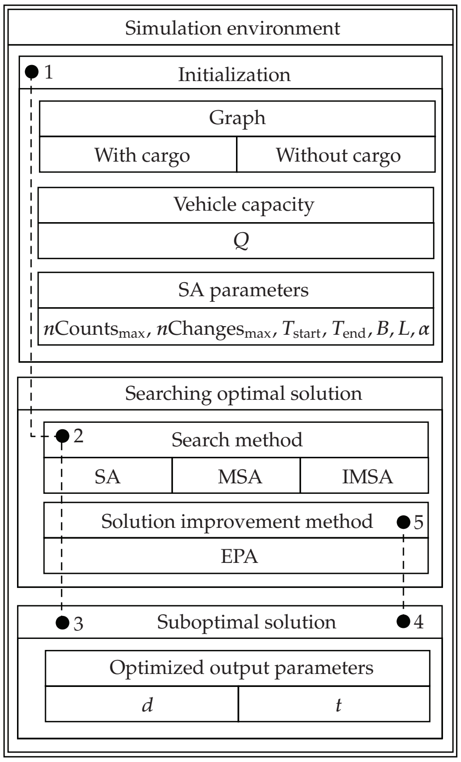

Simulation environment
1
Initialization
Graph
With cargo
Without cargo
Vehicle capacity
Q
SA parameters
$n\text{Counts}_{\max}, n\text{Changes}_{\max}, T_{\text{start}}, T_{\text{end}}, B, L, \alpha$
Searching optimal solution
2
Search method
SA
MSA
IMSA
Solution improvement method
5
EPA
3
Suboptimal solution
4
Optimized output parameters
d
t

Figure 3: Simulation environment.

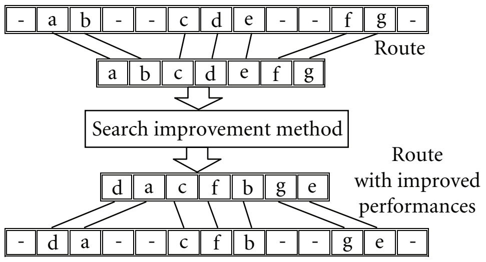

- a b - - c d e - - f g -
Route
a b c d e f g
Search improvement method
d a c f b g e
Route with improved performances
- d a - - c f b - - g e -

Figure 4: Route improvement method based on an interactive GUI.

indicates that the product needs to be collected. Shaded circle with demand 0 is the depot vertex and corresponds to the vehicle start and end positions. Figure 5 illustrates an optimal route for the smallest vehicle capacity value $Q = 10$. Figures 6 and 7 represent the resulting routes for $Q = 15$ and $Q = 20$, respectively. The longest route length corresponds to the smallest vehicle capacity. Larger vehicle capacity results in a shorter route.

Possible practical application of such a graph representation is when there are several places with same sort of material storage and other places that demand such material. For instance, the 1-VRPPD application may be used in newspaper redelivery service. A newspaper company puts a newspaper in circulation, that is, they produce certain amount of newspapers and deliver them to the newspaper stands. In the morning, shops receive newspapers and sell them during the day with different quantities. In the end of the day,

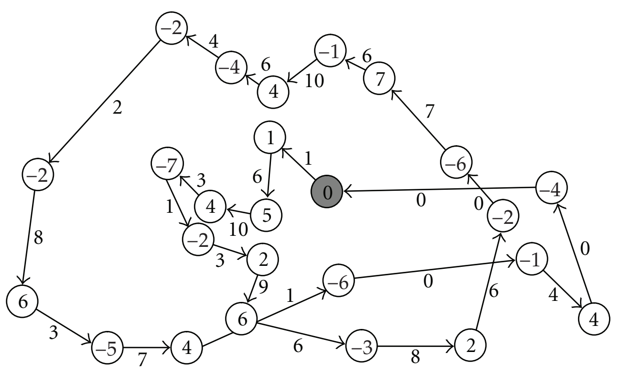


Figure 5: IMSA route when $Q = 10$.

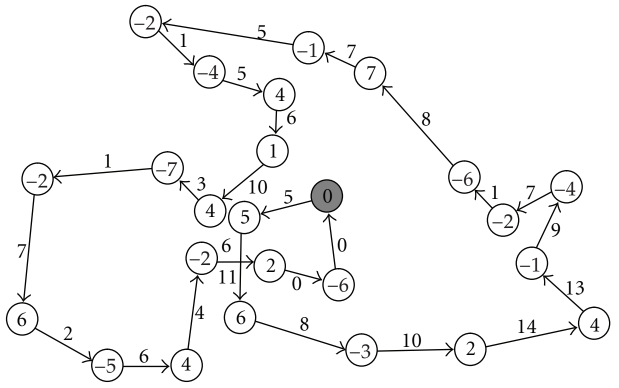


Figure 6: IMSA route when $Q = 15$.

newspapers are not appropriate for tomorrow sales. In order to reduce company losses, shops that posses lesser newspaper quantities require newspapers from shops that posses more newspapers in stock. A small vehicle with capacity Q, has a task to redeliver newspapers among newspaper stands.

The heuristic algorithm presented in Algorithm 1 is appropriate for solving combinatorial optimization problems, which may be $\mathcal{NP}$-hard. Besides 1-VRPPD, other problems can be solved with proposed algorithm. Such problems should have feasible initial solution, which is later improved when solution space is more explored. Inputs to the algorithm are: the algorithm parameters, a graph with vertexes which represents customers and their demands, and finally a depot vertex which represents start and goal position. As a result, the proposed algorithm always returns a Hamiltonian cycle with a suboptimal distance between vertexes.

Another practical application is in urgent medicament transport, when several hospitals have more medicaments of one type than the others do. Practical transportation of H5N1 medicament is feasible in an unpredictable epidemical contamination. Another practical application is in concrete industry where customers are served with a truck which can load concrete at several secondary depots, that is, at Pickup customers, and unload it according to customer demands. Another practical application is a sensor network, where sensors need to be transported in order to improve network efficiency. This is the case in

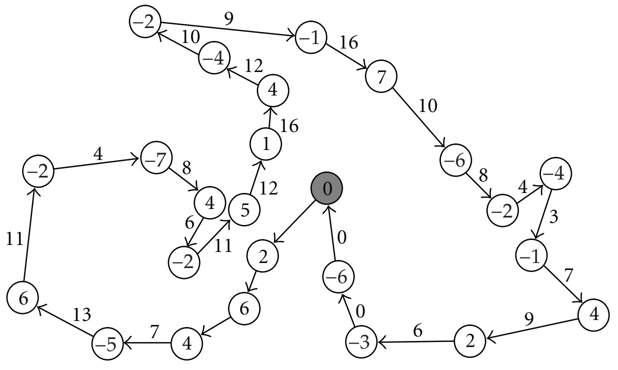


Figure 7: IMSA route when $Q = 20$.

military and meteorology applications. Due to some natural forces, a sensor network may be interrupted. Thus, our algorithm may reduce network sensor transportation expenses.

When vehicle capacity Q tends to infinity, 1-VRPPD requires less computational power than in cases where vehicle capacity Q is small. According to SA execution times shown in Table 2, feasible solutions are found quicker when vehicle capacity Q is larger. Thus for the same graph, when SA is used, execution times are highest when capacity constraints are stringent, when Q is smaller (Figure 5). As mentioned before, MSA has larger neighborhood space to explore than SA. With the number of customers $n > 50$, the average route length and execution time is smaller when vehicle capacity is larger. Consequently, IMSA average route length d is the smallest for largest Q. According to Table 2, the time needed for enumeration of the optimal feasible route is proportional to the size of explored neighborhood space.

Table 2 presents our experimental results, where graph instance features are denoted as follows. n is the number of vertices, K is quantity of products that needs to be collected and picked-up (3.2), Q is vehicle capacity, $d*$ is best solution found, *rsd* is the route length's relative standard deviation and d and t are average route length and average execution time. Vehicle capacity Q has an experimental variable in range [10, 30] in steps of 5. Proposed algorithm is random based and average values obtained from 100 experiments.

Route calculations with the IMSA last longer than with the MSA. Thus, using IMSA is more appropriate in the case when the user working with our application has customer demands several days before a vehicle starts with Pickup and delivery services. Then the user can setup even a daylong search strategy in order to get shorter routes. The number $nCounts_{\max}$ (Algorithm 1), different in MSA and IMSA, needs to be set appropriately in order to satisfy user needs. If the route length is more important than the execution time, $nCounts_{\max}$ must be set to a relatively large value. Otherwise, if the execution time is preferable, $nCounts_{\max}$ must be set to a relatively small value.

In this paper, the IMSA is demonstrated as the algorithm resulting in short routes, and the MSA as the algorithm with the short execution time.

In Table 3, dependencies between the number of customers and average length and execution times for EPA, SA, MSA, and IMSA are proposed. For instances, with 15 customers, EPA execution time is about 3 hours long, making EPA impractical for larger instances.

Table 3 illustrates the characteristics of methods analyzed in this paper, average values of route lengths and execution times gathered in Table 2. In this way, IMSA is presented as the

Table 2: Experimental results.

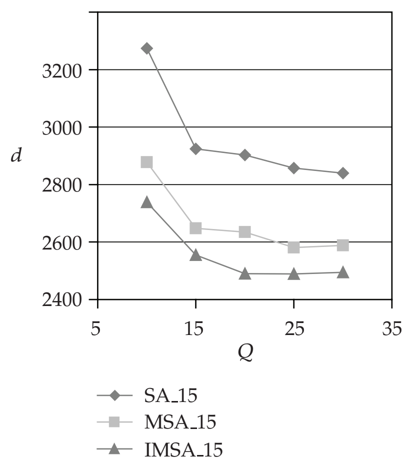
n	K	Q	SA				MSA				IMSA			
			$d*$	d	rsd (%)	t (ms)	$d*$	d	rsd (%)	t (ms)	$d*$	d	rsd (%)	t (ms)
15	31	10	2669	3274	8.19	3	2582	2878	5.44	10	2571	2739	4.28	833
15	31	15	2456	2924	9.65	2	2329	2648	7.60	11	2329	2555	6.30	938
15	31	20	2443	2903	7.99	3	2329	2635	6.34	12	2329	2490	5.42	1081
15	31	25	2443	2857	8.25	2	2329	2581	6.67	12	2273	2489	5.43	1158
15	31	30	2421	2840	9.30	3	2273	2588	6.40	13	2273	2495	4.56	1175
25	45	10	4254	5001	9.92	4	3926	4565	8.20	23	3734	4258	6.02	1888
25	45	15	3921	4611	7.04	4	3568	4201	6.47	23	3541	3995	6.51	2088
25	45	20	3352	4545	9.69	5	3270	3920	8.09	24	3332	3882	7.46	2313
25	45	25	3273	4449	11.75	4	3418	4008	7.05	27	3397	3907	6.49	2562
25	45	30	3892	4592	10.32	4	3203	3923	8.56	28	3270	3863	7.14	2637
40	66	10	5954	7042	7.31	3	5433	6282	6.07	31	5639	6072	4.78	2782
40	66	15	5460	6551	8.62	5	5128	5850	5.94	33	5120	5650	5.38	2983
40	66	20	5259	6383	7.58	4	4977	5685	6.59	36	4953	5466	5.45	3251
40	66	25	5273	6429	7.55	5	4660	5496	5.86	38	4582	5396	6.01	3753
40	66	30	5206	6405	9.46	5	4618	5480	6.83	41	4738	5296	5.81	3934
55	84	10	7662	8691	6.06	5	6944	7944	6.57	44	6695	7501	5.94	3897
55	84	15	6406	7733	8.42	6	6192	7216	6.11	46	6097	6776	5.34	4173
55	84	20	6379	7564	7.51	6	5787	6777	6.31	50	5689	6615	5.88	4365
55	84	25	6563	7433	7.00	6	5691	6463	6.47	55	5786	6353	5.06	5392
55	84	30	5980	7228	7.98	6	5799	6557	6.09	58	5502	6262	5.91	5951
70	113	10	8477	9792	6.69	7	8423	9630	7.61	69	7899	8882	6.48	5240
70	113	15	8042	9116	7.23	7	7421	8528	6.18	61	6867	8226	7.61	5479
70	113	20	7887	9150	6.83	7	7256	8371	5.62	64	7136	8026	5.32	5778
70	113	25	7600	8762	6.12	7	7189	8182	5.48	69	7064	7783	5.45	6473
70	113	30	7578	8828	7.58	9	7004	8048	6.38	75	6787	7865	6.07	6988
Average values			5234	6204	8.16	5	4870	5618	6.60	38	4784	5394	5.84	3484

Table 3: Average route lengths and execution times based on 100 experiments.

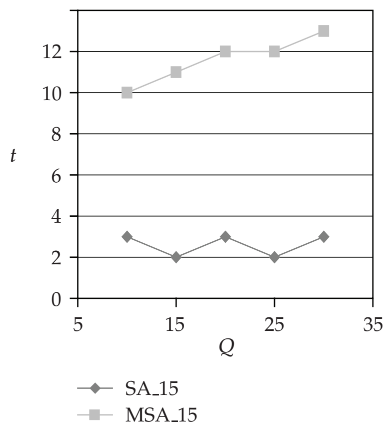
n	$d*$				d				t (ms)			
	EPA	SA	MSA	IMSA	EPA	SA	MSA	IMSA	EPA	SA	MSA	IMSA
15	**2273**	2486	2368	2355	**2273**	2960	2666	2554	$1 \cdot 10^7$	**3**	12	1037
25	—	3738	3477	**3455**	—	4640	4123	**3981**	—	**4**	25	2298
40	—	5430	4963	**5006**	—	6562	5759	**5576**	—	**4**	36	3341
55	—	6598	6083	**5954**	—	7730	6991	**6701**	—	**6**	51	4756
70	—	7917	7459	**7151**	—	9130	8552	**8156**	—	**7**	68	5992

algorithm that has the shortest route length and the longest execution time. The SA algorithm is the algorithm with the shortest execution time and the longest route length.

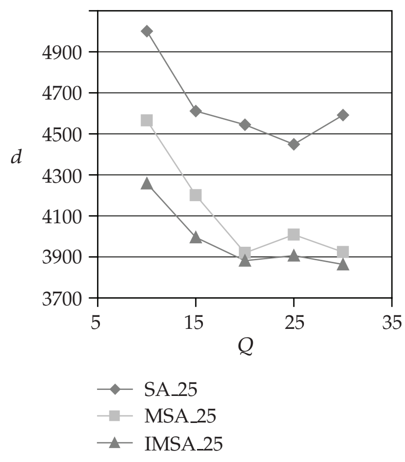
Dependency between average route length d and vehicle capacity is proposed in Figure 8. Figure 8 represents dependencies between average execution time t and vehicle capacity Q. Figures 9(a) and 9(b) illustrate the values of average lengths and execution times with respect of the number of customers.



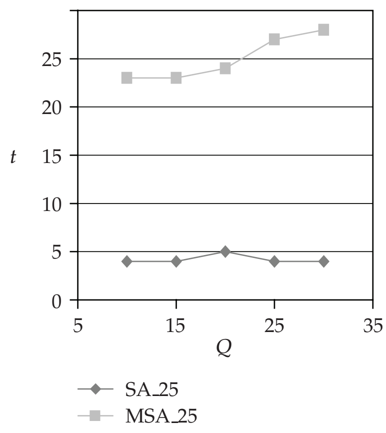
(a)



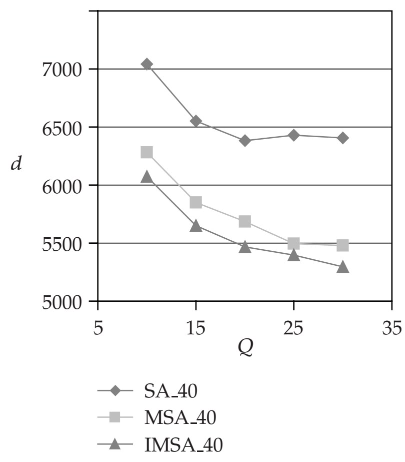
(b)



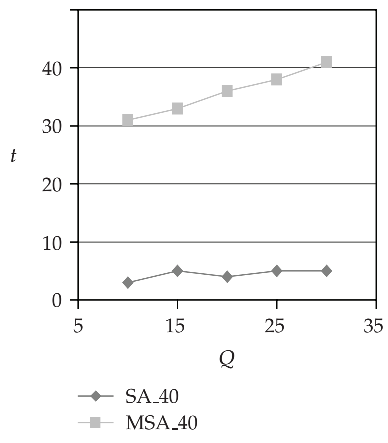
(c)



(d)

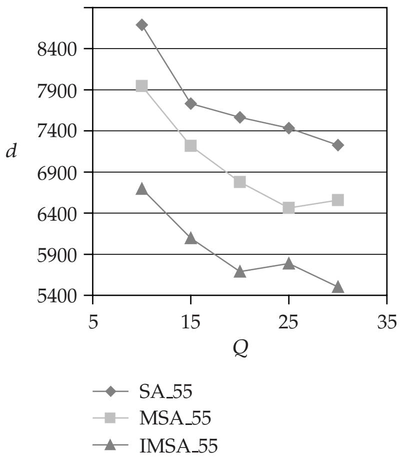


(e)

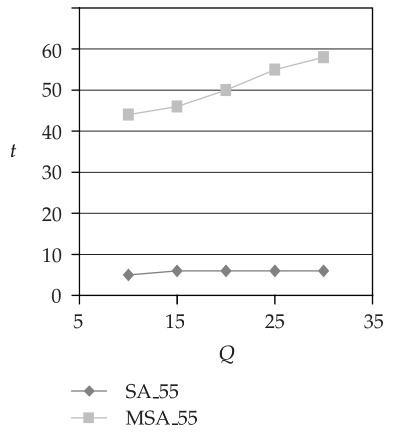


(f)

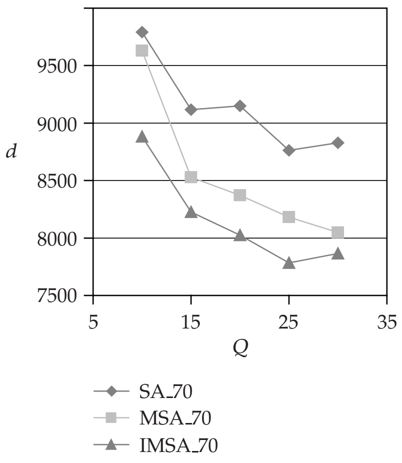
Figure 8: Continued.



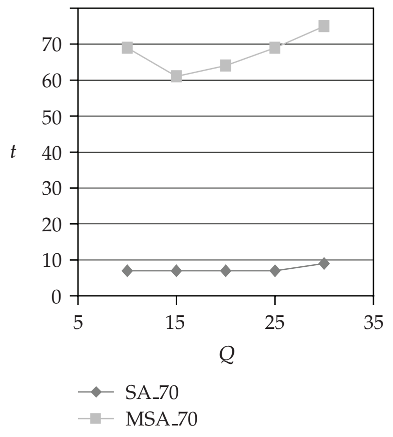
(g)



(h)



(i)



(j)

Figure 8: Average route length d (pixels) and average execution time t (ms) output parameters for various graph instances with variable vehicle capacity Q values.

The chart in Figures 8(a), 8(c), 8(e), 8(g), 8(i) illustrate how a change in vehicle capacity affects average route length d. The legend in Figure 8 illustrates which method is used and how many customers are involved in a single route. The smallest instance with 15 customers is marked as SA_15, MSA_15 and IMSA_15, respectively. Other instances with a different number of customers are marked accordingly. In the charts Figures 8(a), 8(c), 8(e), 8(g), 8(i), we see that the route length becomes shorter as the vehicle capacity growths. This is the case for all methods applied. Figure 9(a) illustrates the average route length values shown in Table 3. It shows that the average route length depends linearly on a number of customers that needs to be served. Route lengths are highest in case of SA, than there comes MSA, and finally the smallest is for IMSA. This means that SA provides suboptimal routes that are on average the longest suboptimal routes. IMSA routes are on average the shortest

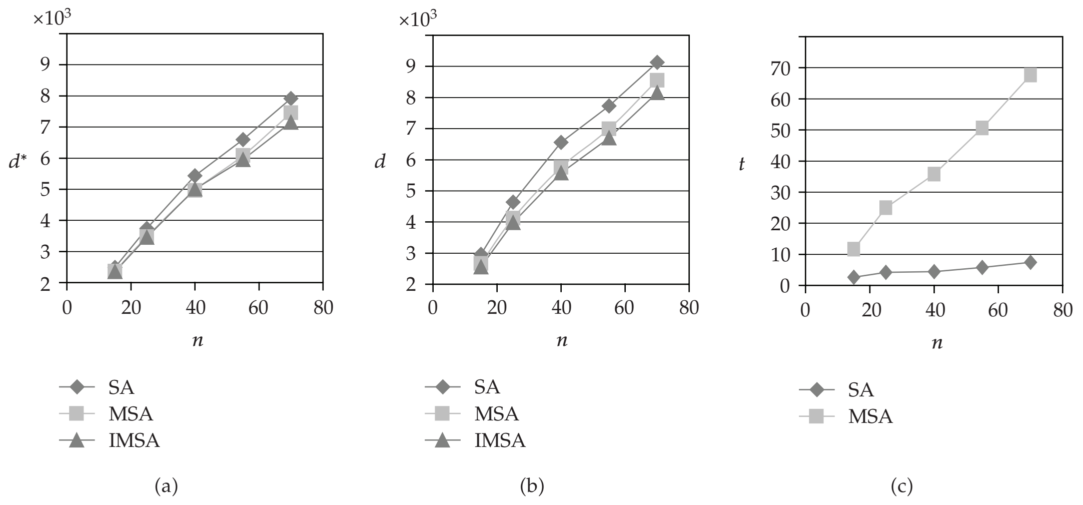


Figure 9: Best found route length $d*$ (pixels), average route length d (pixels) and average execution time t (ms) output parameters for various graph instances with variable n number of customers.

Table 4: IMSA algorithm output parameters compared with heuristic algorithm [13].

Name	n	K	Q	H		MSA			IMSA		
				d	t (ms)	$d*$	d	t (ms)	$d*$	d	t (ms)
n30q20B	30	62	20	**5109**	90	5883	6714	**25**	5778	6368	2158
n30q20E	30	66	20	**4916**	60	6545	7641	**23**	6473	7047	2101
n40q20B	40	85	20	**5334**	110	6031	6928	**33**	5914	6817	2993
n40q20I	40	90	20	**5262**	140	7299	8160	**33**	6274	7927	2922
n50q20A	50	99	20	**5908**	180	7128	8211	**45**	7046	7985	4158
n50q20C	50	112	20	**6962**	460	8616	9984	**42**	8575	9477	3857
n60q20A	60	126	20	**6696**	430	8059	9321	**45**	7763	8889	4136
n60q20B	60	146	20	**6730**	410	8663	9641	**44**	7959	9349	4012
n30q40B	30	62	40	**4529**	50	5421	6359	**30**	5130	6038	2751
n30q40E	30	66	40	**4822**	50	5530	6299	**30**	5451	6111	2795
n40q40B	40	85	40	**5315**	90	6028	6697	**40**	5802	6719	3987
n40q40I	40	90	40	**4967**	90	6026	7476	**41**	5865	7125	3791
n50q40A	50	99	40	**5816**	140	6867	8069	**58**	6849	7691	5671
n50q40C	50	112	40	**6284**	190	7400	8711	**48**	7359	8589	4493
n60q40A	60	126	40	**6156**	240	6839	8394	**61**	6729	8072	5699
n60q40B	60	146	40	**6524**	230	7643	8959	**54**	7328	8607	5199
Average values . . .				5708	185	6874	7973	41	6643	7676	3795

ones. Resulting MSA routes are on average little longer than the IMSAs, while they are much shorter than the SAs suboptimal routes.

The charts in Figures 8(b), 8(d), 8(f), 8(h), 8(j) illustrate how vehicle capacity affects execution time t. Since IMSAs execution lasts much longer than SA and MSA, the execution time for IMSA is not displayed in Figures 8 and 9. As illustrated in Figure 9(b), the execution time of SA is little shorter than of MSAs. The shortest execution times are on average for the

SA search method. MSAs execution times are little longer than SA's, and much shorter than IMSA's, (see Table 3).

Results proposed in Table 4 compare our route length results with heuristic algorithm used in [13]. H represents heuristic algorithm proposed in [13], $d*$ is the shortest route length, d is an average route length, and t is the average execution time. Used graph instances with Pickup and delivery demands are available at [18]. We chose instances with positive vehicle capacity at the depot. These instances are presented in Table 4. Among tested instances, MSA and IMSA algorithms have longer route lengths, while execution times are shorter with MSA algorithm for several instances. As the number of customer increases, compared results have proportional growth of average execution time.

As illustrated in Table 4, IMSA heuristics is the algorithm that can solve 1-VRPPD and is comparable with heuristic H presented in [13]. MSA has 450% shorter execution time than H, while H provides 35% shorter route length. IMSA repeats MSA for 100 times. Thus, IMSA execution time is approximately 100 times longer than MSA.

7. Conclusion

In this paper, we have presented the (IMSA) algorithm. IMSA is an iterative approach heuristic algorithm used for solving the combinatorial optimization problem of 1-VRPPD. Both MSA and IMSA inherit features of the simulated annealing (SA) algorithm with different calculation of neighbor solutions.

After the suboptimal route is calculated, several local improvements along the route are possible. By using an interactive graphical user interface, a user has the ability to select a sub graph on which solution improvement method is applied. The exact permutation algorithm (EPA) was used as a solution improvement method. In order to have the results in reasonable time interval, selected sub graph should have less than 15 nodes, due to the time complexity of EPA. Such improvements are useful because of large search space of 1-VRPPD, since 1-VRPPD generalize VRP, which is already $\mathcal{NP}$ hard. As the experimental results substantiates, when the execution time is long enough or if the number of iterations is large enough, MSA and IMSA provide a good approximation to the global optimum in large search space.

References

[1] H. Hernández-Pérez and J.-J. Salazar-González, "A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery," *Discrete Applied Mathematics*, vol. 145, no. 1, pp. 126–139, 2004.

[2] G. M. Giaglis, I. Minis, A. Tatarakis, and V. Zeimpekis, "Minimizing logistics risk through real-time vehicle routing and mobile technologies: research to date and future trends," *International Journal of Physical Distribution & Logistics Management*, vol. 34, no. 9, pp. 749–764, 2004.

[3] B. L. Hollis, M. A. Forbes, and B. E. Douglas, "Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post," *European Journal of Operational Research*, vol. 173, no. 1, pp. 133–150, 2006.

[4] G. Jeon, H. R. Leep, and J. Y. Shim, "A vehicle routing problem solved by using a hybrid genetic algorithm," *Computers & Industrial Engineering*, vol. 53, no. 4, pp. 680–692, 2007.

[5] G. Mosheiov, "Vehicle routing with pick-up and delivery: tour-partitioning heuristics," *Computers & Industrial Engineering*, vol. 34, no. 3, pp. 669–684, 1998.

[6] J. Brandão, "A new tabu search algorithm for the vehicle routing problem with backhauls," *European Journal of Operational Research*, vol. 173, no. 2, pp. 540–555, 2006.

[7] H. Min, "The multiple vehicle routing problem with simultaneous delivery and pick-up points," *Transportation Research Part A*, vol. 23, no. 5, pp. 377–386, 1989.

[8] F. A. Tang Montané and R. D. Galvão, "A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service," *Computers & Operations Research*, vol. 33, no. 3, pp. 595–619, 2006.
[9] A. G. H. Kek, R. L. Cheu, and Q. Meng, "Distance-constrained capacitated vehicle routing problems with flexible assignment of start and end depots," *Mathematical and Computer Modelling*, vol. 47, no. 1-2, pp. 140–152, 2008.
[10] F. Li, B. Golden, and E. Wasil, "The open vehicle routing problem: algorithms, large-scale test problems, and computational results," *Computers & Operations Research*, vol. 34, no. 10, pp. 2918–2930, 2007.
[11] K. Fleszar, I. H. Osman, and K. S. Hindi, "A variable neighbourhood search algorithm for the open vehicle routing problem," *European Journal of Operational Research*, vol. 195, no. 3, pp. 803–809, 2009.
[12] K. Ganesh and T. T. Narendran, "CLOVES: a cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up," *European Journal of Operational Research*, vol. 178, no. 3, pp. 699–717, 2007.
[13] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, "An exact algorithm for the traveling salesman problem with deliveries and collections," *Networks*, vol. 42, no. 1, pp. 26–41, 2003.
[14] F. Wang, A. Lim, and Z. Xu, "The one-commodity pickup and delivery travelling salesman problem on a path or a tree," *Networks*, vol. 48, no. 1, pp. 24–35, 2006.
[15] P. Toth and D. Vigo, Eds., *The Vehicle Routing Problem*, vol. 9 of *SIAM Monographs on Discrete Mathematics and Applications*, SIAM, Philadelphia, Pa, USA, 2002.
[16] "Calculating Permutations and Job Interview Questions," September 2008, http://www.bearcave.com/random_hacks/permute.html.
[17] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[18] Pickup-and-Delivery Site: http://webpages.ull.es/users/hhperez/PDsite/index.html.