AN EFFICIENT ITERATED LOCAL SEARCH ALGORITHM FOR THE VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICKUP AND DELIVERY

Anand Subramanian, Luiz Satoru Ochi
Universidade Federal Fluminense (UFF)
Instituto de Computação, Rua Passo da Pátria 156 - Bloco E - 3º andar, São Domingos Niterói - RJ - CEP: 24210-240 – Niterói, RJ.
{anand, satoru}@ic.uff.br

Lucídio dos Anjos Formiga Cabral
Universidade Federal da Paraíba (UFPB)
Departamento de Estatística, Centro de Ciências Exatas e da Natureza – Campus I – Cidade Universitária, Castelo Branco, CEP: 58051-900 – João Pessoa, PB.
lucidio@de.ufpb.br

ABSTRACT

This paper deals with the Vehicle Routing Problem with Simultaneous Pickup and Delivery (VRPSPD). A procedure based on the Iterated Local Search (ILS) metaheuristic that uses the Variable Neighborhood Descent (VND) method for performing the local search is proposed. According to literature, the most successful algorithms for the VRPSPD are pure or hybrid versions of the Tabu Search metaheuristic. Our objective here is to show that the ILS can also produce highly competitive results. The algorithm developed was tested on benchmark problems available in the VRPSPD literature and it was found capable of improving several of the known solutions.

KEYWORDS. Vehicle Routing. Metaheuristics. Hybrid Heuristics.

RESUMO

Este artigo trata do Problema de Roteamento de Veículos com Coleta e Entrega Simultânea (PRVCES). Um procedimento baseado na metaheurística *Iterated Local Search* (ILS) que utiliza o método de descida em vizinhança variável (VND) para realização da busca local é proposto. De acordo com a literatura, as melhores heurísticas para o PRVCES são versões puras ou hibridas do método Busca Tabu. O objetivo deste trabalho é mostrar que o método ILS também pode produzir resultados altamente competitivos. O algoritmo desenvolvido foi testado em instâncias encontradas na literatura relativas ao PRVCES, tendo sido capaz de melhorar algumas das soluções conhecidas.

PALAVRAS CHAVE. Roteamento de Veículos. Metaheuristicas. Heurísticas Híbridas.

1. Introduction

The Vehicle Routing Problem (VRP) is a well-known combinatorial optimization problem proposed in the late 1950's and it is still one of the most studied in the field of Operations Research. The great interest in the VRP is due to its practical importance as well as the high level of difficulty in solving it.

Recognized as NP-hard, the VRP with Pickup and Delivery (VRPPD), i.e., the problem where objects or people should be collected and distributed, constitutes an important class of the VRP (BERBEGLIA *et al.*, 2007). The most VRPPD variants studied in the literature are: VRP with Backhauls, VRP with mixed Pickup and Delivery; Dial-a-ride Problem; and the VRP with Simultaneous Pickup and Delivery (VRPSPD).

In this work, our interest lies on the VRPSPD, which can be described as follows. Let $G = (V, E)$ be a complete and directed graph with a set of vertices $V = \{0,...,n\}$, where the vertex 0 represents the depot ($V_0 = \{0\}$) and the remaining ones the clients. Each edge $(i, j) \in E$ has a non-negative cost c_{ij} satisfying the triangular inequality. Each client $i \in V - V_0$ has a demand $q_i \in D$ for delivery and $p_i \in P$ for pickup, where D and P are the sets containing the amount of a certain cargo (or people) to be distributed and collected respectively. Let $C = \{1,..., m\}$ be the set of available vehicles with capacity Q . The VRPSPD consists in constructing a set up to m routes with the following requisites: (i) all the pickup and delivery demands should be accomplished; (ii) the capacity of the vehicle should not be exceeded; (iii) only one vehicle can visit a determined client; (iv) the sum of costs should be minimized.

The algorithm proposed in this paper consists of an extension of the heuristic developed by Subramanian and Cabral (2008) for the VRPSPD with time limits to accomplish a route. Two new perturbation mechanisms are incorporated into this procedure.

The rest of the paper is organized as follows. Section 2 lists some works related to the VRPSPD. Section 3 deals with the proposed algorithm, describing the constructive and local search procedures, as well as the perturbation mechanisms adopted. Section 4 contains the results obtained and a comparison with the ones found in the literature. Section 5 presents the concluding remarks of this work.

2. Related Works

The Chart 1 illustrates some VRPSPD related works, describing their main contributions and/or approaches.

Chart 1 – VRPSPD related works

Work	Contribution and/or Approach
Min (1989)	First Work Case study in a public library
Salhi and Nagy (1999)	Insertion heuristics
Dethloff (2001)	Insertion heuristic based on the cheapest feasible insertion criterion, radial surcharge and residual capacity
Angelelli and Mansini (2003)	Branch-and-price for the VRPSPD with Time Windows
Vural (2003)	Genetic Algorithm
Gokçe (2003)	Ant Colony
Röpke and Pisinger (2004)	Large Neighborhood Search
Nagy and Salhi (2005)	Heuristics with different levels of feasibility
Cripsim and Brandão (2005)	Tabu Search + VND Algorithm
Dell'Amico *et al.* (2005)	Branch-and-price based on dynamic programming
Chen and Wu (2006)	Record-to-record travel + Tabu Lists
Montané and Galvão (2006)	Tabu Search Algorithm
Gribkovskaia *et al.* (2007)	Tabu Search for the VRPSPD with a single vehicle
Bianchessi and Righini (2007)	Constructive and Local Search Heuristics Tabu Search with variable neighborhood search

Subramanian and Cabral (2007)	GRASP + VND Heuristic
Zachariadis *et al.* (2007)	Tabu Seach + Guided Local Search
Wassan *et* al. (2008)	Reactive Tabu Search
Subramanian and Cabral (2008)	An ILS-VND Heuristic for the VRPSPD with Time Limit Perturbation utilized: Double-Bridge

From Chart 1, it can be seen that the interest in the VRPSPD considerably grew up in the past decade. Among the different approaches proposed, the heuristics are the most used so far. In addition, it is possible to verify that, in the last five years, the metaheuristics are being widely employed in the literature. One the other hand, the exact strategies have not been much explored if compared to the heuristics methods.

3. Proposed Algorithm

The proposed algorithm (ILS-VND) works as follows. The procedure is executed *MaxIter* times, where an initial solution is generated by a greedy heuristic and then it is improved by a procedure based on the ILS metaheuristic which realizes a local search by means of a VND (*Variable Neighborhood Descent*) heuristic.

```
Procedure ILS-VND(MaxIter, MaxIterILS, seed, γ, v, N(.), f(.), r)
LoadData( );
f* ← ∞;
for k = 1,..., MaxIter do
    s ← GenerateInitialSolution(γ, seed, v);
    s' ← s;
    iterILS ← 0;
    while iterILS < MaxIterILS do
        s ← VND(N(.), f(.), r, s);  {r = nº of neighborhoods}
        if f(s) < f(s')
            s' ← s;
            f(s') ← f(s);
            iterILS ← 0;
        end if;
        s ← Perturb(s');
        iterILS ← iterILS + 1;
    end while;
    if f(s') < f*
        s* ← s';
        f* ← f(s');
    end if;
end;
return s*;
End ILS-VND
```

Figure 1 – Pseudocode of the ILS-VND algorithm

The pseudocode is described in Fig.1, where s^* corresponds to the best solution, v is the number of vehicles (or routes) imputed, N is the set of neighborhood structures and γ is a parameter treated in detail in Subsection 3.1.

3.1 Constructive Procedure

The method employed for building a feasible initial solution involves a greedy approach and is an adaptation of Dethloff's insertion heuristic. The pseudocode is described in Fig. 2.

To begin with, the number of vehicles v to be considered for constructing the initial solution is pre-determined. Then, all routes are filled with a client e, chosen at random from the

Candidate List (CL). Later, the clients belonging to the CL are evaluated according to the insertion criterion expressed by eq. (1).

```
Procedure GenerateInitialSolution (seed, γ, v)
 s ← Ø;
Initialize the Candidate List (CL);
Let s = {s^1, s^2,...,s^v}  be the set composed by  v empty routes;
t ← 1;
while t ≤ v do
     s^t ← e ∈ LC randomly selected;
     Update LC;
     t ← t + 1;
end while;
while LC ≠ Ø do
Evaluate the value of each cost g(e) for e ∈ LC;
     g^min ← min{g(e) | e ∈ LC};
     n ← client e associated to g^min;
     s ← s ∪ {n};
     Update LC;
end while;
return  s ;
Fim GenerateInitialSolution
```

Figure 2 – Pseudocode of the constructive procedure

$$g(e^v) = (C_{ik} + C_{kj} - C_{ij}) - \gamma(C_{0k} + C_{k0}) \qquad (1)$$

The first part of eq. (1) is related to the well-known cheapest feasible insertion criterion, which consists of a greedy approach that takes into account the least additional cost regarding the insertion of the node k between the nodes i and j of the route v. Naturally, only the feasible insertions are admitted. The second part corresponds to a surcharge used to avoid late insertions of customers remotely located. The distance from the depot and back is weighted by a factor $\gamma \in [0,1]$. The client e associated to g^{min} is then added to the partial solution s. The constructive procedure ends when all the clients have been added to the solution s.

3.2 Local Search

The local search phase, responsible to improve the initial solution is performed by a heuristic based on the VND algorithm. Mladenović and Hansen (1997) proposed the variable neighborhood descent method which systematically modifies the neighborhood structures that belong to a set $N = \{N^{(1)}, N^{(2)}, N^{(3)},..., N^{(r)}\}$, in a deterministic way. The neighborhood $N^{(k')} = N^{(k+1)}$ is activated only if there is no improvement in the k neighborhoods tested before regarding improvement of the current solution.

In the proposed algorithm, a set of ten neighborhood structures are used where six of them perform exhaustive movements between clients of different routes. Just the feasible movements are admitted, i.e., the ones that do not violate the maximum load constraint. Thus, every time an improvement occurs, one should check whether this new solution is feasible or not. The other four neighborhood structures are presented in the end of this Subsection. The N set of neighborhoods is described next.

Shift(1,0) – $N^{(1)}$ – A client c is transferred from a route r_1 to a route r_2. The vehicle load is checked as follows. All nodes located before the insertion's position have their loads added by q_c (delivery demand of the client c), while the ones located after have their loads added by p_c (pick-up demand of the client c). It is worth mentioning that certain devices to avoid unnecessary infeasible movements can be employed. For instance, before checking the insertion of c in some particular route, a preliminary verification is performed in r_2 to evaluate the vehicle load before leaving, $\left(\sum_{i\in r_2} q_i\right)+q_c$, the depot and when arriving, $\left(\sum_{i\in r_2} p_i\right)+p_c$, at the depot. If the load exceeds the vehicle capacity Q, then all the remaining possibilities of inserting c in this route will be always violated.

Crossover – $N^{(2)}$ – The arc between adjacent nodes c_1 and c_2, belonging to a route r_1, and the one between c_3 and c_4, from a route r_2, are both removed. Later, an arc is inserted connecting c_1 and c_4 and another is inserted linking c_3 and c_2. The procedure for testing the vehicle load is more complex in comparison to Shift(1,0). At first, the initial (l_0) and final (l_f) vehicle loads of both routes are calculated. If the values of l_0 and l_f do not exceed the vehicle capacity Q then the remaining loads are verified through the following expression: $l_i = l_{i-1} + p_i - q_i$. Hence, if l_i surpasses Q, the movement is unfeasible.

Swap(1,1) – $N^{(3)}$ – Permutation between a node c_1 from a route r_1 and a node c_2, from a route r_2. The loads of the vehicles of both routes are examined in the same manner. For example, in case of r_2, all clients situated before the position in which c_2 was found (now replaced by c_1), have their values added by q_{c1} and subtracted by q_{c2}, while the load of the clients positioned after c_1 is increased by p_{c1} and decreased by p_{c2}.

Shift(2,0) – $N^{(4)}$ – Two consecutive nodes, c_1 and c_2, are transferred from a route r_1 to a route r_2. The vehicle load is tested likewise Shift(1,0).

Swap(2,1) – $N^{(5)}$ – Permutation of two consecutive nodes, c_1 and c_2, from a route r_1 by a node, c_3, from a route r_2. The load is verified by means of an extension of the approach used in the neighborhoods Shift(1,0) and Swap(1,1).

Swap(2,2) – $N^{(6)}$ – Permutation between two consecutive nodes, c_1 and c_2, from a route r_1 by another two consecutive c_3 and c_4, belonging to a route 2. The load is checked just as Swap(1,1).

It should be pointed out that the complexity of all of the six neighborhoods mentioned above is of the order of $O(n^2)$. Fig. 3 shows the pseudocode of the VND procedure.

```
Procedure VND(N(.), f(.), r, s)
    {Let r be the number of different neighborhood structures}
    k ← 1;                    {Current Neighborhood}
    while (k ≤ r) do
        Find the best neighbor s′ of s ∈ N^k(s);
        if f(s′) < f(s)
            then
                s ← s′;
                k ← 1;
                {intensification in the modified routes}
                s′ ← Or-opt(s);
                s″ ← 2-opt(s′);
                s‴ ← Exchange(s″);
                s⁗ ← Reverse(s‴);
                if f(s⁗) < f(s)
                    then
                        s ← s⁗;
                        f(s) ← f(s⁗);
                end if;
            else
                k ← k + 1;
        end if;
    end while;
    return s;
Fim VND
```

Figure 3 – Pseudocode of the VND algorithm

In case of improvement of the current solution, one should aim to further refine the quality of the routes that contributed to reduce the objective function, that is, those which participated in the last betterment move. Hence, four different neighborhoods are explored:

Or-*opt* – Introduced by Or (1976), where one, two or three consecutive clients are removed and inserted in another position of the route. The complexity of this movement is $O(n^2)$.
2-*opt* – Two nonadjacent arcs are removed and another two are added to form a new route. In this movement, the complexity corresponds to $O(n^2)$.
Exchange – Permutation between two nodes. The complexity of this neighborhood is $O(n^2)$.
Reverse – This movement reverses the route direction if the value of the maximum load of the corresponding route is reduced. Its complexity is $O(n)$.

3.3 Perturbation Mechanism

A set of three perturbation mechanisms, $P = \{P^{(1)}, P^{(2)}, P^{(3)}\}$, were adopted in this research. Whenever the Perturb() function is called, one of the movements described below is randomly selected.

Ejection Chain – $P^{(1)}$ – Applied by Rego and Roucairol (1996) for the classical version of the VRP, this movement was employed here as perturbation mechanism and it works as follows. A client from a route r_1 is transferred to a route r_2, next, a client from r_2 is transferred to a route r_3 and so on. The movement ends when one client from the last route r_v is transferred to r_1. The clients are chosen at random.
Double Swap(1,1) – $P^{(2)}$ – Two Swap(1,1) movements are performed in sequence randomly.
Double Bridge – $P^{(3)}$ – Introduced by Martin *et al.* (1991) this perturbation was originally developed for the Traveling Salesman Problem (TSP) and consists in cutting four edges of a given route and inserting another four. In principle, the movement is applied to all routes. When there are a large number of routes involved the perturbation is applied only in some of them. Lourenço *et al.* (2002) state that several applications of the ILS for the TSP have employed this type of perturbation, and it has been noted to be effective for different instance sizes.

4. Computational Results

The proposed algorithm ILS-VND was implemented in C++ programming language and executed in a PC Intel Core 2 Duo 2.13 GHz with 1024 MB of RAM memory and operating system Windows XP – Professional Edition.

The procedure was tested in benchmark problems found in the literature related to the VRPSPD. A comparison was made with the best known results. The number of iterations (*MaxIter*) and perturbations allowed (*MaxIterILS*), was 15 and 30 respectively. They were calibrated empirically after preliminary tests with different values. Thirty executions were performed for each one of the different parameterizations of γ.

The results found by the ILS-VND in the instances generated by Dethloff (2001) are shown in Table 1, where v_i represents the number of vehicles initially imputed and v_f the number of vehicles associated with the final solution. Table 2 shows a comparison between the solutions obtained by ILS-VND and the best ones reported in the literature (as per our knowledge), namely those found by Ropke and Pisinger (2004) and Zachariadis *et al.* (2007).

Table 1 – Results obtained in Dethloff's Instances

Instance	Nº of clients	v_i	v_f	Best Sol.	Time (s)	Avg. Sol.	*Gap* (%)	γ	Avg. Time (s)
SCA3-0	50	4	4	636.06	3.87	636.25	0.03	0.5	4.82
SCA3-1	50	4	4	697.84	4.36	697.84	0.00	0.0	5.20
SCA3-2	50	4	4	659.34	4.70	659.34	0.00	0.0	5.33
SCA3-3	50	4	4	680.04	4.11	680.04	0.00	0.0	5.08
SCA3-4	50	4	4	690.50	5.03	690.50	0.00	0.0	6.03
SCA3-5	50	4	4	659.90	4.48	659.90	0.00	0.2	5.14

SCA3-6	50	4	4	651.09	4.53	651.09	0.00	0.0	5.83
SCA3-7	50	4	4	659.17	5.48	663.50	0.66	0.9	6.04
SCA3-8	50	4	4	719.47	4.23	719.47	0.00	0.0	4.95
SCA3-9	50	4	4	681.00	4.36	681.00	0.13	0.0	4.69
SCA8-0	50	9	9	961.50	6.41	964.97	0.36	0.0	7.58
SCA8-1	50	9	9	1049.65	5.67	1050.07	0.04	0.8	7.29
SCA8-2	50	9	9	1039.64	9.63	1040.63	0.10	0.2	11.11
SCA8-3	50	9	9	983.34	6.61	984.26	0.09	1.0	7.50
SCA8-4	50	9	9	1065.49	5.33	1066.09	0.06	0.5	6.28
SCA8-5	50	9	9	1027.08	6.06	1031.20	0.40	0.9	6.72
SCA8-6	50	9	9	971.82	6.11	972.29	0.05	0.5	6.97
SCA8-7	50	9	9	1051.28	17.19	1057.23	0.57	0.5	19.31
SCA8-8	50	9	9	1071.18	4.80	1071.18	0.00	0.0	5.77
SCA8-9	50	9	9	1060.50	7.30	1062.01	0.14	0.6	8.00
CON3-0	50	4	4	616.52	5.30	617.67	0.19	1.0	6.15
CON3-1	50	4	4	554.47	4.05	554.52	0.01	0.9	4.99
CON3-2	50	4	4	519.11	5.45	520.13	0.20	0.6	5.58
CON3-3	50	4	4	591.19	4.44	591.19	0.00	0.0	5.80
CON3-4	50	4	4	588.79	4.98	589.76	0.16	0.9	5.71
CON3-5	50	4	4	563.70	4.53	563.77	0.01	1.0	5.66
CON3-6	50	4	4	499.05	5.03	500.25	0.24	0.8	5.79
CON3-7	50	4	4	576.48	4.64	577.09	0.11	0.6	5.72
CON3-8	50	4	4	523.05	4.92	523.05	0.00	0.9	6.25
CON3-9	50	4	4	578.24	3.92	582.43	0.72	0.8	5.45
CON8-0	50	9	9	857.17	7.03	857.65	0.06	0.7	8.30
CON8-1	50	9	9	740.85	5.48	740.87	0.00	0.9	6.52
CON8-2	50	9	9	712.89	6.23	713.31	0.06	0.6	6.74
CON8-3	50	10	10	811.07	2.42	812.37	0.16	0.7	2.99
CON8-4	50	9	9	772.25	6.05	772.25	0.00	0.5	7.29
CON8-5	50	9	9	754.88	6.20	756.92	0.27	0.2	6,51
CON8-6	50	9	9	678.92	5.78	680.32	0.21	0.9	6.62
CON8-7	50	9	9	811.96	5.22	813.07	0.14	0.3	5.92
CON8-8	50	9	9	767.53	5.66	768.34	0.11	0.9	6.46
CON8-9	50	9	9	809.00	5.50	809.66	0.08	0.7	6.88

From Table 1 it is possible to affirm that the ILS-VND demonstrated a consistent performance, since the average gap between the best solutions and the average solutions was only 0.13% with the highest value in the instance CON3-9. It can be observed from Table 2 that among the 40 test problems, the ILS-VND has improved the results of 4 instances and equaled another 36, with an average gap of -0.01%.

Table 2 – Comparison between ILS-VND and literature results in Dethloff's instances

Instance	**Ropke and Pisinger**			**Zachariadis *et al.***			**ILS-VND**			***Gap***
	Sol.	***v***	***t****	**Sol.**	***v***	***t*****	**Sol.**	***v***	***t***	**(%)**
SCA3-0	**636.1**	-	232	**636.06**	4	2.83	**636.06**	4	3.87	0.00
SCA3-1	**697.8**	-	170	**697.84**[1]	4	2.12	**697.84**	4	4.36	0.00
SCA3-2	**659.3**	-	160	**659.34**[1]	4	2.58	**659.34**	4	4.70	0.00
SCA3-3	680.6	-	182	**680.04**[1]	4	3.13	**680.04**	4	4.11	0.00
SCA3-4	**690.5**	-	160	**690.50**[1]	4	2.68	**690.50**	4	5.03	0.00
SCA3-5	**659.9**	-	178	**659.90**[1]	4	2.56	**659.90**	4	4.48	0.00
SCA3-6	**651.09**	-	171	**651,09**[1]	4	4.40	**651.09**	4	4.53	0.00
SCA3-7	666.1	-	162	**659.17**[1]	4	2.98	**659.17**	4	5.48	0.00
SCA3-8	**719.5**	-	157	**719.47**[1]	4	3.98	**719.47**	4	4.23	0.00
SCA3-9	**681.0**	-	167	**681.00**[1]	4	3.86	**681.00**	4	4.36	0.00
SCA8-0	975.1	-	98	**961.50**	9	3.21	**961.50**	9	6.41	0.00
SCA8-1	1052.4	-	95	1050.20	9	3.55	**1049.65**	9	5.67	-0.05
SCA8-2	**1039.6**	-	83	**1039.64**	9	4.67	**1039.64**	9	9.63	0.00

SCA8-3	991.1	-	94	**983.34**[1]	9	3.29	**983.34**	9	6.61	0.00
SCA8-4	**1065.5**	-	84	**1065.49**	9	2.68	**1065.49**	9	5.33	0.00
SCA8-5	**1027.1**	-	96	**1027.08**	9	4.50	**1027.08**	9	6.06	0.00
SCA8-6	972.5	-	93	**971.82**	9	2.67	**971.82**	9	6.11	0.00
SCA8-7	1061.0	-	92	1052.17	9	4.32	**1051.28**	9	17.19	-0.08
SCA8-8	**1071.2**	-	85	**1071.18**	9	3.43	**1071.18**	9	4.80	0.00
SCA8-9	**1060.5**	-	86	**1060.50**	9	4.12	**1060.50**	9	7.30	0.00
CON3-0	**616.5**	-	171	**616.52**	9	3.89	**616.52**	9	5.30	0.00
CON3-1	**554.5**	-	190	**554.47**[1]	9	2.97	**554.47**	9	4.05	0.00
CON3-2	521.4	-	176	519.26	9	3.32	**519.11**	9	5.45	-0.03
CON3-3	**591.2**	-	177	**591.19**[1]	9	2.78	**591.19**	9	4.44	0.00
CON3-4	**588.8**	-	173	589.32	9	3.12	**588.79**	9	4.98	0.00
CON3-5	**563.7**	-	179	**563.70**[1]	9	3.45	**563.70**	9	4.53	0.00
CON3-6	**499.1**	-	195	500.80	9	2.98	**499.05**	9	5.03	0,00
CON3-7	**576.5**	-	226	**576.48**	9	2.40	**576.48**	9	4.64	0.00
CON3-8	**523.1**	-	174	**523.05**[1]	9	5.02	**523.05**	9	4.92	0,00
CON3-9	**578.2**	-	163	580.05	9	3.14	**578.24**	9	3.92	0.01
CON8-0	**857.2**	-	86	**857.17**	9	3.40	**857.17**	9	7.03	0.00
CON8-1	**740.9**	-	81	**740.85**[1]	9	3.73	**740.85**	9	5.48	0,00
CON8-2	716.0	-	84	713.14	9	2.87	**712.89**	9	6.23	-0.04
CON8-3	**811.1**	-	91	**811.07**	10	3.82	**811.07**	10	2.42	0.00
CON8-4	**772.3**	-	87	**772.25**[1]	9	2,98	**772.25**	9	6.05	0.00
CON8-5	755.7	-	94	756.91	9	5.76	**754.88**	9	6.20	-0.11
CON8-6	693.1	-	96	**678.92**[1]	9	4.00	**678.92**	9	5.78	0.00
CON8-7	814.8	-	94	**811.96**	9	2.46	**811.96**	9	5.22	0.00
CON8-8	774.0	-	94	**767.53**	9	4.21	**767.53**	9	5.66	0.00
CON8-9	809.3	-	92	**809.00**[1]	9	3.87	**809.00**	9	5.50	0.00

(*) CPU time in seconds in a PC Pentium IV 1.5 GHz.
(**) CPU time in seconds in a PC Pentium IV 2.4 GHz.
([1]) Also found by Montané and Galvão (2006).

The times presented in Table 2 (and also Tables 4 and 6) give an idea of the computational effort demanded, but since they are referred to machines with distinct configurations, it is not possible to make a direct comparison among the respective algorithms. Also, some code optimizations are currently being performed and these are leading to a significant CPU time improvement (approximately 3 times faster) when compared to the version presented in this work.

Table 3 – Results obtained in Salhi and Nagy's Instances

Instance	**N° of clients**	v_i	v_f	**Best Sol.**	**Time (s)**	**Avg. Sol.**	***Gap* (%)**	γ	**Avg. Time (s)**
CMT1X	50	3	3	466.77	4.25	467.32	0.12	0.25	4.80
CMT1Y	50	3	3	466.77	3.86	467.44	0.14	0.30	5.20
CMT2X	75	6	6	684.21	30.31	687.67	0.51	0.10	26.21
CMT2Y	75	6	6	684.60	24.67	687.72	0.46	0.15	26.22
CMT3X	100	5	5	721.40	28.17	726.76	0.74	0.50	33.00
CMT3Y	100	5	5	721.27	34.25	726.26	0.69	0.50	33.53
CMT12X	100	6	5	662.22	34.69	671.04	1.33	0.35	34.80
CMT12Y	100	6	6	663.50	39.81	673.03	1.44	0.00	34.67
CMT11X	120	5	4	842.23	67.63	879.31	4.40	0.50	58.51
CMT11Y	120	5	4	846.23	62.23	880.77	4.08	0.35	57.75
CMT4X	150	7	7	852.83	183.14	864.64	1.38	0.35	209.97
CMT4Y	150	7	7	852.83	225.06	866.93	1.65	0.35	208.44
CMT5X	199	11	10	1031.17	272.30	1065.51	3.33	0.50	240.87
CMT5Y	199	11	10	1030.93	271.56	1059.49	2.77	0.10	240.59

Table 3 presents the results obtained on the test problems generated by Salhi and Nagy (1999) and Table 4 shows a comparison between the ILS-VND and the best known results found in the literature, namely those determined by Zachariadis *et al.* (2007) and Wassan *et al*. (2008).

Analyzing Table 4, it can be verified that the average gap between the best solutions and the average solutions was 1.65%, with the highest value in the instance CMT11X. Table 5 shows that among the 14 instances listed, the ILS-VND algorithm was capable of improving the result of one test problem and equaling one other, resulting in an average gap of 0.96% with respect to the best results found in the literature. It is important to emphasize that the gap in the instances CMT3X, CMT4X, CMT4Y, CMT5X and CMT5Y was up to 0.06%.

Table 4 – Comparison between ILS-VND and literature results in Salhi and Nagy's instances

Instance	Wassan *et al.*			Zachariadis *et al.*			ILS-VND			*Gap*
	Sol.	*v*	*t**	Sol.	*v*	*t***	Sol.	*v*	*t*	(%)
CMT1X	468.30	3	48	469.80	3	2.89	**466.77**	3	4.25	-0.05
CMT1Y	**458.96**	3	69	469.80	3	3.85	466.77	3	3.86	1.70
CMT2X	**668.77**	6	94	684.21	6	7.42	684.21	6	30.31	2.31
CMT2Y	**663.25**	6	102	684.21	6	8.02	684.60	6	24.67	3.22
CMT3X	729.63	5	294	**721.27**	5	11.62	721.40	5	28.17	0.06
CMT3Y	745.46	5	285	**721.27**	5	13.53	**721.27**	5	34.25	0.32
CMT12X	**644.70**	5	242	662.22	5	11.80	662.22	5	34.69	2.72
CMT12Y	**659.52**	6	254	662.22	5	7.59	663.50	6	39.81	0.60
CMT11X	861.97	4	504	**838.66**[1]	4	17.78	842.23	4	67.63	0.43
CMT11Y	**830.39**	4	325	837.08	4	14.26	846.23	4	62.23	1.96
CMT4X	876.50	7	558	**852.46**[2]	7	27.75	852.83	7	183.14	0.04
CMT4Y	870.44	7	405	**852.46**	7	31.20	852.83	7	225.06	0.04
CMT5X	1044.51	9	483	**1030.55**	10	51.67	1031.17	10	272.30	0.06
CMT5Y	1054.46	9	533	**1030.55**	10	58.81	1030.93	10	271.56	0.04

(*) CPU time in seconds in a PC Pentium IV 2.4 GHz.
(**) CPU time in seconds in a PC Pentium IV 2.4 GHz.
([1]) A better result was found by Ropke and Pisinger: 837.
([2]) A better result was found by Chen and Wu (2005): 852.35.

Table 5 – Results obtained in Montané and Galvão's instances

Instance	Nº of clients	v_i	v_f	Best Sol.	Time (s)	Avg. Sol.	*Gap* (%)	γ	Avg. Time (s)
r101	100	12	12	1012.96	42.03	1024.66	1.16	0.2	40.14
r201	100	3	3	666.20	26.00	666.54	0.05	0.3	31.60
c101	100	16	16	1220.99	15.52	1228.88	0.65	0.9	15.90
c201	100	5	5	662.07	21.45	662.32	0.04	0.1	23.56
rc101	100	10	10	1059.32	41.22	1063.80	0.42	0.4	48.21
rc201	100	3	3	672.92	21.80	672.95	0.00	0.8	27.54
r1_2_1	200	24	23	3393.02	236.00	3433.12	1.18	0.4	236.72
r2_2_1	200	6	5	1666.43	185.25	1682.35	0.96	0.1	167.18
c1_2_1	200	29	28	3644.30	221.84	3668.31	0.66	1.0	193.27
c2_2_1	200	9	9	1729.59	254.44	1737.85	0.48	0.5	227.57
rc1_2_1	200	24	24	3340.04	292.91	3381.15	1.23	0.2	315.93
rc2_2_1	200	6	5	1560.00	147.25	1565.03	0.32	0.2	158.12
r1_4_1	400	55	54	9761.34	1455.47	9869.86	1.11	0.7	1308.32
r2_4_1	400	11	10	3594.90	1549.99	3646.47	1.43	0.6	1464.61
c1_4_1	400	64	63	11179.23	1707.76	11271.89	0.83	0.6	1390.00
c2_4_1	400	16	15	3577.56	1456.14	3633.26	1.56	0.8	1592.26
rc1_4_1	400	53	52	9677.44	1782.87	9757.13	0.82	0.7	1602.63
rc2_4_1	400	12	11	3426.60	1468.34	3484.95	1.70	0.3	1238.98

Table 5 presents the results found in the instances proposed by Montané and Galvão (2006), while Table 6 illustrates a comparison between the results determined by ILS-VND and those of Montané and Galvão (2006) and Zachariadis *et al.* (2007). The perturbation mechanisms employed in these instances were Double Swap ($P^{(2)}$) and Double Bridge ($P^{(3)}$). The Ejection Chain perturbation was not efficient in these test problems, particularly in those involving a large number of vehicles, because the solutions were very much modified, and it also frequently produced unfeasible solutions.

From Table 6, it is observed that the average gap between the average solutions and the best solutions was 0.81%, with the highest gap in the instance rc2_4_1. When comparing the results found by the literature with the ones determined by the ILS-VND (Table 6), one can notice that the ILS-VND had improved the results of 12 test problems and equaled another 5, leading to an average gap of -0.34%.

Table 6 – Comparison between ILS-VND and literature results in Montané and Galvão's instances

Instance	**Montané and Galvão**			**Zachariadis *et al.***			**ILS-VND**			***Gap***
	Sol.	***v***	***t****	**Sol.**	***v***	***t*****	**Sol.**	***v***	***t***	**(%)**
r101	1042.62	12	13.20	1019.48	12	10.5	**1012.96**	12	42.03	-0.64
r201	671.03	3	12.02	**666.20**	3	8.7	**666.20**	3	26.00	0.00
c101	1259.79	17	12.07	**1220.99**	16	10.2	**1220.99**	16	15.52	0.00
c201	666.01	5	12.40	**662.07**	5	5.7	**662.07**	5	21.45	0.00
rc101	1094.15	11	12.30	**1059.32**	10	12.9	**1059.32**	10	41.22	0.00
rc201	674.46	3	12.07	**672.92**	3	10.5	**672.92**	3	21.80	0.00
r1_2_1	3447.20	23	55.56	3393.31	23	61.8	**3393.02**	23	236.00.	-0.01
r2_2_1	1690.67	5	50.95	1673.65	5	47.4	**1666.43**	5	185.25	-0.43
c1_2_1	3792.62	29	52.21	3652.76	28	66.3	**3644.30**	28	221.84	-0.23
c2_2_1	1767.58	9	65.79	1735.68	9	60.9	**1729.59**	9	254.44	-0.35
rc1_2_1	3427.19	24	58.39	3341.25	23	45.3	**3340.04**	24	292.91	-0.04
rc2_2_1	1645.94	5	52.93	1562.34	5	62.4	**1560.00**	5	147.25	-0.15
r1_4_1	10027.81	54	330.42	**9758.77**	54	315.3	9761.34	54	1455.47	0.03
r2_4_1	3695.26	10	324.44	3606.72	10	273.6	**3594.90**	10	1549.99	-0.33
c1_4_1	11676.27	65	287.12	11207.37	63	283.5	**11179.23**	63	1707.76	-0.25
c2_4_1	3732.00	15	330.20	3630.72	15	336.0	**3577.56**	15	1456.14	-1.46
rc1_4_1	9883.31	52	286.66	9697.65	52	145.8	**9677.44**	52	1782.87	-0.21
rc2_4_1	3603.53	11	328.16	3498.30	11	345.0	**3426.60**	11	1468.34	-2.05

(*) CPU Time in seconds in a PC Athlon XP 2.0.
(**) CPU Time in seconds in a PC Pentium IV 2.4 GHz.

5. Concluding Remarks

This paper dealt with the Vehicle Routing Problem with Simultaneous Pickup and Delivery. In order to solve it, an algorithm based on the Iterated Local Search metaheuristic, which uses a VND procedure in the local search phase, was proposed. It is an extension of the heuristic developed by Subramanian and Cabral (2008) for the VRPSPD with lime limits in which two new perturbation mechanisms were added, specifically, Ejection Chain and Double Swap.

The algorithm developed was tested in 72 instances reported in the literature and it was found capable of improving the result of 17 test problems and had equaled the solution of another 42. In the 40 instances generated by Dethloff (2001), the ILS-VND improved the results of 4 instances and equaled 36, with an average gap of -0.01% with respect to the best results indicated in the literature. In the 14 test problems formulated by Salhi and Nagy (1999), one result was improved, while another one equaled the best known solution. In addition, the gap in another 5 cases was up to 0.06%. In the instances proposed by Montané and Galvão (2006), the ILS-VND improved the solution of 12 test problems and equaled another 5, with an average gap of -0.34%. The main characteristic of these test problems is the fact of having some instances with more clients than the other ones proposed by Dethloff (2001) and Salhi and Nagy (1999). Hence, the results obtained are very promising since it shows the efficiency of the proposed algorithm in solving instances with higher dimensions.

Finally, for future work, one can suggest: (i) incorporating more efficient procedures to reduce the dependence of the factor γ for generating the initial solution, (ii) searching for alternatives to reduce the computational effort in some neighborhoods in such a way that the local search performance is not compromised, (iii) implementing other perturbation mechanisms, (iv) performing hybridizations, (v) combining exact and heuristic methods and (vi) developing parallel strategies for the proposed algorithm.

References

Angelelli, E. and Mansini, R. A branch-and-price algorithm for a simultaneous pick-up and delivery problem, Working Paper, *Article presented at the EURO/INFORMS Meeting,* 2003.

Berbeglia, G.; **Cordeau, J-F**.; **Gribkovskaia, I**. and **Laporte, G**. (2007) Static Pickup and Delivery Problems: A Classification Scheme and Survey. *Top*, 15, 1-31.

Bianchessi, N. and Righini, G. (2007), Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery, *Computers & Operations Research*, 34, 578-594.

Chen, J. F. and Wu, T. H. (2006), Vehicle routing problem with simultaneous deliveries and pickups, *Journal of the Operational Research Society*, 57, 579-587.

Chen, J. F. (2006) Approaches for the Vehicle Routing Problem with Simultaneous Deliveries and Pickups. *Journal of the Chinese Institute of Industrial Engineers*, 23, 2, 141-150.

Crispim, J. and Brandão, J. (2005), Metaheuristics applied to mixed and simultaneous extensions of vehicle routing problems with backhauls, *Journal of the Operational Research Society*, 56, 1296-1302.

Dell'Amico, M., Righini, G. and Salanim, M. (2006), A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection, *Transportation Science*, 40, 235-247.

Dethloff, J. (2001), Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up, *OR Spektrum*, 23, 79-96.

Gökçe, E. I. *A revised ant colony system approach to vehicle routing problems*, Master Thesis, Graduate School of Engineering and Natural Sciences, Sabanci University, Turkey, 2004.

Gribkovskaia, I., Halskau, Ø., Laporte, G. and Vlcek, M. (2007) General solutions to the single vehicle routing problem with pickups and deliveries, *European Journal of Operational Research*, 180, 2, 126-141.

Lourenço, H. R., Martin, O. C. and Stützle, T. Iterated Local Search. Fred Glover e Gary A. Kochenberger (Eds.), Handbook of Metaheuristics, Kluwer Academic Publishers, Norwell, MA, 321-353, 2002.

Min, H. (1989), The multiple vehicle routing problem with simultaneous delivery and pick-up points, *Transportation Research*, 23, 377-386.

Mladenović, N. and Hansen, P. (1997), Variable Neighborhood Search. *Computers & Operations Research*, 24, 1097-1100.

Martin, O.; Otto, S. W. and Felten, E. W. (1991), Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5, 299-326.

Montané, F.A.T. and Galvão, R. D. (2006), A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service, *Computers & Operations Research*, 33, 595-619.

Nagy, G. and Salhi, S. (2005), Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries, *European Journal of Operational Research*, 162, 126-141.

Or, I. *Traveling salesman-type combinational problems and their relation to the logistics of blood banking*. PhD Thesis, Northwestern University, USA, 1976.

Rego, C. and Roucairol, C. A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem. I. Osman and J. Kelly (Eds), Meta-heuristics Theory and Applications, Kluwer, Dordrecht, p. 253–295, 1996.

Ropke, S. and Pisinger, D. A unified heuristic for a large class of vehicle routing problems with backhauls. *Technical Report* n. 2004/14, University of Copenhagen, Denmark, 2004.

Salhi, S. and Nagy, G. (1999), A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling. *Journal of the Operational Research Society*, 50, 1034-1042.

Subramanian, A. and Cabral, L. A. F. GRASP/VND applied to the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Proc. of the Brazilian Symposium of Operational Research*, SBPO, Rio de Janeiro, 2007 (*in Portuguese*).
Subramanian, A. and Cabral, L. A. F. (2008), An ILS based heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery and Time Limit. *Lecture Notes in Computer Science*, 4972, 135-146.
Vural, A. V. *A GA based meta-heuristic for capacited vehicle routing problem with simultaneous pick-up and deliveries*, Master Thesis, Graduate School of Engineering and Natural Sciences, Sabanci University, Turkey, 2003.
Wassan, N. A.; Wassan, A. H. and Nagy, G. (2008), A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. Journal of Combinatorial Optimization, 15, 4, 368-386.
Zachariadis, E. E.; Tarantilis, C. D. and Kiranoudis, C. T. (2007), A hybrid metaheuristic algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. *Expert Systems with Applications*, in press.