A Simple and Efficient Perturbation Heuristic to solve theVehicle Routing Problem

Sylvain Girard

Jacques Renaud

and

Fayez F. Boctor

August 2005

Working Paper DT-2005-JR-1

Network Organization Technology Research Center (CENTOR),

Université Laval, Québec, Canada

A Simple and Efficient Perturbation Heuristic to solve the Vehicle Routing Problem

Sylvain Girard
Jacques Renaud[1]
Fayez F. Boctor

Network Organisation Technology Research Center (CENTOR)
Faculté des Sciences de l'administration, Université Laval, Québec, Canada G1K 7P4

Abstract

Many efficient meta-heuristics were developed to solve the *vehicle routing problem* (VRP). However, these heuristics often require large computation times and significant effort to implement. In this paper we propose a simple and efficient heuristic that only uses the Clarke and Wright's savings heuristic, the 2-opt and the 3-opt improvement heuristics. It will be shown that the proposed heuristic yields solutions almost as good as those of most efficient meta-heuristics published previously. The proposed heuristic, for 34 benchmark instances, yielded solutions whose values lie on average within 0.97% of the best known solutions.

Key words: Vehicle routing, Savings algorithm, Perturbation heuristics.

1- Introduction

The standard *Vehicle Routing Problem* (VRP) is the problem of determining routes for vehicles delivering goods located at a central depot to a number of scattered customers. As goods are stored in the depot, all vehicles start from and return to this depot. Each customer has a given demand and vehicles are of limited capacity. Distances between customer locations and between them and the depot are known and the constructed routes should minimize the total distance traveled by the vehicles. The length or travel time of any route may be required not to exceed a specified value.

The VRP received a lot of attention during the last four decades and many heuristics were proposed to solve several classes of this problem. Laporte and Semet (2002) propose a classification of the published heuristics into two main groups: *classical heuristics* and *meta-heuristics*. They divide classical heuristics in three sub-groups: *constructive heuristics, two-phase heuristics* and *improvement heuristics*. Two-phase heuristics are also divided into two categories called *cluster-first, route-second heuristics* and *route-first, cluster-second heuristics*. They review

[1] Corresponding author, email : jacques.renaud@fsa.ulaval.ca

classical heuristics and provide some indications about their relative performance. Meta-heuristics are reviewed in Gendreau, Laporte and Potvin (1997, 2002) and are divided into six types: *Simulated Annealing*, *Deterministic Annealing*, *Tabu Search*, *Genetic Algorithms*, *Ant Systems* and *Neural Networks*. Finally, Laporte and Osman (1995) provide an extensive list of publications dealing with routing problems.

An important family of constructive heuristics contains the savings-based heuristics. The first savings heuristic was developed by Clarke and Wright (1964). Several researchers used the same savings criterion or modified criteria to develop saving-based heuristics. Among such heuristics are those developed by Gaskell (1967), Yellow (1970), Desrochers and Verhoog (1989), Altinkemer and Gavish (1991), Wark and Holt (1994) and Altinel and Öncan (2005).

These heuristics are relatively simple, much faster and easier to implement than meta-heuristics but, unfortunately, they produce less-quality solutions. According to Girard (2005) and based on the 14 original benchmark instances of Christofides, Mingozzi and Toth (1979), the average deviation of the solutions obtained by Clarke and Wright's method (with respect to the best published solutions) is 7.68%. This average deviation is 6.05% for Yellow's heuristic, 6.60% for the heuristic by Desrochers and Verhoog (1989), 3.42% for Altinkemer and Gavish's heuristic and 0.64% for the heuristic by Wark and Holt. These results are quite similar to those reported in Laporte and Semet (2002).

On the other hand, best published meta-heuristics produce average deviation varying between 0.15% (Reimann, Doerner and Hartl, 2004) and 1.72% (Xu and Kelly, 1996). The best known solutions are due to Rochat and Taillard (1995) but all these meta-heuristics require significantly long computation times.

The purpose of this paper is to develop a new and efficient savings-based perturbation heuristic to solve the VRP. The proposed method uses only three basic and simple components: the savings algorithm of Clarke and Wright, the 2-opt and the 3-opt improvement heuristics proposed by Lin (1965). These three components are embedded within a two phase perturbation heuristic. The first phase constructs a solution using a randomized savings formula. The second phase is an improvement phase that uses perturbation techniques. These techniques were used

with success to solve many routing problems; see Codenotti *et al.* (1996), Renaud, Boctor and Laporte (2002) and Boctor, Laporte and Renaud (2003).

The remainder of this paper is organized as follows. In the next section we present the main published savings-based heuristics. In the third section we introduce the general framework of our perturbation heuristic and describe its components. Finally in the fourth section we present our computational results and in the fifth section we draw our conclusions.

2- Savings-based heuristics

In this section we present Clarke and Wright's savings algorithm (1964), its modifications proposed by Gaskell (1967), Yellow (1970) and Altinel and Öncan (2005), as well as the savings-based matching algorithms proposed by Desrochers and Verhoog (1989), by Altinkemer and Gavish (1991) and by Wark and Holt (1994).

Clarke and Wright heuristic

This heuristic is an iterative procedure which, at each iteration, merges two routes into a single longer one. It starts by creating n routes where each route visits only one of the n customers. Then it merges the pair of routes that leads to the largest possible savings. The savings of merging route (0, …, i, 0) and route (0, j, …, 0) into route (0,…, i, j,…., 0) depends only on the cost of traveling from i to j and the cost to reach them from the depot. Precisely, these savings, denoted s_{ij}, equal $c_{0i} + c_{0j} - c_{ij}$, where c_{ij} is the traveling cost between i and j. Thus the heuristic calculates the savings for all pairs of customers and sorts these pairs in the decreasing order of the corresponding savings. Then, starting from the top of the list, the algorithm determines if the customers of the considered pair are on separate routes that can be feasibly merged (respecting both capacity and route length restrictions), and if they are the first or last customer of their respective routes. If this is the case, then the two routes are merged.

The procedure described above is called the parallel version of the heuristic as several routes can be constructed in parallel. A sequential version is also suggested where only one route is expanded until no more routes can be merged to this route. Then another route is started.

Gaskell, Yellow and Altinel & Öncan's versions of Clarke and Wright's heuristic

As the order in which savings are considered determines the final solution, we may have different outcomes by changing slightly the order of the savings list. Both Gaskell (1967) and Yellow (1970) suggested a modified savings formula: $s_{ij} = c_{0i} + c_{0j} - \lambda c_{ij}$ where λ is a parameter that allows putting less or more emphasis on the distance between *i* and *j*. Tests made by Golden, Magnanti and Nguyen (1977) show that λ=0.4 or λ=1.0 yield good results. Altinel and Öncan (2005) suggested a different savings formula that takes into account customers' demand. The proposed formula is: $s_{ij} = c_{0i} + c_{j0} - \lambda c_{ij} + \mu\left|c_{0i} - c_{j0}\right| + \nu\frac{d_i + d_j}{\bar{d}}$, where d_i is the demand of customer *i* and $\bar{d}$ is the average demand of all customers.

Also, in order to reduce computational times, some enhancements of Clarke and Wright algorithm were suggested by Golden, Magnanti and Nguyen (1977), by Nelson *et al*. (1985) and by Paessens (1988). However, we may notice that the significant increase in computers speed observed since the publication of these research works made these enhancements less effective.

Desrochers and Verhoog (1989) and Altinkemer and Gavish (1991) algorithms

Within these iterative heuristics, the savings of merging route *p* and route *q* is calculated as $s_{pq} = L_p + L_q - L_{pq}$ where L_p is the optimal length of the Hamiltonian cycle joining the depot and customers of route *p* and L_{pq} is the optimal length of the Hamiltonian cycle joining the depot and the customers of routes *p* and *q* together.

At each iteration, a max-weight 1-matching problem over all routes that can be merged is solved (using s_{pq} values as matching weights) and the merges suggested by the optimal matching are implemented. The procedure stops as soon as no more savings can be made. Altinkemer and Gavish (1991) present several variants of the heuristic; one of them uses an approximation of L_p instead of computing it exactly.

Wark and Holt (1994) algorithm

Starting with a set of *n* clusters where each one contains only one customer, this procedure successively merges clusters at their end points by solving a matching problem. The matching weight is either the standard savings s_{ij} or a modified savings formula designed to favour merging

routes that use a small portion of the vehicle capacity or routes such as their length is far below the allowed limit. Also, once a final solution is obtained (all clusters are self matched), the algorithm randomly splits some of the routes and the procedure is repeated.

A comparison between the results reported by the authors of the previously described methods (based on the 14 benchmark instances of Christofides, Mingozzi and Toth, 1979) is given in Laporte and Semet (2002) and in Girard (2005). As pointed out by Laporte and Semet, the results reported by Altinkemer and Gavish (1991), showing an average deviation of 3.42% from the best known solutions, are the best of 40 different runs using several parameters and algorithm rules. Results of Wark and Holt (1994), showing an average deviation of 0.64%, are the best of 5 runs while those of Desrochers and Verhoog (1989), showing an average deviation of 6.60%, were obtained by only one run. Furthermore we notice that the distance or cost rounding rule is not the same for these three numerical tests. Altinkemer and Gavish rounded distances to the nearest integer while Wark and Holt used the real distance without any rounding.

Despite the above, it seems that Wark and Holt's heuristic is the best among the three matching-based heuristics. Also it is clear that matching-based heuristics are better than the simple Clarke and Wright heuristic and its modifications presented above.

3- The proposed heuristic

In this section we present first the general framework of the proposed heuristic, called *Perturb-CW*, followed by a detailed description of its different components.

General framework

The general framework of *Perturb-CW* is given in Figure 1. To obtain an initial solution rapidly we repeatedly apply a new modified version of Clarke and Wright's heuristic, called CW^+. After each application of CW^+, every constructed route is improved by applying the 3-opt improvement heuristic. Then the best obtained solution is improved by applying a heuristic we call *Subsets–*CW^+. This new improvement heuristic applies CW^+ a number of times to each of a number of different subsets of the customers set. Afterwards, we repeatedly apply the following procedure: Perturb the current solution and then reapply the *Subsets–*CW^+ heuristic again. The procedure stops if no better solution is found during a number T of successive iterations.

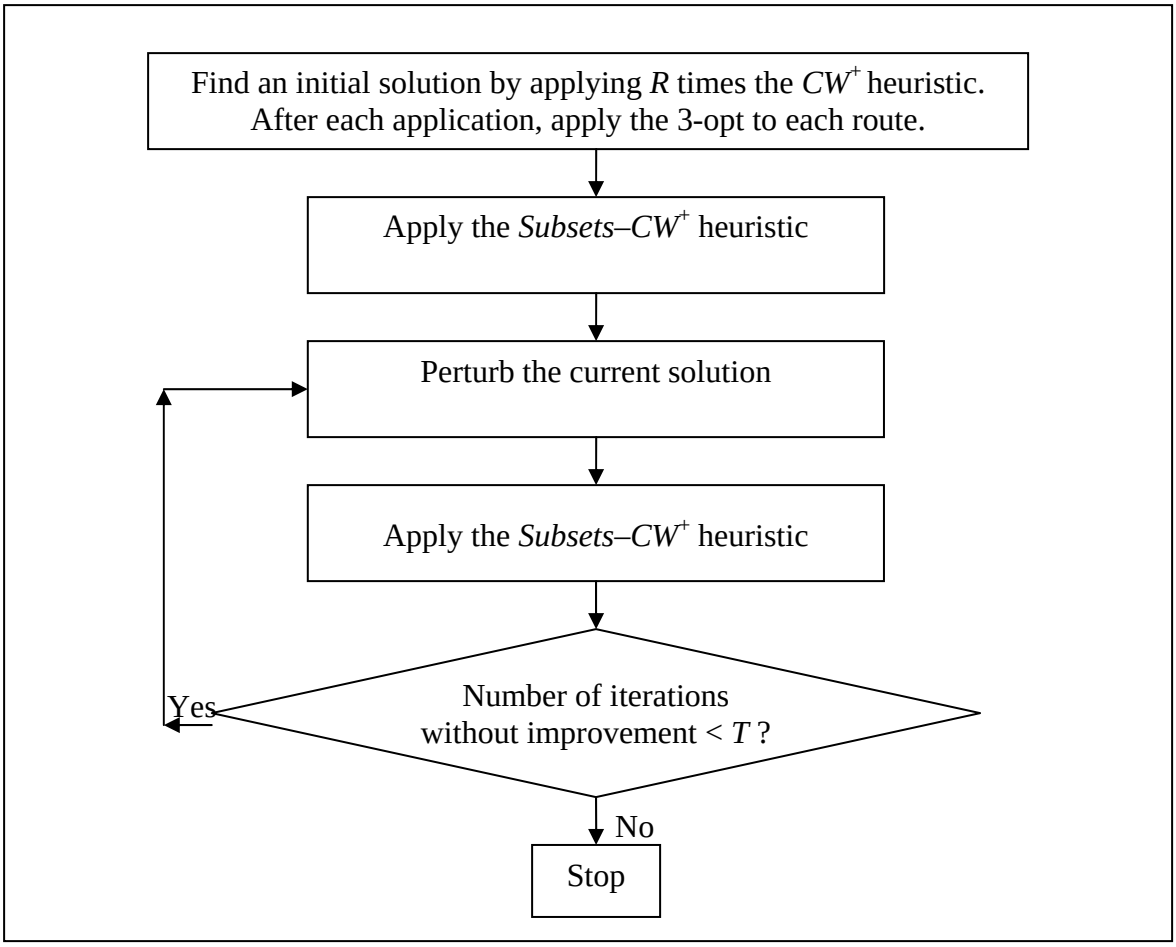

Figure 1: The general framework of *Perturb-CW*

The CW^+ heuristic

The CW^+ heuristic works as the original Clarke and Wright's algorithm but uses the following modified savings formula: $s_{ij}^{+} = c_{0i} + c_{j0} - \lambda_{ij} c_{ij}$, where the weight λ_{ij} is randomly chosen between two predetermined limits λ^- and λ^+ for every pair (i,j).

As CW^+ is very fast, we suggest to apply the 2-opt or the 3-opt heuristic to improve each route of the obtained solution. Using 3-opt leads to better solutions but to reduce computing times we may use the 2-opt heuristic. We may also repeat CW^+ a number of times, say R, and retain the best obtained solution.

The Subsets–CW^+ heuristic

The *Subsets–*CW^+ is an improvement procedure that can be applied to any feasible VRP solution. Its main steps are exhibited in Figure 2. The main idea of this heuristic is to repeatedly

apply the CW^+ heuristic to a number of different subsets of customers. The 2-opt heuristic is used after each execution of CW^+ to improve each of the constructed routes. Within the *Subsets*-CW^+, the 3-opt heuristic is only applied to improve the best among the solutions obtained by CW^+.

Starting from an initial solution having L routes, each route of the current solution is considered as a separate subset. These subsets are ordered according to their proximity to each other. In our implementation, subsets are ordered according to the polar angle of their center of gravity with respect to the depot location. We then solve all the sub-problems composed of m consecutives subsets by applying CW^+ several times followed by the 2-opt heuristic as mentioned above. Starting from $m = 2$ to $m = M$, the above is repeated as long as a non negative gain is obtained. Here, M is parameter indicating the maximum number of subsets to be considered simultaneously.

```
Best solution=initial solution
10: Put the customers of each of the L routes of the best solution in a separated subset
   Order these subsets (using a measure of their proximity to each other)
   For m=2 to M, Do
     For t=1 to L, Do
       Repeat R times (producing R solutions)
          Apply CW+ to the customers visited by route t and the m-1 subsequent routes
          Apply the 2-opt improvement heuristic to each obtained route
       End Repeat
       Apply the 3-opt heuristic to improve the routes of the best of the R obtained solutions
       If the resulting solution is better than the best solution, save it as the best solution and go to 10
     End Do
   End Do
```

Figure 2: Main steps of the *Subsets–*CW^+ heuristic.

Solution Perturbation procedure

To modify the current solution and obtain a new starting solution to be improved by the *Subsets–*CW^+ heuristic, we use the perturbation scheme presented in Figure 3. This scheme is quite simple. It removes a percentage P of the arcs of the current solution (where P is randomly drawn between two limits P^- and P^+) and replaces each removed arc by two arcs joining each of its two customers to the depot. Thus each time we remove an arc we replace its route by two shorter feasible routes. Afterwards we apply Clarke and Wright's heuristic to construct a solution from the resulting set of routes.

Draw a random number P from a uniform distribution between P^- and P^+
For each arc (i,j) of the L routes of the current solution, Do
  Draw a random number X from a uniform distribution between 0 and 100
  If $X<P$ then
    Remove the arc (i,j) from the current solution
    Divide the corresponding route into two separate routes by adding arcs $(0,i)$ and $(0,j)$
  End If
End Do
Apply Clarke and Wright's heuristic to construct a solution with the resulting set of routes

Figure 3: The solution perturbation procedure

4- Computational results

The proposed heuristic was coded in Delphi 7 on a Pentium-M PC clocked at 1.6 GHz under Windows XP. Our numerical experiment used two sets of Euclidean benchmark instances. The first set contains the 14 instances proposed by Christofides, Mingozzi and Toth (1979) (http://mscmga.ms.ic.ac.uk/jeb/orlib.). These *small* instances have between 50 and 199 customers. Their best solutions are given in Laporte and Semet (2002). The second set contains the 20 *larger* instances (having between 200 and 483 customers) presented by Golden *et al*. (1998) (http://www-bus.colorado.edu/publications/workingpapers/kelly). Detailed characteristics of all these instances as well as their best known solutions are reported in Reimann *et al*. (2004). In the following we present the numerical experiment undertaken to evaluate the performance of the proposed heuristic but first we evaluate the performance of each of its components independently. Notice that in all these tests the exact real distances are used.

Performance evaluation of CW^+ alone

Table 1 shows the average percentage deviations with respect to the best known solutions as obtained by repeatedly applying the CW^+ heuristic alone in function of the values of its parameters λ^- , λ^+ and in function of the number of repetition R. After each repetition of the CW^+, the 3-opt improvement heuristic is used to improve individual routes. The rows of the table are divided in four blocks and the average value of λ_{ij} for these four blocks is 0.4, 1.0, 1.1 and 1.4 respectively. For each combination of λ^- , λ^+ and R the table gives the average percentage deviation above the best know solutions and the average computing time (in seconds) for the 14 *small* instances, for the 20 *large* instances and for *all* 34 instances. As expected, the deviation decreases as R increases and the best results are obtained with R=100. The table shows that, for

the whole set of instances, the best combinations of λ^- and λ^+ are: [1.2 , 1.6] and [1.1 , 1.7] with an average (over all instances) percentage deviations of 4.84% and 4.89% respectively. For the 14 small instances, the best results are obtained with λ^-=0.9 and λ^+=1.3 with an average percentage deviation of 3.18% in less than 4 seconds in average. Finally, if solutions are required in less than 10 seconds, the combination λ^-=1.2, λ^+=1.6 and R=20 is a good compromise leading to an overall average deviation of 5.49% in an average of 8.4 seconds. This combination of parameters will be used in our later tests.

Table 2 compares the average results obtained by Clarke and Wright's heuristic, by Yellow's heuristic and by CW^+ with λ^-=1.2 and λ^+=1.6 for both R=30 and R=100. The table shows that CW^+ outperforms the compared heuristics. Notice that we reported the results for R=30 as CW^+ with this value of R required computing times very close to those required by Yellow's heuristic. Also notice that all tested heuristics were followed by the 3-opt improvement heuristic and the implemented version of Yellow's heuristic uses 12 values of λ (λ=0.4, 0.5, …, 1.5).

Performance evaluation of Subsets-CW^+ alone

Table 3 presents the results obtained by the *Subsets-CW^+* heuristic alone where initial solutions were obtained with CW^+ with λ^-=1.2, λ^+=1.6 and R=20. This table shows the results for 36 different parameter combinations. A first interesting observation is that the *Subsets–CW^+* produced better results than CW^+ in shorter computing times. The smallest average (over all instances) percentage deviation obtained by CW^+ is 4.84% and was obtained in 42 seconds (see Table 1). On the other hand, *Subsets-CW^+* with R=5, M=5, λ^-=1.2 and λ^+=1.6 gives an average percentage deviation of 2.96% in only 20.8 seconds.

As shown in Table 3, *Subsets-CW^+* with λ^-=1.1 and λ^+=1.7 seems to give the best results. The smallest average percentage deviation (over all instances) is 1.91% obtained in 224 seconds with $M=L/2$, R=30, λ^-=1.1 and λ^+=1.7. Using these parameters, the average deviation for the 20 large instances is 2.12% in 368 seconds. For the 14 small instances, the smallest average deviation is 1.50% in 9 seconds. These are better than those of many well known heuristics. Notice also that with M=5, R=40, λ^-=1.1 and λ^+=1.7, the *Subsets-CW^+* heuristic produced an average percentage deviation (over all instances) of 1.92% in 88 seconds. In conclusion, it seems that *Subsets-CW^+* is a fast and efficient improvement procedure for the VRP.

Table 1: Results obtained by CW^+ alone

Problem set	λ^-	λ^+	*R*																			
			10		20		30		40		50		60		70		80		90		100	
			% dev	Sec.	% dev	Sec.	% dev	Sec.	% dev	Sec.	% dev	Sec.	% dev	Sec.	% dev	Sec.	% dev	Sec.	% dev	Sec.	% dev	Sec.
Small	0.1	0.7	19.32	0.4	17.92	0.8	17.35	1.2	17.01	1.6	16.02	2.0	15.94	2.4	15.94	2.8	15.94	3.3	15.94	3.7	15.94	4.1
Large			23.24	9.2	22.49	18.3	22.37	27.4	22.37	36.5	21.93	45.6	21.85	54.7	21.66	63.8	21.03	73.0	21.01	82.1	20.96	91.2
All			21.63	5.5	20.61	11.1	20.30	16.6	20.16	22.1	19.50	27.7	19.41	33.2	19.30	38.7	18.94	44.3	18.92	49.8	18.73	55.3
Small	0.2	0.6	11.24	0.4	10.30	0.8	10.30	1.2	10.06	1.6	9.89	1.9	9.74	2.3	9.70	2.7	9.64	3.1	9.41	3.5	9.02	3.9
Large			14.43	10.3	14.03	20.7	13.66	31.0	13.58	41.3	13.41	51.5	13.41	61.8	13.36	72.1	13.26	82.3	13.12	92.6	13.08	102.9
All			13.12	6.2	12.50	12.5	12.28	18.7	12.14	24.9	11.96	31.1	11.90	37.3	11.85	43.5	11.77	49.7	11.59	55.9	11.41	62.1
Small	0.3	0.5	9.71	0.4	9.30	0.8	8.67	1.1	8.48	1.5	8.45	1.9	8.17	2.3	8.08	2.7	7.73	3.0	7.73	3.4	7.72	3.0
Large			11.77	11.9	11.08	23.8	10.78	35.8	10.46	47.7	10.26	59.5	10.26	71.4	10.15	83.3	10.10	95.2	10.03	107.1	10.00	119.0
All			10.92	7.2	10.34	14.3	9.91	21.5	9.65	28.7	9.51	35.8	9.40	42.9	9.30	50.1	9.12	57.3	9.08	64.4	9.06	71.6
Small	0.7	1.3	4.87	0.4	4.43	0.8	4.07	1.2	4.01	1.6	3.77	2.0	3.67	2.4	3.67	2.8	3.41	3.2	3.41	3.6	3.23	4.0
Large			8.26	7.6	7.81	15.2	7.45	22.7	7.20	30.4	7.16	38.0	7.10	45.6	7.09	53.3	7.09	60.9	7.04	68.6	6.96	76.2
All			6.86	4.6	6.41	9.3	6.06	13.9	5.88	18.5	5.76	23.2	5.69	27.8	5.68	32.5	5.57	37.1	5.54	41.8	5.42	46.5
Small	0.8	1.2	4.83	0.4	4.35	0.8	4.07	1.2	3.94	1.6	3.93	2.0	3.93	2.4	3.90	2.8	3.77	3.2	3.74	3.5	3.71	3.9
Large			7.42	7.7	7.10	15.4	7.04	23.1	6.77	30.7	6.66	38.4	6.66	46.0	6.56	53.8	6.47	61.5	6.47	69.2	6.34	77.0
All			6.36	4.7	5.97	9.4	5.82	14.1	5.60	18.7	5.53	23.4	5.53	28.1	5.46	32.8	5.36	37.5	5.35	42.1	5.26	46.9
Small	0.9	1.1	4.88	0.4	4.60	0.8	4.50	1.2	4.40	1.6	4.37	2.0	4.18	2.4	4.11	2.8	3.96	3.2	3.95	3.6	3.95	3.9
Large			7.27	8.1	6.92	16.1	6.68	24.1	6.56	32.2	6.55	40.1	6.52	48.1	6.35	56.2	6.30	64.1	6.27	72.2	6.24	80.1
All			6.28	4.9	5.97	9.8	5.78	14.7	5.67	19.6	5.66	24.4	5.55	29.3	5.43	34.2	5.33	39.0	5.32	43.9	5.30	48.8
Small	0.8	1.4	4.32	0.4	3.88	0.8	3.86	1.2	3.71	1.6	3.66	2.0	3.56	2.4	3.56	2.8	3.55	3.2	3.49	3.5	3.48	3.9
Large			7.75	7.5	7.42	15.0	7.10	22.5	6.87	30.0	6.79	37.5	6.78	45.0	6.71	52.6	6.70	60.1	6.63	67.8	6.63	75.4
All			6.34	4.6	5.96	9.1	5.76	13.7	5.57	18.3	5.50	22.9	5.45	27.5	5.41	32.1	5.40	36.7	5.33	41.3	5.33	46.0
Small	0.9	1.3	4.23	0.4	3.93	0.8	3.84	1.2	3.79	1.6	3.47	2.0	3.42	2.3	3.39	2.7	3.27	3.1	3.25	3.5	3.18	3.9
Large			6.90	7.5	6.79	15.0	6.66	22.7	6.52	30.1	6.40	37.7	6.31	45.3	6.27	52.8	6.26	60.4	6.25	68.0	6.21	75.5
All			5.80	4.6	5.62	9.2	5.50	13.8	5.40	18.4	5.19	23.0	5.12	27.6	5.08	32.2	5.03	36.8	5.01	41.4	4.96	46.0
Small	1.0	1.2	4.82	0.4	4.36	0.8	4.25	1.2	4.21	1.6	4.05	1.9	3.98	2.3	3.96	2.7	3.96	3.1	3.96	3.5	3.96	3.8
Large			7.02	7.8	6.74	15.7	6.61	23.5	6.43	31.3	6.30	39.1	6.25	46.9	6.24	54.6	6.09	62.5	6.05	70.3	5.97	78.1
All			6.11	4.7	5.76	9.5	5.64	14.3	5.52	19.0	5.38	23.8	5.31	28.5	5.30	33.2	5.21	38.0	5.19	42.8	5.14	47.5
Small	1.1	1.7	4.85	0.4	4.25	0.8	3.84	1.2	3.75	1.6	3.62	1.9	3.61	2.3	3.39	2.7	3.31	3.1	3.31	3.5	3.29	3.9
Large			6.91	7.2	6.57	14.1	6.32	21.2	6.25	28.3	6.18	35.3	6.13	42.3	6.08	49.4	6.07	56.5	6.05	63.4	6.02	70.5
All			6.06	4.4	6.51	8.6	5.30	13.0	5.22	17.3	5.12	21.6	5.09	25.9	4.97	30.2	4.93	34.5	4.92	38.7	4.89	43.1
Small	1.2	1.6	4.82	0.4	4.46	0.8	4.06	1.1	3.98	1.5	3.79	1.9	3.79	2.3	3.76	2.7	3.76	3.1	3.74	3.4	3.74	3.8
Large			6.57	6.9	6.21	13.8	6.10	20.6	6.02	27.4	5.93	34.3	5.91	41.2	5.83	48.1	5.81	55.0	5.62	61.9	5.60	68.7
All			5.85	4.2	5.49	8.4	5.26	12.6	5.18	16.8	5.05	21.0	5.04	25.2	4.98	29.4	4.97	33.6	4.85	37.8	**4.84**	**42.0**
Small	1.3	1.5	4.90	0.4	4.61	0.7	4.44	1.1	4.44	1.5	4.39	1.9	4.38	2.2	4.37	2.6	4.32	3.0	4.32	3.3	4.28	3.7
Large			6.64	6.9	6.32	13.8	6.26	20.6	6.15	27.5	6.13	34.3	6.09	41.1	6.07	48.1	5.99	54.9	5.92	61.8	5.89	68.7
All			5.92	4.2	5.61	8.4	5.51	12.6	5.45	16.8	5.42	20.9	5.38	25.1	5.37	29.3	5.30	33.5	5.26	37.7	5.23	42.0

Table 2: Comparative results

Problem set	Clarke and Wright + 3-opt		Yellow + 3-opt λ= 0.4, 0.5, …, 1.5		CW^+ λ^-=1.2, λ^+=1.6, R=30		CW^+ λ^-=1.2, λ^+=1.6, R=100	
	% dev.	Sec.	% dev.	Sec.	% dev.	Sec.	% dev.	Sec.
Small instances	6.42	0.1	4.73	0.6	4.06	1.1	3.74	3.8
Large instances	8.59	1.2	6.57	18.7	6.10	20.6	5.60	60.7
All instances	7.70	0.7	5.81	11.2	5.26	12.6	4.84	42.0

Table 3: Average percentage deviations of solutions obtained by *Subset–CW*$^+$ alone

Problem set	M	R	*Subset–CW*$^+$ λ^-=0.9, λ^+=1.3		Subset–CW$^+$ λ^-=1.2, λ^+=1.6		Subset–CW$^+$ λ^-=1.1, λ^+=1.7	
			% dev.	Sec.	% dev.	Sec.	% dev.	Sec.
Small	5	2	2.68	1.4	3.24	1.3	2.87	1.3
Large			4.67	23.0	4.39	23.6	4.53	2.4
All			3.85	14.1	3.92	14.4	3.85	14.9
Small	L/2		2.75	2.0	2.57	2.1	2.81	2.2
Large			4.03	45.7	3.89	49.7	3.97	45.8
All			3.50	27.6	3.35	30.1	3.49	27.9
Small	5	5	2.18	2.2	2.23	2.3	2.27	2.5
Large			3.96	31.8	3.47	33.7	3.84	31.3
All			3.23	19.6	2.96	20.8	3.20	19.4
Small	L/2		1.89	3.7	2.31	3.7	1.89	3.3
Large			3.90	81.6	3.01	84.1	3.43	75.0
All			3.07	49.5	2.72	51.0	2.80	45.5
Small	5	10	2.23	3.9	2.37	3.8	2.05	3.8
Large			3.57	49.2	3.25	47.4	2.97	48.3
All			3.02	30.5	2.89	29.5	2.59	30.0
Small	L/2		2.10	7.1	2.37	7.2	1.70	6.5
Large			2.93	139.5	2.99	153.5	3.07	126.4
All			2.59	84.9	2.74	93.1	2.50	76.9
Small	5	20	1.83	5.8	2.39	16.9	1.67	6.6
Large			3.14	80.8	3.08	91.9	2.54	80.3
All			2.60	49.9	2.80	56.9	2.18	49.9
Small	L/2		1.95	12.8	2.07	11.6	1.77	11.8
Large			2.61	251.5	2.60	263.7	2.49	244.6
All			2.34	152.7	2.38	159.5	2.20	149.1
Small	5	30	1.53	10.9	1.84	9.3	1.50	9.0
Large			2.90	102.0	2.70	119.2	2.60	110.4
All			2.34	64.6	2.34	73.8	2.15	68.7
Small	L/2		1.69	16.4	2.22	16.8	1.62	18.1
Large			2.78	353.7	2.49	406.7	2.12	368.2
All			2.34	214.6	2.38	245.9	**1.91**	**224.3**
Small	5	40	1.55	14.2	1.68	12.7	1.50	10.5
Large			2.82	130.3	3.02	134.5	2.21	143.2
All			2.29	82.3	2.46	83.9	**1.92**	**88.2**
Small	L/2		1.99	31.4	1.84	2.7	1.76	18.8
Large			2.51	470.1	2.18	552.5	2.37	445.8
All			2.29	289.3	2.04	335.8	2.12	269.6

Performance evaluation of Perturb-CW

To implement *Perturb-CW* we need to choose the values to give to its parameters. These parameters are:

λ^- and λ^+	lower and upper limits on the random weight λ_{ij} used by CW^+,
R	number of repetitions of CW^+,
M	maximum number of subsets to be used by the *Subsets-*CW^+ heuristic,
P^- and P^+	lower and upper limits on the percentage of arcs to remove from the current solution,
T	maximum allowable number of perturbation iterations without improvement

In our implementation the initial solution is constructed by CW^+ with λ^- =1.2, λ^+ =1.6 and R=20. In addition we tested the following values of *Perturb-CW* parameters:

λ^- and λ^+	1.1 and 1.7,
R	5 or 20,
M	5 or L/2,
P^- and P^+	(0.6 and 0.8) or (0.5 and 0.7) or (0.3 and 0.5),
T	2, 5 or 10.

Table 4 gives the obtained results. First we notice that *Perturb-CW* gave significantly better results than *Subset–*CW^+. For the 14 small instances, *Perturb-CW* with most of the tested parameters produced an average percentage deviation of less than 1%. Using λ^- =1.1, λ^+ =1. 7, R=20, P^-=0.6; P^+=0.8, M=L/2 and T=10, we were able to obtain an average percentage deviation of 0.58% in 206 seconds of average computing time. As shown in Table 5, this result is better than that obtained by the following heuristics: Improved Petal of Renaud, Boctor and Laporte (1996), Flower algorithm of Rego (1998), Tabu search algorithm of Osman (1993), Taburoute of Gendreau, Hertz and Laporte (1994), Repeated matching algorithm of Wark and Holt (1994), Granular Tabu search of Toth and Vigo (2003), and Ejection-Chains Tabu search of Rego and Roucairol (1996).

For the 20 large instances, *Perturb-CW*, using the same parameters, was able to produce an average percentage deviation of 1.24% in 4463 seconds of average computing time. As shown in Table 6, this result is better than that obtained by the following heuristics: Record-to-record of Golden *et al*. (1998), Tabu search of Xu and Kelly (1996) and Granular tabu search of Toth and Vigo (2003).

Based on the obtained results we recommend to use two sets of parameters. The first produce good results faster than the second one who produced the best results. Detailed results of these two versions of *Perturb-CW* are given in Table 7.

Table 4: Results obtained by *Perturb-CW*

λ^-=1.1, λ^+=1. 7 and R=5								
P^-; P^+	M	Problem set	T = 2		T = 5		T = 10	
			% dev.	Sec.	%dev.	Sec.	% dev.	Sec.
0.6 ; 0.8	5	Small	1.32	9.5	1.15	21.2	0.94	38.2
		Large	2.72	122.1	2.64	297.5	2.37	441.7
		All	2.14	75.7	2.03	183.4	1.79	275.6
	L/2	Small	1.35	15.4	1.03	48.1	1.05	54.9
		Large	2.41	361.1	2.10	790.8	1.92	1455.6
		All	1.97	218.9	1.67	485.0	1.57	878.9
0.5 ; 0.7	5	Small	1.52	9.0	0.98	23.2	0.99	39.2
		Large	3.03	154.2	2.56	252.1	2.34	447.7
		All	2.40	94.4	1.92	157.8	1.78	279.5
	L/2	Small	1.20	16.2	1.10	34.1	0.89	63.7
		Large	2.53	343.4	2.14	991.4	2.07	1562.2
		All	1.99	208.7	1.72	597.2	1.58	945.2
0.3 ; 0.6	5	Small	1.49	9.6	1.10	19.0	0.81	37.8
		Large	3.06	128.9	2.68	270.4	2.48	521.8
		All	2.41	79.7	2.03	166.9	1.79	322.5
	L/2	Small	1.17	16.4	1.04	44.2	0.90	84.4
		Large	2.30	401.4	2.17	888.4	1.99	2046.4
		All	1.84	242.9	1.71	540.8	1.54	1238.7
λ^-=1.1, λ^+=1. 7 and R=20								
P^-; P^+	M	Problem set	T = 2		T = 5		T = 10	
			% dev.	Sec.	% dev.	Sec.	% dev.	Sec.
0.6 ; 0.8	5	Small	1.03	30.7	0.84	70.4	0.83	106.9
		Large	2.28	319.2	1.95	795.2	1.75	1548.4
		All	1.77	200.4	1.49	497.6	1.37	954.8
	L/2	Small	0.95	68.2	0.72	132.0	**0.58**	206.0
		Large	1.92	965.2	1.45	2915.1	**1.24**	4463.2
		All	1.50	595.4	1.15	1771.3	**0.97**	2710.2
0.5 ; 0.7	5	Small	0.94	31.3	0.77	81.1	0.68	118.6
		Large	1.93	379.0	1.84	949.3	1.75	1506.4
		All	1.53	235.8	1.40	592.5	1.31	935.0
	L/2	Small	0.84	64.5	0.70	142.0	0.73	276.9
		Large	1.83	1134.0	1.54	2419.3	1.29	5793.8
		All	1.43	693.9	1.19	1475.2	1.06	3522.1
0.3 ; 0.6	5	Small	1.05	36.1	0.80	90.4	0.64	133.7
		Large	1.96	434.3	1.83	911.3	1.71	1932.9
		All	1.59	269.8	1.40	573.8	1.27	1192.0
	L/2	Small	0.98	46.6	0.69	158.1	0.68	289.6
		Large	1.91	1130.5	1.55	2831.9	1.27	5968.9
		All	1.53	682.7	1.20	1735.4	1.03	3630.4

Table 5: Comparison between best published heuristics based on the 14 small instances

Heuristic	Reference	Average deviation from best known solutions (in %)	Average computation time (sec.)	Computer characteristics
Enhanced Clarke and Wright	Altinel & Öncan (2005)	3.83[a]	202	Sun Ultrasparc III
Improved petal	Renaud, Boctor & Laporte (1996)	2.43	209	Sun Sparc 2
Tabu search	Xu & Kelly (1996)	1.72 [b]	7175[b]	DEC Alpha
Flower (best)	Rego (1998)	1.54	133	HP 9000/712
Tabu search	Osman (1993)	1.01	2016	VAX 8600
Taburoute (standard)	Gendreau, Hertz & Laporte (1994)	0.86	2809	Silicon Graphics
Granular tabu search	Toth & Vigo (2003)	0.64	230	Pentium 200 MHz PC
Repeated matching	Wark & Holt (1994)	0.64	2674[c]	Sun 4/360 MP
Tabu search – ejection chains	Rego & Roucairol (1996)	0.71	1501[d]	Sun Sparc 4
Perturb-CW		**0.58**	**206**	Pentium M 1,6GHz
Genetic algorithm	Prins (2004)	0.23	311	Pentium 1 GHz
D-Ants (best of 10 runs)	Reimann, Doerner & Hartl (2004)	0.15	2280[e]	Pentium II 900MHz

[a] Results for the 7 capacitated instances only (instances with no restriction on route length). The reported solution is the best over 8 820 different solutions obtained with different parameters values. The computing time reported is cumulative over the 8 820 resolutions. On the same 7 instances, *Perturb-CW* gives an average deviation of 0.49%.

[b] Solutions and computing times taken from Xu and Kelly (1996) and Golden *et al*. (1998), p. 42. The average deviation is computed over 13 instances as no solution is reported for instance 9.

[c] Reported solutions are the best over five runs. Final solution truncated after the first decimal. Computational time is the average of five runs.

[d] Computing times are for the parallel implementation using four Sun Sparc computers.

[e] Reported average computing time was 3.8 minutes for each run and the reported solution is the best over 10 runs.

Table 6: Comparison between best published heuristics based on the 20 large instances

Heuristic	Reference	Average deviation from best known solutions (in %)	Average computation time (sec.)	Computer characteristics
Record-to-record	Golden *et al*. (1998)	3.43	2229	Pentium 100 MHz
Tabu search	Xu & Kelly (1996)	2.71[a]	210700[a]	DEC Alpha
Granular tabu search	Toth & Vigo (2003)	2.39	1053	Pentium 200MHz
Perturb-CW		**1.24**	4463	Pentium M 1,6GHz
Genetic algorithm	Prins (2004)	0.85	4014	Pentium 1 GHz
D-Ants (average of 10 runs)	Reimann, Doerner & Hartl (2004)	0.42[b]	9658	Pentium II, 900MHz
D-Ants (best of 10 runs)	Reimann, Doerner & Hartl (2004)	0.12	96580	Pentium II, 900MHz

[a] Based on solution to instances 9 to 20 as reported in Prins (2004). Prins reported that if real numbers are used Xu & Kelly results for problems 1-8 are infeasible. Prins also recomputed the solutions of instances 10 to 20 using real distances (see Prins, 2004, p. 1998-1999). Computing times as reported by Golden et al. (1998, p. 52). On the same 11 instances, *Perturb-CW* gives 1.30%.

[b] Detailed results are not given, only the average deviation over the best known solution.

Table 7: Results obtained by two proposed versions of *Perturb-CW*

Set	Problem number	Best Known solution	λ^-=1.1, λ^+=1. 7, R=20, M=L/2, P^-=0.6; P^+=0.8, T=5			λ^-=1.1, λ^+=1. 7, R=20, M=L/2, P^-=0.6; P^+=0.8, T=10		
			Solution	Percentage deviation	Time (sec.)	Solution	Percentage deviation	Time (sec.)
Small instances	1	524.61	525.00	0.00	5.5	524.61	0.00	7.0
	2	835.26	839.00	0.40	24.5	838.60	0.40	47.4
	3	826.14	832.00	0.74	46.7	834.31	0.99	45.3
	4	1028.42	1 040.00	1.28	211.0	1037.45	0.88	183.3
	5	1291.45	1 310.00	1.30	639.0	1301.76	0.80	743.2
	6	555.43	558.00	0.44	3.0	555.43	0.00	15.1
	7	909.68	927.00	1.90	23.5	924.56	1.64	74.3
	8	865.94	869.00	0.36	48.2	869.04	0.36	82.7
	9	1162.55	1 170.00	1.04	222.0	1173.93	0.98	342.2
	10	1395.85	1 420.00	1.62	461.0	1413.83	1.29	865.6
	11	1042.11	1 050.00	0.39	34.6	1046.19	0.39	62.6
	12	819.56	820.00	0.00	17.9	819.56	0.00	35.1
	13	1541.14	1 550.00	0.63	73.3	1547.48	0.41	324.6
	14	866.37	867.00	0.07	44.0	867.04	0.08	56.7
	Average			**0.72**	**132.0**		**0.58**	**206.0**
Large instances	1	5644.02	5784.79	2.49	425.8	5757.14	2.00	427,9
	2	8447.92	8643.17	2.31	1088.3	8621.75	2.06	2815,2
	3	11036.22	11345.76	2.80	1847.5	11261.76	2.04	1541,9
	4	13624.52	13734.81	0.81	4423.6	13674.97	0.37	2643,4
	5	6460.98	6460.98	0.00	161.7	6460.98	0.00	198,3
	6	8412.80	8424.68	0.14	550.7	8435.45	0.27	624,5
	7	10195.59	10281.72	0.84	734.9	10259.04	0.62	1364,4
	8	11828.78	11995.83	1.41	1712.6	12062.78	1.98	3575,5
	9	586.87	601.80	2.54	553.1	596.43	1.63	2541,6
	10	746.56	762.19	2.09	1231.8	759.34	1.71	2918,3
	11	927.27	944.04	1.81	5252.4	947.30	2.16	3609,1
	12	1133.79	1145.53	1.04	4531.9	1141.18	0.65	9652,7
	13	865.07	880.16	1.74	1415.2	878.48	1.55	1970,3
	14	1086.24	1110.18	2.20	2121.2	1110.62	2.24	5318,1
	15	1358.21	1375.88	1.30	7849.3	1376.01	1.31	8483,5
	16	1635.16	1665.95	1.88	7455.2	1659.71	1.50	10621,2
	17	708.76	712.38	0.51	1061.7	712.39	0.51	1125,1
	18	998.83	1013.65	1.48	2158.6	1011.90	1.31	2324,9
	19	1367.20	1373.80	0.48	4921.7	1375.98	0.64	13035,8
	20	1822.94	1841.67	1.03	8784.0	1830.74	0.43	14472,9
	Average			**1.45**	**2915.1**		**1.24**	**4463.2**
All	Average			**1.15**	**1771.3**		**0.97**	**2710.2**

5- Conclusions

In this paper we described a new and efficient perturbation heuristic to solve the vehicle routing problem. The main iteration of the heuristic can be described as follows. Starting from an initial solution, we repeatedly apply a modified version of Clarke and Wright heuristic to different subsets of customers in order to improve the solution. Afterwards, the obtained solution is perturbed and the main step is applied again. This procedure stops if no improvement is achieved during a predetermined number of consecutive iterations. The proposed heuristic, named *Perturb-CW* uses only three simple algorithms: Clarke and Wright's savings heuristic, 2-opt and 3-opt improvement heuristics. It outperforms many much harder to implement meta-heuristics. Consequently, it can be seen as an interesting and simple alternative for solving vehicle routing problems.

Acknowledgement

This research work was partially supported by the *Canadian Natural Sciences and Engineering Research Council* (*NSERC*) under grants OGP0172633 and OPG0036509. This support is gratefully acknowledged.

References

Altinel I.K. and T. Öncan, A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 56, 954-961, 2005.

Altinkemer K. and B. Gavish, Parallel savings based heuristic for the delivery problem, *Operations Research*, 39, 456-469, 1991.

Boctor, F.F., G. Laporte and J. Renaud, Heuristics for the traveling purchaser problem, *Computers and Operations Research*, 30, 491-504, 2003.

Christofides, N., A. Mingozzi and P. Toth, The vehicle routing problem, in N. Christofides, A. Mingozzi, P. Toth and C. Sandi (editors) *Combinatorial optimization*, Wiley, 315-338, 1979.

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

Codenotti, B., G. Manzini, L. Margara and G. Resta, Perturbation: an efficient technique for the solution of very large instances of Euclidean TSP, *INFORMS Journal on Computing*, 8, 125-133, 1996.

Desrochers, M and T. W. Verhoog, A Matching based savings algorithm for the vehicle routing problem, Working paper G-89-04, GERAD, Université de Montréal, 1989.

Gaskell, T.J., Bases for vehicle fleet scheduling, *Operational Research Quarterly*, 18, 281-295, 1967.

Gendreau, M., Hertz A. and Laporte G., A tabu search heuristic for the vehicle routing problem, *Management Science*, 40, 1276-1290, 1994.

Gendreau, M., G. Laporte and J. Y. Potvin, Vehicle routing: modern heuristics, in E.H.L. A. arts and J. K. Lenstra (editors) *Local Search in Combinatorial Optimization*, Wiley, Chichester, UK, 311-336, 1997.

Gendreau, M., G. Laporte and J.Y. Potvin, Metaheuristics for the capacitated VRP, in P. Toth and D. Vigo (editors) *The vehicle routing problem*, Siam monographs on discrete mathematics and applications, 129-154, 2001.

Girard, S., Résolution du problème de tournées de véhicules par perturbation, M. Sc. Thesis, Faculté des sciences de l'administration, Université Laval, 2005.

Golden, B. L., T. L. Magnanti and H. Q. Nguyen., Implementing vehicle routing algorithms, *Networks*, 7, 113-148, 1977.

Golden, B. L., E.A. Wasil, J.P. Kelly and I. M. Chao, The impact of meta-heuristics on solving the vehicle routing problem: Algorithms, problem sets and computational results, in T. Crainic and G. Laporte (editors.) *Fleet Management and Logistics*, Kluwer, 33-56, 1998.

Laporte G. and I. H. Osman, Routing Problems: A Bibliography, *Annals of Operations Research*, 61, 227-262, 1995.

Laporte, G. and F. Semet, Classical heuristics for the capacitated VRP, in P. Toth and D. Vigo (editors) *The vehicle routing problem*, Siam monographs on discrete mathematics and applications, 109-128, 2001.

Lin S., Computer solutions of the traveling salesman problem, *Bell System Technical Journal*, 44, 2245-2269, 1965.

Nelson M. D., K. E. Nygard, J. H. Griffin and W. E. Shreve, Implementation techniques for vehicle routing problem, *Computers and Operations Research*, 12, 273-283, 1985.

Osman, I. H., Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, *Annals of Operations Research*, 41, 421-451, 1993.

Paessens H., The savings algorithm for the vehicle routing problem, *European Journal of Operational Research*, 34, 366-344, 1988.

Prins, C., A simple and effective evolutionary algorithm for the vehicle routing problem, *Computers and Operations Research*, 31, 1985-2002, 2004.

Rego, C., A subpath ejection method for the vehicle routing problem, *Management Science*, 44, 1447-1459, 1998.

Rego C. and C. Roucairol, A parallel tabu search algorithm using ejection chains for the vehicle routing problem. In I. H. Osman and J. P. Kelly, editors, Mata-heuristics: Theory and Applications, Kluwer, Boston, MA, 661-675, 1996.

Reimann, M., K. Doerner and R.F. Hartl, D-Ant: Savings based ants divide and conquer the vehicle routing problem, *Computers and Operations Research*, 32, 563-591, 2004.

Renaud, J., F.F. Boctor and G. Laporte, An improved petal heuristic for the vehicle routing problem, *Journal of the Operational Research Society*, 47, 329-336, 1996.

Renaud, J., F. F. Boctor and G. Laporte, Perturbation heuristics for the pickup and delivery traveling salesman problem, *Computers and Operations Research*, 29, 1129-1141, 2002.

Rochat, Y. and E. D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, *Journal of Heuristics*, 1, 147-167, 1995.

Toth, P. and D. Vigo, The granular tabu search and its application to the vehicle routing problem, *INFORMS Journal on Computing*, 15, 333-346, 2003.

Wark, P. and J. Holt, A Repeated matching heuristic for the vehicle routing problem, *Journal of the Operational Research Society*, 45, 1156-1167, 1994.

Xu J. and J. P. Kelly, A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30, 379-393, 1996.

Yellow, P. C., A Computational modification to the savings method of vehicle scheduling, *Operational Research Quarterly*, 12, 281-283, 1970.