A Variable Neighbourhood Search Combining Constraint Programming and Lagrangean Relaxation for Solving Routing Problems

Rosa Herrero, Daniel Guimarans, Juan José Ramos, Silvia Padrón
Dpt. Telecommunication and Systems Engineering, Universitat Autònoma de Barcelona, Spain
Rosa.Herrero@uab.cat, Daniel.Guimarans@uab.cat, JuanJose.Ramos@uab.cat, Silvia.Padron@uab.cat

Keywords: Vehicle Routing, Constraint Programming, Lagrangean Relaxation, Variable Neighbourhood Search, Hybrid Algorithms

Abstract

This paper presents a methodology based on the Variable Neighbourhood Search metaheuristic, applied to the Capacitated Vehicle Routing Problem. The presented approach uses Constraint Programming and Lagrangean Relaxation methods in order to improve algorithm's efficiency. The complete problem is decomposed into two separated submodels, to which the mentioned techniques are applied to obtain a complete solution. With this decomposition, the methodology provides a quick initial solution which is rapidly improved by means of metaheuristics' iterative process. Constraint Programming and Lagrangean Relaxation are also embedded within this structure, in order to ensure constraints satisfaction and to reduce the calculation burden. Remarkable results have been obtained using this methodology, including a new best-known solution for a rarely solved 200-customers test instance and a better alternative solution for another benchmark problem.

1. INTRODUCTION

The Vehicle Routing Problem (VRP) is among the most popular research areas in combinatorial optimization. Routing vehicles to collect or deliver goods is a problem which many companies face each day, laying at the heart of many distribution systems. In practice, objectives and constraints are highly variable and, most of times, complex. In fact, real problems often require a specific modelling and solving methodology. For these reasons, several variants of the VRP have been proposed and studied since it was first defined by Dantzig and Ramser [11]. Among them, the Capacitated Vehicle Routing Problem (CVRP) is the most basic VRP, assuming a fleet of vehicles with homogeneous capacity housed in a single depot. It is so a generalization of the Travelling Salesman Problem (TSP) [15] and is therefore NP-hard [20].

Several formulations and exact algorithms have been proposed to solve the CVRP. However, for large instances the time required to solve them becomes absolutely prohibitive due to its NP-hardness. Thus, exact algorithms may only deal with small instances, up to 100 customers [10], solving them to optimality. Numerous heuristics and metaheuristics have also been studied for different VRP variants. In most cases, these methods may solve larger instances but loosing optimality guarantees. Among metaheuristics, Variable Neighbourhood Search (VNS) [16] is a quite recent method with far less application examples in VRP research. However, some interesting results have been obtained even applying the simplest VNS algorithm [9] [14] [19]. For this reason, VNS has been selected as the general framework where to embed Constraint Programming (CP) and Lagrangean Relaxation (LR) approaches to the CVRP. By using these two well-known paradigms within the VNS local search process, calculation time may be reduced with respect to classical VNS schemes.

The remainder of this article is structured as follows. Section 2 provides a general overview of CVRP formulation, emphasizing the decomposition used in the proposed method. Section 3 is devoted to the proposed method, based on the VNS metaheuristic; the general algorithm, moves used within its structure and the adapted LR-method are introduced in this section. Next, computational results are presented and discussed. Finally, some conclusions are outlined in the last section.

2. PROBLEM FORMULATION

The symmetric CVRP can be considered as a complete undirected graph $G = (I, E)$, connecting the vertex set $I = \{1, 2, ..., n\}$ through a set of undirected edges $E = \{(i, j) | i, j \in I\}$. The edge $e_{ij} \in E$ has associated a travel cost c_{ij}, supposed to be the lowest cost route connecting node i to node j. Each vertex $i \in I - \{1\}$ has a nonnegative demand q_i, while vertex 1 corresponds to a depot without associated demand. A fixed fleet of m identical vehicles $V = \{1, 2, ..., m\}$, each of capacity Q, is available at the depot to accomplish the required tasks.

Solving the CVRP consists of determining a set of m routes whose total travel cost is minimised and such that: (a) each customer is visited exactly once by a single vehicle, (b) each route starts and ends at the depot and (c) the total demand of the customers assigned to a route does not exceed the vehicle capacity Q. Therefore, a solution to the CVRP is a set of m cycles sharing a common vertex at the depot. Usually, the fleet size is not fixed and minimizing the total number of used vehicles becomes an additional objective.

In the proposed model, the CVRP has been divided into two subproblems, concerning customers' allocation and routing optimization separately. The first is aimed to assign customers to the minimum number of required vehicles fulfilling capacity limitations. The latter is used to solve each independent route to optimality, giving the best solution for a particular allocation. Thus, routing optimization process can be viewed as solving a set of m independent symmetric TSP. CP is used to find a feasible solution in terms of capacity, while routing problems are solved by using LR.

2.1. Capacity problem

Constraint Programming is a powerful paradigm for representing and solving a wide range of combinatorial problems. Problems are expressed in terms of three entities: variables, their corresponding domains and constraints relating them. The problems can then be solved using complete techniques such as depth-first search for satisfaction and branch and bound for optimization, or even tailored search methods for specific problems [18].

The proposed customers' allocation subproblem uses two lists of variables. A list R of size n, with integer domains $R_i \in [1..m] | i \in I$, indicates which vehicle is serving the i^{th} customer. Q_v is a list of m variables with real domain $Q_v \in [0..Q]$ used to trace the cumulative capacity at each one of the m routes. Therefore, capacity constraints are enforced through domains definition since Q_v cannot get higher values than the maximum capacity Q.

A set of dimension $m \times n$ of binary variables B has been introduced to relate R and Q_v values. For each vehicle $v \in V$, a list of n binary variables $B_{vi} \mid i \in I$ is defined, taking value 1 whenever customer i is assigned to vehicle v and 0 otherwise. Since each customer i is visited by a single vehicle, for all values of v the binary variable B_{vi} can take value 1 only once. This constraint is expressed in terms of the global constraint *cardinality_atmost* [6] aiming to ensure a faster propagation [7].

The binary set B and allocation variables R are related through the following statement:

$$R_i = r_i \rightarrow B_{r_i i} = 1 \ \forall i \in I \tag{1}$$

Expression (1) states that the i^{th} element of the r_i list of B will have value 1 whenever the i^{th} component of R takes value r_i. The global constraint *cardinality_atmost* ensures propagation so all values of B_{vi} $v \in V \setminus r_i$ are set to 0 automatically. Therefore, cumulative capacities can be traced simply by using the following equation:

$$Q_v = \sum_{i \in I} B_{vi} q_i \ \forall v \in V \tag{2}$$

The proposed formulation is used to find a partial initial solution fulfilling capacity constraints. By solving resultant routing problems, which are always feasible because they do not contain any additional constraints, a complete initial solution may be easily obtained in most cases. Thus, capacity problem's goal is to find a feasible solution with the minimum number of required vehicles. With this objective, a depth-first search method is applied to find a feasible solution that uses all available vehicles. A vehicle is removed from the list and the process is repeated recursively. The algorithm stops when unfeasibility is reached, returning the last feasible solution found in the previous iteration.

2.2. Routing Problem

The routing problem, tackled for each vehicle separately, can be viewed as a TSP instance. For each vehicle v, the related TSP can be considered as a complete undirected graph $G = (I_v, E_v)$, connecting assigned customers $I_v = \{i \in I | R_i = v\}$ through a set of undirected edges $E_v = \{(i, j) \in E | i, j \in I_v\}$. The solution is a path connected by edges belonging to E_v that starts and ends at the depot ($i = 1$) and visits all assigned customers.

The proposed mathematical formulation requires defining the binary variable x_e to denote that the edge $e_{ij} \in E_v$ is used in the path. That is $x_e = 1$ if customer j is visited immediately after i; otherwise $x_e = 0$. Thus, the formulation for the TSP problem is as follows:

$$\min \sum_{e \in E_v} c_e x_e \tag{3}$$

subject to

$$\sum_{e \in \delta(i)} x_e = 2 \ , \qquad \forall i \in I_v \tag{4}$$

$$\sum_{e \in E_v(S)} x_e \leq | S | - 1 \ , \qquad \forall S \subset I_v \ , \ | S | \leq \frac{1}{2} | I_v | \tag{5}$$

where

- $\delta(i) = \{e \in E_v : \exists j \in I_v, \ e = (i, j) \ or \ (j, i)\}$ represents the set of arcs whose starting or ending node is i.
- $E_v(S) = \{e_{ij} \in E_v : i, j \in S\}$ represents the set of arcs whose nodes is in the subset S of vertices.
- $n_v = |I_v|$
- c_e is the associated cost to the undirected edge $e_{ij}(e_{ji})$.

Constraint (4) states that every node $i \in I_v$ must be visited once, that is, every customer must have two incident edges. Subtour elimination constraint (5) states that the tour must be a Hamiltonian path, so it cannot have any subcycle. Then a feasible solution of the TSP should, by definition, also satisfy constraints (a) and (b) of the CVRP, minimising the total travel cost of the route.

Algorithm 1 The Proposed LR-based Method

```
0   Initialization
1       Initialize parameters $u^0 = 0; \delta_0 = 2; \rho = 0.95; \alpha_L = 1/3$
2       Obtain an $UB$ applying Nearest Neighbour Heuristic
3       Initialize $\overline{L} = L(u^0) + \alpha_L(UB - L(u^0))$
4   Iteration $k$
5       Solve the Lagrangean function $L(u^k)$
6       Check the subgradient $\gamma_i^k = 2 - \sum_{e\in\delta(i)} x_e$
7           if $\| \gamma^k \|^2 = 0$ then Optimal solution is found $\Rightarrow$ EXIT
8           if $\| \gamma^k \|^2 < \xi$ then apply a heuristic to improve the $UB$
9       Check the parameter $\overline{L}$
10      Calculate the step-size $\lambda_k = \delta_k \frac{\overline{L} - L(u^k)}{\|\gamma^k\|^2}$
11      Update the multiplier $u^{k+1} = u^k + \lambda_k \gamma^k$
12      $k \leftarrow k + 1$
```

3. METHODOLOGY

The described problem has been tackled using a hybrid approach. In the proposed methodology, a general VNS framework has been chosen to embed CP and LR paradigms, in order to improve algorithm's performance. A complete and revised description of different VNS algorithms can be found in [13].

During algorithm's initialization, CP is used to find an initial feasible solution in terms of capacity. CP is also used to check solutions feasibility within diversification and local search processes. In turn, a tailored LR method is applied to calculate routes every time a partial solution is generated either during initialization, diversification or local search processes. Applying LR allows avoiding routing post-optimization methods which use single-route moves. So, the proposed LR approach provides quick optimal routes and, at the same time, permits reducing algorithm's definition and complexity.

3.1. Proposed Lagrangean Relaxation method

Given the assigned customers to each vehicle, a Lagrangean Relaxation approach is used to solve associated routing problems. LR is a well-known method to solve large-scale combinatorial optimization problems. It works by moving hard-to-satisfy constraints into the objective function associating a penalty in case they are not satisfied. An excellent introduction to the whole topic of LR can be found in [12].

LR exploits the structure of the problem, so it reduces considerably problem's complexity. However, it is often a major issue to find optimal Lagrangean multipliers. The most commonly used algorithm is the Subgradient Optimization (SO). The main difficulty of this algorithm lays on choosing a correct step-size λ_k in order to ensure algorithm's convergence [17].

Therefore, the proposed method combines the SO algorithm with a heuristic to obtain a feasible solution from a dual solution. It can get a better upper bound UB, so it improves the convergence on the optimal solution departing from an initial UB obtained with a Nearest Neighbour Heuristic. If the optimal solution is not reached at a reasonable number of iterations, the proposed method is able to provide a feasible solution with a tight gap between the primal and the optimal cost.

The proposed LR relaxes the constraint set requiring that all customers must be served (4), since all subcycles can be avoided constructing the solution x as a 1-tree. Actually, a feasible solution of the TSP is a 1-tree having two incident edges at each node [15]. The advantage is that finding a minimum 1-tree is relatively easy.

The Lagrangean Dual problem obtained from the TSP formulation, moving into the objective function equalities (4) weighting them with a multiplier u, is:

$$\max_{u\in\Re^{n_v}} L(u) \tag{6}$$

where

$$L(u) = \min_{x\ 1-tree} \sum_{e\in E_v} c_e x_e + \sum_{i\in I_v} u_i (2 - \sum_{e\in\delta(i)} x_e) \tag{7}$$

The proposed LR-based method, shown in Algorithm 1, can be considered a specification of the Lagrangean Metaheuristic [8], where a tailored heuristic to get a feasible solution improving the UB is only applied when the 1-tree is nearly a Hamiltonian path (step 8).

As mentioned, algorithm's convergence is critically influenced by the step-size λ_k. As the LB and the UB are normally unknown or bad estimated, the use of a parameter $\overline{L}$, such that $LB \le \overline{L} \le UB$, is proposed. The calculation of the step-size then turns into $\lambda_k = \delta_k \frac{\overline{L}-L(u^k)}{\|\gamma^k\|^2}$ with $0 < \delta_k \leq 2$. Convergence is so guaranteed if the term $\overline{L} - L(u^k)$ tends to zero. In turn, convergence efficiency can be improved as long as the new $\overline{L}$ parameter gets closer to the (unknown) optimal value. The main idea is very simple: as the algorithm converges to the solution, new better LB are known and new better UB estimations can be obtained by using the heuristic designed to get feasible solutions. Therefore, the parameter $\overline{L}$ is updated according to $\overline{L} = L(u^k) + \alpha_L(UB - L(u^k))$ with $0 < \alpha_L < 1$. Finally, the parameter δ_k is initialized to the value 2 and is updated as [22] suggest.

3.2. Inter-route Moves

VNS metaheuristic is based on exploring alternatively different neighbourhoods around a known feasible solution. In order to establish these neighbourhoods, different moves are to be defined.

Since results provided by the LR method are optimal, no routing post-optimization process is needed. Thus, using LR for solving routing subproblems allows avoiding the definition of single-route moves. For this reason, only four different inter-routes classic moves [21] have been defined to be used in the VNS metaheuristic:

- *Relocate*: moves a customer from one route to a different one.
- *Swapping*: exchanges two customers belonging to two different routes.
- *Chain*: is a specialization of 3-opt that swaps sections of two contiguous customers from two different routes.
- *Ejection chain*: swaps the end portions of two different routes.

3.3. Variable Neighbourhood Search Framework

A general VNS framework, as explained in [13] (Algorithm 2), has been implemented embedding CP and LR methods. At each iteration, a local minimum is reached departing from an initial solution. A diversification process (*shaking*) ensures that different regions of the search space are explored. Every time the shaking process generates a new point to diversify the search, its feasibility is immediately checked using CP. If the generated point is unfeasible, the process is repeated until a new feasible point is found.

In the proposed implementation, all four mentioned moves have been selected to be used in shaking (N_k) and local search (N_l) neighbourhoods. As a first step for the algorithm, an initial feasible solution is found using CP and LR described methods. CP is used to assign all customers to available vehicles fulfilling capacity constraints, while resulting routes are solved to optimality by means of LR. This solution is improved by means of a Variable Neighbourhood Descent (VND) method [13] using the four defined moves. Thus, the VNS starts from a local optimum.

At step 4, a new point is generated at random from the k^{th} neighbourhood $N_k(x)$ of x in order to diversify the search. New allocation's feasibility is immediately checked using CP. If the generated point is unfeasible, the process is repeated until a new feasible point is found. If the valley surrounding the solution x is large, a thorough diversification should be done aiming to avoid getting trapped in a local optimum. For this reason, the implemented shaking process is repeated several times. Solutions' values are ignored until the last iteration, when routes are calculated using LR to provide a complete solution.

The local search process for each neighbourhood $N_l(x')$ (step 8) performs an exhaustive exploration with a best-accept

Algorithm 2 Algorithm Variable Neighbourhood Search

```
0   Initialization. Select the set of neighbourhood structures $N_k$, for $k = 1,...,k_{max}$, that will be used in the shak-
    ing phase, and the set of neighbourhood structures $N_l$ for $l = 1,...,l_{max}$ that will be used in the local search;
    find an initial solution $x$; choose a stopping condition (usually, time or maximum number of iterations);
1   Repeat the following sequence until the stopping condition is met:
2       Set $k \leftarrow 1$;
3       Repeat the following steps until $k = k_{max}$:
4           (a) Shaking. Generate a point $x'$ at random from the $k^{th}$ neighbourhood $N_k(x)$ of $x$;
5           (b) Local search by VND.
6               (b1) Set $l \leftarrow 1$;
7               (b2) Repeat the following steps until $l = l_{max}$;
8                   - Exploration of neighbourhood. Find the best neighbour $x''$ of $x'$ in $N_l(x')$;
9                   - Move or not. If $f(x'') < f(x')$ set $x' \leftarrow x''$ and $l \leftarrow 1$; otherwise set $l \leftarrow l+1$;
10          (c) Move or not. If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and
            continue the search with $N_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k+1$;
```

strategy. New solutions' feasibility is also checked using CP. Whenever a solution is proved feasible, LR is used to recalculate only modified routes. Therefore, this approach permits to consider only two routes per solution, reducing the computation time. Finally, the best neighbour x'' is chosen in terms of its solution value $f(x'') = \sum_{v \in V} UB_v$. If its value is lower than the original $f(x')$, the solution is updated and neighbourhoods' exploration is restarted (step 9).

When the VND process reaches a local optimum, no solution improvement may be found according to defined neighbourhoods (step 10). If this local optimum is better than the incumbent, it is accepted as the current solution $x \longleftarrow x''$ and the search is restarted from the first shaking neighbourhood. Otherwise, the algorithm keeps x as the best solution found so far and continues exploring the next neighbourhood. If there are no remaining neighbourhoods to be explored, the process is restarted (step 2) until the stopping condition is met (step 1).

4. APPLICATION AND RESULTS

The methodology described in the present paper has been implemented in Java and linked to the open-source CP software system ECLiPSe 6.0 [3]. All tests have been performed on a non-dedicated server with an Intel i5 processor at 2.66GHz and 16GB RAM. In general, five to seven processes were launched in parallel to solve different problems.

A total of 97 classical CVRP benchmark instances obtained from www.branchandcut.org have been used to test the efficiency of the proposed approach. Only those instances whose distance is defined as Euclidean or Geographic have been selected, in order to ensure triangular inequality's fulfilment. Table 1 shows the total number of problems chosen from each class, as well as how many have been successfully solved to optimality. This table also shows the average (% Dev.), maximum (% Max) and minimum (% Min) deviation from the best published value for those problems that could not be solved to optimality after 40 iterations. A low deviation is observed for most problem sets, comparable to results obtained by means of other metaheuristics.

As mentioned, the initial solution is obtained by solving separately capacity and routing problems. This approach is able to provide a low-quality quick solution, since both subproblems are easily solved but variables are unlinked. However, this solution may be highly improved applying a VND method, providing an initial solution whose value is usually close to the final result. Values and times needed to solve problems of class A are shown on Table 2, both applying a VND improving method and using the initial solution provided by the CP/LR scheme. This table also shows the gap between best-known and final solutions, as well as the number of iterations needed to reach the best value (-1 in case the best-known value is not reached). In many cases, it can

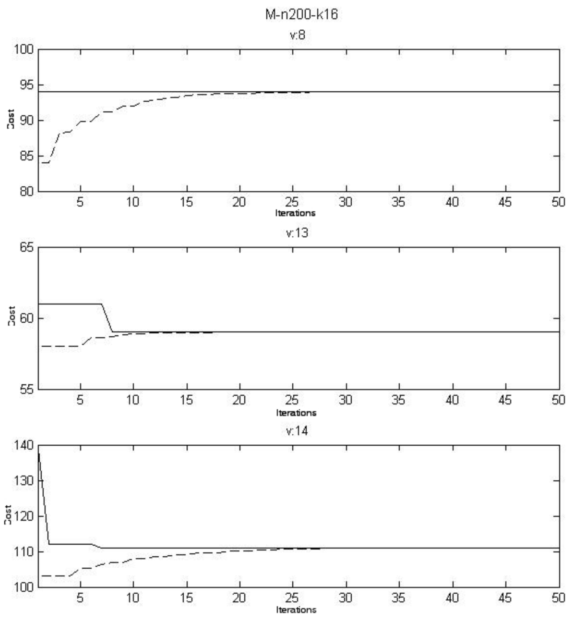

Figure 1. Convergence of LB (dashed line) and UB (solid line) in three routes of problem M-n200-k16.

be observed that an initial good solution does not lead to a better final one. Since the shaking process makes the algorithm start from a randomly chosen feasible solution, initial solution quality does not affect dramatically algorithm's performance. For this reason, it may be preferable to get a worse initial solution but with a lower computation time.

Furthermore, the use of LR ensures the partial optimality of all solutions from the routing perspective. The reason is that the proposed approach can optimally solve all TSP instances, due to the number of associated customers is always low. As can be seen in Figure 1, LB and UB converge rapidly, keeping their gap between 0 and 10^{-10} and guaranteeing so solution's optimality. In addition, LR solves all routes in negligible times. Thus, LR has demonstrated to be an efficient alternative for intra-route optimization processes.

The proposed methodology performs similarly both for small and large instances. Thus, its applicability is not restricted. It is remarkable that the algorithm eventually reaches the optimal solution for smaller problems (50 customers or less), but it stops near the optimum for larger instances. Table 3 shows that the presented approach is able to provide state-of-art results for large problems. As observed, final values are normally close to best known solutions. It is also remarkable the result obtained for the largest selected test instance G-n262-k25, which stays slightly over (0.44%) the best known value [14].

Finally, two problems deserve special attention: M-n200-k16 and P-n55-k8. The proposed methodology is able to find

Table 1. Summary of results obtained with the proposed methodology.

Class	Problems	Opt.	No opt.	Not solved	% Dev.	% Max	% Min
A	27	14 (51.85%)	13 (48.15%)	0 (0%)	0.65	2.14	0.14
B	23	12 (52.17%)	11 (47.83%)	0 (0%)	1.79	4.28	0.13
E	11	5 (45.45%)	6 (54.55%)	0 (0%)	0.68	1.59	0.41
F	3	1 (33.33%)	2 (66.67%)	0 (0%)	4.22	4.22	4.22
G	1	0 (0%)	1 (100%)	0 (0%)	0.44	0.44	0.44
M	5	1 (20%)	4 (80%)	0 (0%)	2.44	4.35	0.69
P	24	12 (50%)	10 (41.67%)	2 (8.33%)	0.88	3.3	0.15
TSPLib	3	1 (33.33%)	2 (66.67%)	0 (0%)	2.28	3.23	1.33
	97	48 (49.48%)	47 (48.45%)	2 (2.06%)	1.67	2.94	0.94

Table 2. Results obtained for class A problems. Results marked with (1) have been obtained applying a VND method to improve the initial solution, while those marked with (2) correspond to an initial solution obtained by means of the CP/LR scheme.

Problem	BKS	Init. Sol.		Time (s)		Fin. Sol.		Total time (s)		Gap BKS-FS (%)		# iter.	
		(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
A-n32-k5	784	989	1243	1.75	0.06	784	784	13.12	24.50	-	-	1	2
A-n33-k5	661	734	1185	1.71	0.01	661	661	24.52	16.24	-	-	1	1
A-n33-k6	742	792	1289	1.84	0.01	742	742	160.10	161.32	-	-	26	26
A-n34-k5	778	816	1259	2.32	0.01	778	778	42.20	14.40	-	-	4	1
A-n36-k5	799	832	1207	2.57	0.01	799	799	152.90	186.96	-	-	14	20
A-n37-k5	669	697	960	3.26	0.01	669	669	22.90	36.51	-	-	1	1
A-n37-k6	949	966	1393	2.85	0.01	949	949	37.73	23.59	-	-	4	2
A-n38-k5	730	730	1240	2.89	0.01	730	730	9.09	12.82	-	-	1	1
A-n39-k5	822	826	1291	4.00	0.02	825	822	371.51	304.94	0.36	-	-1	32
A-n39-k6	831	891	1523	3.37	0.01	833	833	634.35	565.19	0.24	0.24	-1	-1
A-n44-k6	937	991	1547	7.11	0.01	939	937	675.92	298.89	0.21	-	-1	14
A-n45-k6	944	1006	1826	6.15	0.02	963	976	367.14	413.02	2.01	3.39	-1	-1
A-n45-k7	1146	1161	1768	11.28	0.02	1146	1146	198.88	147.53	-	-	7	3
A-n46-k7	914	986	1711	7.61	0.02	914	914	372.14	93.30	-	-	13	1
A-n48-k7	1073	1183	1840	7.85	0.02	1086	1073	1094.28	588.82	1.21	-	-1	20
A-n53-k7	1010	1088	1841	15.54	0.03	1010	1017	1208.76	1471.50	-	0.69	33	-1
A-n54-k7	1167	1178	1883	20.17	0.03	1172	1174	1391.63	1517.61	0.43	0.6	-1	-1
A-n55-k9	1073	1084	2074	19.62	0.03	1073	1073	731.67	262.96	-	-	20	6
A-n60-k9	1354	1485	2224	22.19	0.04	1354	1364	339.23	2191.01	-	0.74	4	-1
A-n61-k9	1034	1158	2045	22.39	0.03	1037	1036	1321.61	1310.40	0.29	0.19	-1	-1
A-n62-k8	1288	1363	2344	27.13	0.03	1290	1312	2283.54	2611.07	0.16	1.86	-1	-1
A-n63-k10	1314	1402	2275	29.79	0.04	1319	1325	2438.17	2565.06	0.38	0.84	-1	-1
A-n63-k9	1616	1723	2659	23.07	0.03	1630	1616	1972.98	843.10	0.87	-	-1	14
A-n64-k9	1401	1483	2211	34.41	0.04	1418	1414	2253.30	2429.19	1.21	0.93	-1	-1
A-n65-k9	1174	1315	2331	26.77	0.04	1178	1182	2203.19	1968.11	0.34	0.68	-1	-1
A-n69-k9	1159	1194	2463	71.32	0.04	1167	1167	3227.33	3320.63	0.69	0.69	-1	-1
A-n80-k10	1763	1963	3165	89.32	0.05	1783	1787	7196.77	7781.76	1.13	1.36	-1	-1

Table 3. Results obtained for some representative large problems. Improved solutions are marked in bold.

Problem	BK	Init.Sol.	Fin.Sol.	Gap BK-FS (%)	# iter.
E-n101-k8	817	1628	817	-	18
E-n101-k14	1067	2106	1080	1.22	-1
M-n101-k10	820	1091	840	2.44	-1
M-n121-k7	1034	1227	1079	4.35	-1
M-n151-k12	1015	2481	1022	0.69	-1
M-n200-k16	1371	3287	**1335**	-2.63	13
M-n200-k17	1275	3201	1304	2.27	-1
G-n262-k25	5685	14563	5710	0.44	-1

a new best solution for the first and an alternative solution for the latter. For the test instance M-n200-k16, a new best solution with a value of 1335 has been obtained (Figure 2). To the best of our knowledge, only one previous feasible solution with a cost of 1371 was known [14]. Taking into account the best known lower bound for this instance (1256.4) [5], the solution found reduces the gap between bounds from the previous value of 8.36% to 5.89%.

The solution found for the test instance P-n55-k8 becomes an alternative to the known optimum [4]. For this case, a solution with a value of 576 using only 7 vehicles has been found. As far as we know, only two previous works have also presented this alternative value as the best known solution for this instance [1] [2], while most authors keep the original value of 588 using 8 vehicles as optimal.

5. CONCLUSIONS

The present paper has presented a methodology combining CP and LR within a general VNS framework. This scheme

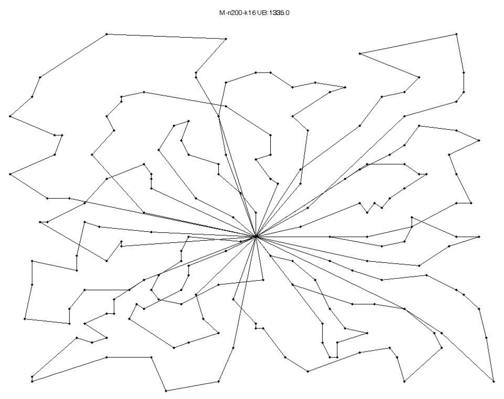

Figure 2. The best knwon solution so far for the test instance M-n200-k16.

has been used to tackle the CVRP, obtaining state-of-art results comparable to other metaheuristics. As a hint, it has provided a new best-known solution for the test instance M-n200-k16 and an alternative solution for the problem P-n55-k8, while keeping a competitive performance for most CVRP instances.

In the proposed approach, the CVRP has been decomposed into two separated subproblems. The first one is aimed to assign customers to vehicles in terms of capacity, while the second is used to optimize corresponding routes. This approach allows reducing the computation time, since problems to be solved are far less complex than the original CVRP, although still NP-hard. In fact, the allocation problem may be assimilated to the Bin Packing Problem, while routing subproblem's goal is solving a set of TSP. In both cases, two well-known paradigms aimed to solve combinatorial problems, CP and LR, have been applied obtaining good results. Thus, combining this decomposition with selected techniques provides a methodology able to get a quick initial solution to the CVRP problem, even for larger instances. Although solution's quality may be low, it may be rapidly improved by applying a local search process, such as the VND algorithm.

Furthermore, the proposed LR-based method presents an improved convergence with respect to the SO classical algorithm. It may provide optimal routes when the number of customers is relatively small, as it is for all CVRP benchmark instances. Combining these characteristics with the adopted approach to the CVRP allows reducing the computation time. On the one hand, the selected decomposition makes LR only necessary to recalculate two routes at each iteration. On the other hand, the LR-based method is faster and simpler than other routing post-optimization processes, since no intra-route moves are to be defined and it is less likely to get trapped in iterative processes.

Finally, several lines for future research are open. First, different VNS schemes are to be studied, such as Variable Neighbourhood Decomposition Search, whose shaking process can be improved by embedding CP techniques. Second, heuristic methods are to be included into the neighbourhood exploration phase. Finally, the presented methodology is to be adapted to different VRP variants, especially those including time windows or pick-up and delivery side constraints.

ACKNOWLEDGMENTS

This work has been supported by the Agència de Gestió d'Ajuts Universitaris i de Recerca (AGAUR) of the Generalitat de Catalunya (project 2009 SGR 629).

REFERENCES

[1] E. Alba and B. Dorronsoro. A hybrid cellular genetic algorithm for the capacitated vehicle routing problem. In A. Abraham, C. Grosan, and W. Pedrycz, editors, *Engineering Evolutionary Intelligent Systems*, pages 379–422. Springer-Verlag, 2008.

[2] I. Altinel 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.

[3] K. Apt and M.G. Wallace. *Constraint Logic Programming using ECLiPSe*. Cambridge University Press, 2007.

[4] P. Augerat, J.M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. Technical Report 949-M, Université Joseph Fourier, Grenoble, France, 1995.

[5] R. Baldacci, N. Christofides, and A. Mingozzi. An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts. *Mathematical Programming*, 115(2):351–385, 2008.

[6] N. Beldiceanu, M. Carlsson, and J.-X. Rampon. Global constraint catalog. Technical Report 2005:08, Swedish Institute of Computer Science (SICS), Kista, Sweden, 2005.

[7] C. Bessiere. Constraint propagation. In F. Rossi, P. van Beek, and T. Walsh, editors, *Handbook of Constraint Programming*, pages 29–83. Elsevier, 2006.

[8] M. Boschetti and V. Maniezzo. Benders decomposition, lagrangean relaxation and metaheuristic design. *Journal of Heuristics*, 15:283–312, 2009.

[9] O. Bräysy. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368, 2003.

[10] J.-F. Cordeau, G. Laporte, M.W.P. Savelsbergh, and D. Vigo. Vehicle routing. In C. Barnhart and G. Laporte, editors, *Handbook in Operations Research and Management Science*, volume 14, pages 367–428. Elsevier, 2007.

[11] G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

[12] M.L. Fisher. The lagrangean relaxation method for solving integer programming problems. *Management Science*, 27:1–18, 1981.

[13] P. Hansen and N. Mladenovic. A tutorial on variable neighborhood search. Technical Report G-2003-46, Groupe d'Études et de Recherche en Analyse des Décisions (GERAD), Montreal, Canada, 2003.

[14] G. Hasle and O. Kloster. Industrial vehicle routing. In G. Hasle, K.-A. Lie, and E. Quak, editors, *Geometric Modelling, Numerical Simulation, and Optimization*, pages 397–435. Springer-Verlag, 2007.

[15] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees: Part ii. *Mathematical Programming*, 1:6–25, 1971.

[16] N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[17] G. Reinelt. The traveling salesman: computational solutions for tsp applications. *Lecture notes in computer science*, 840, 1994.

[18] F. Rossi, P. Van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[19] L.M. Rousseau, M. Gendreau, and G. Pesant. Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of Heuristics*, 8:43–58, 2002.

[20] M.W.P. Savelsbergh. Local search in routing problems with time windows. *Annals of Operations Research*, 4:285–305, 1985.

[21] M.W.P. Savelsbergh. Computer aided routing. Technical report, Centrum voor Wiskunde en Informatica, Amsterdam, Netherlands, 1988.

[22] R. Zamani and S.K. Lau. Embedding learning capability in lagrangean relaxation: An application to the travelling salesman problem. *European Journal of Operational Research*, 201(1):82–88, 2010.

Biography

Rosa Herrero is a PhD Student holding a fellowship at the Telecommunication and Systems Engineering Department at the Universitat Autònoma de Barcelona. She is a member of the Logisim Research Group. Her research interests are integer programming, lagrangean relaxation and techniques to solve industrial combinatorial problems.

Daniel Guimarans is a PhD Student and Assistant Teacher at the Telecommunication and Systems Engineering Department at the Universitat Autònoma de Barcelona. He is an active member of the Logisim Research Group. His research interests are constraint programming, simulation and hybridisation of different techniques to solve industrial combinatorial problems.

Juan José Ramos is an Associated Professor at the Telecommunication and Systems Engineering Department at the Universitat Autònoma de Barcelona. He is coordinator and an active member of the Logisim Research Group. His research interests are modelling, simulation and optimisation of logistics problems.

Silvia Padrón is a PhD Student holding a fellowship at the Telecommunication and Systems Engineering Department at the Universitat Autònoma de Barcelona. She is a member of the Logisim Research Group. Her research interests are modelling, simulation and optimisation of logistics problems.