A Two-Phase Vehicle based Decomposition Algorithm for Large-Scale Capacitated Vehicle Routing with Time Windows

Matija Gulic
Protok d.o.o.
Zagreb, Croatia
Email: matija.gulic.hr@gmail.com

Drazen Lucanin
Information System Institute
Vienna University of Technology
Email: drazen@infosys.tuwien.ac.at

Nina Skorin-Kapov
Faculty of Electrical Engineering and Computing
University of Zagreb
Email: nina.skorin-kapov@fer.hr

***Abstract*—With significant advances in computing power during recent years, increasingly complex variants of the vehicle routing problem (VRP) with added constraints are coming into focus. VRP is a combination of the classical traveling salesman and bin packing problems, with many real world applications in various fields – from physical resource manipulation planning to virtual resource management in the ever more popular cloud computing domain. In this paper, we consider large-scale VRP problem instances with time window constraints. Due to their complexity, we propose a solution approach based on the divide and conquer paradigm, decomposing problem instances into smaller, mutually independent sub-problems which can be solved using traditional algorithms and integrated into a global solution of reasonably good quality. Numerical results indicate the efficiency and scalability of the proposed approach, making it highly applicable to large-scale realistic VRP problem instances.**

***Index Terms*—vehicle routing problem, time windows, large-scale, greedy search, decomposition, scalability, optimization**

I. Introduction

Travel logistics have always been important to minimize expenses (be it raw materials such as fuels or the manpower behind it) and mantain a good quality of service. Today, they have become even more so, when we know that the environment is at stake as well – in 2010 the transport sector was responsible for 23% of all the CO_2 emissions in the EU [1]. A good formal representation of real-life transportation and service delivery problems is the capacitated vehicle routing problem with time windows (VRPTW). It is a problem of having to minimize the distance that vehicles need to travel in order to deliver a service with multiple realistic constraints regarding space, time and capacity. It was shown that solutions for this problem could be applied to real-life domains and significantly improve their efficiency – up to 30% in some cases [2]. Consequently, in this paper we propose improvements to existing solutions of the VRPTW.

VRPTW is an NP-hard problem [3], so the aim is to find an approximate solution of high-enough quality. There are many solutions to the VRPTW and the most promising ones, such as [4], [5] apply the so-called optimization decomposition technique to divide the search into separate constraint dimensions. The method we propose here applies a decomposition based on separate vehicles and then uses parallel local search for optimisation within each decomposed dimension.

The main *contribution* of this paper is an efficient decomposition heuristic comprised of an initial greedy step followed by parallel local search.

The rest of the paper is organised as follows. Section II lists the related work in the field. Section III gives a formal problem statement and introduces the considered model. In Section IV we present our decomposition method and parallel local search algorithm. An evaluation method and the numerical results are described in Section V. Section VI gives possible future research directions and finally, section VII concludes the paper.

II. Related work

A lot of work has been invested in creating efficient solvers for the vehicle routing problem.

A constraint decomposition approach, where a possible solution is optimized for each of the constraints in turn and then combined, was proposed in [4], [5]. The problem space is decomposed to spatial, time and vehicle dimensions. In our proposed solution we also decompose the problem space based on the vehicle dimension.

As for the optimization techniques inside the VRP domain [6], various algorithms are used. A simulated-annealing-like local search was proposed in [7]. In [8] a tabu search optimisation is used. The naive ejection chain method for local neighbourhood searching is proposed in [9] with a high potential for solution diversification and total vehicle number reduction. Models for evolutionary methods were presented in [10], [11]. Ant-colony optimisation was used in [12]. Our optimisation approach is based on a parallel local search algorithm once the solution space is decomposed into small-enough instances.

III. Problem and model statement

The vehicle routing problem with time windows is an extension of the well-known vehicle routing problem (VRP, defined in [13]). The VRP is described as follows: "A set of n customers must be serviced from a central office using vehicles of equal given capacity". Each customer must be served from exaclty one vehicle. Usually two objective criteria are used, the primary to minimize the number of vehicles and the secondary to minimize the total travel distance. An additional constraint associated with the time windows extension is that every customer must be serviced within a given time frame. If a vehicle arrives earlier it must wait for the window opening time. If the vehicle arrives after the end of the time window, the solution is not valid. Every customer has the following parameters defined:

- readyTime - window opening time
- dueDate - window closing time
- serviceTime - time needed for the customer to be serviced
- geographical data

Other parameters:

- distance - geographical distance between two customers
- timeDistance - time needed to travel from one customer to another
- windowTime - difference between dueDate and readyTime

In the algorithm we propose here to solve the VRP, we consider a single objective criterion which is to minimize the total number of vehicles. Generally, a smaller vehicle number implies a smaller travel distance. Thus, to simplify the algorithm without loss of generality, the secondary criterion will not be considered here.

IV. A two-phase Vehicle based Decomposition algorithm

The presented algorithm is divided in two major parts. Firstly, an initial solution is generated using a fast greedy algorithm. Then the number of vehicles is reduced using parallel customer insertion. One of the most important advantages of the implemented parallel method is easy patching of sub-results into a global result which fully meets the defined constraints.

The proposed algorithm is illustrated in Fig. 1, resembling a state-transition diagram. It can be seen that our algorithm consist of 5 states and 5 transitions. The start and end states differ from the middle states because they represent the initial points (customers) and the final points (vehicle routes). The second state is an abstraction for selecting a route which will be removed. The center state expresses parallel insertion of customers from the removed route (second state). In the fourth state we decide which of the existing vehicles (routes) will accept customers from the deleted route. The central states presents vehicles as lines and customers as points on those lines. Transitions are intuitive and describe what will be done in the next state. A special type of transition is a backward transition which allows removing routes, one by one, if the current route is not removed or if the current vehicle number has been reduced.

A. Greedy initial solution creation

To create an initial solution, a greedy algorithm is used that creates a list of customers for each vehicle. The input parameters are the initial customer, the vehicle capacity and a list of unvisited customers (that still need to be visited). The pseudocode of phase 1 of the algorithm follows:

```
function GREEDY
    while $unvisitedCustomer\ exists$ do
        $start\ newRoute$
        $set\ newRoute.first(home\ depot)$
        $nextCustomer \leftarrow greedy.getNextCustomer$
        while $nextCustomer\ exists$ do
            $newRoute.add(nextCustomer)$
            $nextCustomer \leftarrow greedy.getNextCustomer$
        end while
        $set\ newRoute.last(home\ depot)$
        $add\ newRoute\ to\ result[route]$
    end while
end function
function GETNEXTCUSTOMER
    $get\ unvisited\ customers\ which\ meets\ route\ capacity\ limitation$
    $get\ customers\ which\ carry\ out\ time\ restrictions$
    $customer \leftarrow select\ customer\ with\ minimal\ OBJ\ value$
    if $customer\ selected$ then
        return $customer$
    else
        return $null$
    end if
end function
```

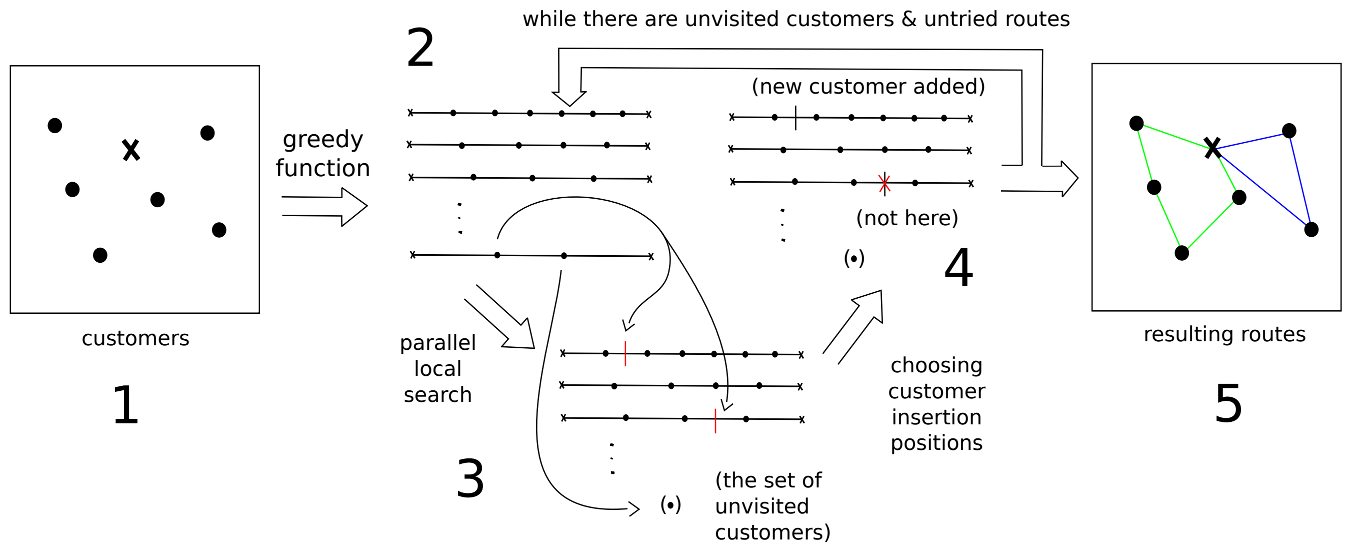
In the algorithm, we basically build a list of customers to visit by first taking into consideration the minimization criterion expressed with the getNextCustomer function. Many optimization criteria were tested, including OBJ=

- Distance
- ReadyTime
- DueDate
- VisitedTime
- WindowTime
- fun1: $fun1 = x * distance + y * waitTime + z * (dueTime - visitedTime)$

We focused on a custom function fun1, which combines three main criteria: distance, wait time and urgency. Once the vehicle is full ($current\_load \geq capacity$) the vector of customers is taken as a sub-problem and a new vector is created for the next vehicle, for as long as there are unvisited customers.

B. Vehicle reduction

Once an initial solution is created, the parallel optimisation algorithm (phase 2) starts. The main idea is to divide the current result among independent units. This is possible by assigning every vehicle to a separate parallel job. Iterating through the appointed vehicles list, the current vehicle is deleted and its customers are sequentially redirected to the



while there are unvisited customers & untried routes
1
customers
greedy function
2
parallel local search
3
(the set of unvisited customers)
(•)
(new customer added)
(not here)
(•)
4
choosing customer insertion positions
5
resulting routes

Fig. 1. A diagram illustrating the steps that comprise vehicle based decomposition

remaining vehicle routes. If there are customers which do not fit into any available vehicle route, they are added to a new vehicle, which means that vehicle reduction did not succeed in this scenario. Fitting customers into other vehicle routes is performed in a parallel. The insertion function considers fitting customers into every possible position in the current customer list, as long as the capacity and time constraints have been met. After customer fitting is finished, it is necessary to decide to which new vehicle (route) will each of the customers be assigned. The vehicle which has the least spare capacity is selected in the vehicle route selection process.

```
function VehicleReduction
    $order\ routes\ ascending\ by\ customers\ count$
    for route in 50% smallest routes do
        $currCustomers \leftarrow route.Customers$
        $remove\ route\ from\ routes$
        for $customer\ in\ currCustomers$ do
            $StartParallelInsertion(customer)$
            if $a\ customer\ is\ not\ inserted$ then
                $create\ newRoute$
                $add\ uninserted\ customers\ to\ newRoute$
                $add\ newRoute\ to\ Routes$
            end if
        end for
    end for
end function
function $StartParallelInsertion(customer)$
    $send\ customer\ to\ all\ routes$
    $start\ parallel\ insertion\ in\ each\ route$
    $wait\ for\ all\ routes\ to\ finish$
    $select\ which\ route\ will\ accept\ customer$
end function
```

The algorithm always starts from routes which have the smallest customer count, as it is generally easier to redirect fewer customers.

V. Evaluation

Experiments were conducted on Solomon's 56 benchmark problems [14] and Gehring and Homberger 300 benchmark problems [15]. The Solomon benchmark problems consist of 100 customers, while the Gehring and Homberger problems consist of 200, 400, 600, 800 and 1000 customers. Both of the benchmark sets are divided into 6 groups (C1, C2, R1, R2, RC1, RC2). Groups named Cx contain problems in which geographical data is clustered, those named Rx have randomly generated geographical data and RCx contain problems with a combination of both. Problems in groups x1 have a short scheduling horizon and allow only a small number of customers per route. In contrast, groups x2 have a long scheduling horizon and the number of customers per route is significantly bigger. For all problems, the travel time and distance is equal to the corresponding euclidian distance [16]. All problems have a central depot, as well as capacity and time window constraints.

Each of reported results is the average of 3 runs on an Intel Pentium Dual CPU T3200 (2 cores). The parameters used for function fun1 were set to x=0.8 , y=0.4 and z=0.1, determined experimentally.

Table I shows results obtained by our 2-VD (two-phase Vehicle based Decomposition) algorithm on all 356 test instances. The sum of the best known results is 10702 vehicles [17], this result is obtained by combining many different algorithms which provide the best know solutions for only part of the test instances. To the best of our knowledge there is no algorithm that was tested on all instances. The calculated best result obtained using the objective fun1 in our algorithm is only 5.51% worse than the sum of the best known solutions.

The results presented in the continuation of this section were all obtained using a greedy initial solution with the objective criterion OBJ=fun1.

Fig. 2 presents the influence of the number of customers on the algorithm execution time. For 60 test instances with 1000 customers, the algorithm runs 2 minutes and 14 seconds, which is approximately 2.23 seconds per problem instance.

Tables II and III report the execution time and vehicle count for the 2-VD algorithm for problems with 1000 customers for each specific test group. Results for group C2 are very poor both with respect to run time and vehicle count. Group R1, which is the most complementary to group C2, gives very good results. Evaluated instances with randomly generated geographical data and fewer customers per route provide better results than instances with clustered geographical data and a larger number of customers per route. The percentages in Table II gives a comparison of our results and best known results. For group combination Rx our results are only 1.71% worse than the best know solution combining several algorithms.

TABLE II
RESULTS BASED ON BENCHMARK PROBLEMS GROUP COMBINATIONS

Group	Vehicles Count best known	VC (fun1)	%	VC after VR	%
x1	2760	3008	8.99	2826	2.39
x2	661	773	16.94	731	10.59
Cx	1230	1318	7.16	1310	6.5
Rx	1109	1269	14.43	1128	1.71
RCx	1082	1194	10.35	1119	3.42

Figure 3 shows the number of cores influence on the vehicle reduction execution time. Vehicle reduction execution time is a major part of the whole algorithm execution time. Results are evaluated for problem instances with 1000 customers. As can be seen, parallel jobs scale well and multiplying cores provides the expected results of improving speed.

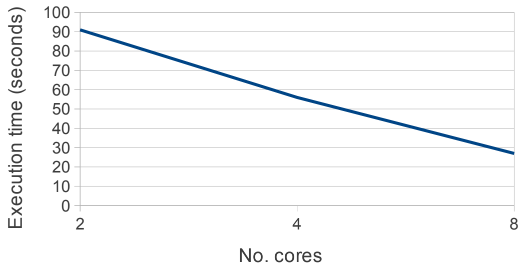


Fig. 3. Cores count and vehicle reduction execution time

VI. FUTURE WORK

Future work will primary focus on exploring additional customer-to-route insertion methods. According to the obtained results, using more cores gives additional time for insertion exploration. Some possible methods are the 2-opt heristic, ruin and recreate tehnique [10], CROSS-exchange operator and GENIUS-exchange operator [8].

One of our aims is to apply the proposed algorithm to the domain of cloud computing. Scheduling resources in cloud computing is a problem where constraints in a multiple-dimensioned solution space have to be satisfied (energy wastage minimization, service level agreement fullfilment). Given that the 2-VD algorithm has shown to be good for this problem with constraints in various solution space dimensions (time, space, capacity), it is our belief that it would be well suited for the domain of cloud computing as well. By using user resource demand forecasting (based on time series forecasting), we could obtain information regarding customer time windows and could look at physical machines as capacitated vehicles that need to visit these customers.

VII. CONCLUSION

A 2-phase Vehicle based Decomposition algorithm has been proposed. First, the initial solution is obtained using a customized greedy algorithm, followed by a vehicle reducing method. Vehicle reduction consist of removing vehicles with less customers and parallel insertion of customers from the removed vehicles to the available ones. Parallel insertion is implemented using multi-threading. After insertion, only one vehicle is selected for each customer. The obtained results (vehicle count) are within 5.51% of the sum of the best known results obtained by a diverse algorithm set. Experimental results show that parallel jobs scale well and the program's execution time decreases linearly with the increase in CPU cores.

ACKNOWLEDGEMENTS

The authors wish to express their gratitude to Ivan Fabek for his valuable contribution in implementing the result visualization program.

REFERENCES

[1] "EU ENERGY IN FIGURES 2010 - CO2 emissions by sector," 2010. [Online]. Available: http://ec.europa.eu/energy/publications/doc/statistics/ext_co2_emissions_by_sector.pdf
[2] T. Carić, "Improving of transport organization using heuristics methods," *(Croatian) Ph. D thesis, University of Zagreb, Croatia*, 2006.
[3] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, Jun. 1981. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/net.3230110211/abstract
[4] R. Bent and P. Hentenryck, "Spatial, temporal, and hybrid decompositions for large-scale vehicle routing with time windows," *Principles and Practice of Constraint Programming–CP 2010*, p. 99–113, 2011.
[5] ——, "A two-stage hybrid local search for the vehicle routing problem with time windows," *Transportation Science*, vol. 38, no. 4, p. 515–530, 2004.
[6] P. Toth and D. Vigo, *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics ans Applications, SIAM, Philadelphia, 2002.
[7] H. Li and A. Lim, "Local search with annealing-like restarts to solve the VRPTW," *European journal of operational research*, vol. 150, no. 1, p. 115–127, 2003.
[8] E. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J. Potvin, "A tabu search heuristic for the vehicle routing problem with soft time windows," *Transportation science*, vol. 31, no. 2, p. 170–186, 1997.

TABLE I
RESULTS FOR DIFFERENT GREEDY ALGORITHMS

GreedyType	Time (greedy)	Vehicles Count	Time (Vehicle Reduction)	VC after VR	Time (sum)
distance	1 min 44 sec	20 618	7 min 58 sec	12 692	9 min 42 sec
ready time	1 min 29 sec	14 328	7 min 20 sec	13 108	8 min 49 sec
due date	1 min 29 sec	13 772	5 min 54 sec	12 746	7 min 23 sec
visited time	1 min 44 sec	12 408	4 min 48 sec	11 710	6 min 32 sec
window time	1 min 31 sec	23 241	19 min 2sec	15 304	20 min 33 sec
fun1	2 min 14 sec	11 962	3 min 30 sec	11 261	5 min 44 sec

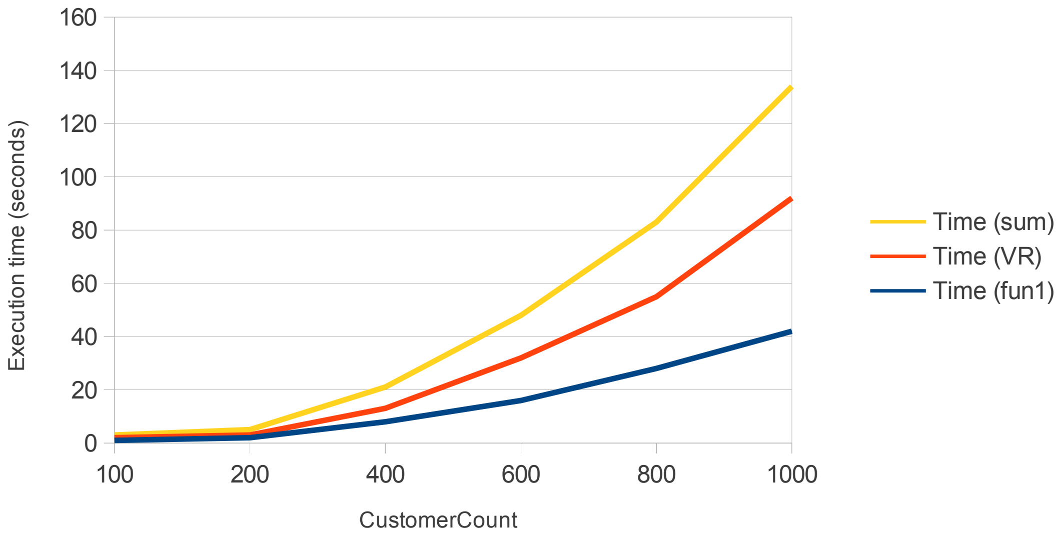


Fig. 2. Customer count and execution time

TABLE III
EXECUTION TIME AND VEHICLE COUNT PER BENCHMARK PROBLEM GROUPS

Group	Time (fun1)	Vehicles Count	Time (VR)	VC after VR	Time (sum)	VC sum of best known
C1	6 sec	987	10 sec	985	16 sec	941
C2	7 sec	331	36 sec	325	42 sec	289
R1	7 sec	1047	19 sec	930	26 sec	919
R2	8 sec	222	11 sec	198	19 sec	190
RC1	6 sec	974	8 sec	911	14 sec	900
RC2	8 sec	220	7 sec	208	15 sec	182

[9] L. Rousseau, M. Gendreau, and G. Pesant, “Using constraint-based operators to solve the vehicle routing problem with time windows,” *Journal of Heuristics*, vol. 8, no. 1, p. 43–58, 2002.

[10] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, and G. Dueck, “Record breaking optimization results using the ruin and recreate principle,” *Journal of Computational Physics*, vol. 159, no. 2, p. 139–171, 2000.

[11] D. Mester and O. Bräysy, “Active guided evolution strategies for large-scale vehicle routing problems with time windows,” *Computers & Operations Research*, vol. 32, no. 6, p. 1593–1614, 2005.

[12] L. Gambardella, E. Taillard, and G. Agazzi, “Macs-vrptw: A multiple colony system for vehicle routing problems with time windows,” in *New ideas in optimization*, 1999.

[13] J. Homberger, H. Gehring *et al.*, “Two evolutionary metaheuristics for the vehicle routing problem with time windows,” *Infor-Information Systems and Operational Research*, vol. 37, no. 3, p. 297–318, 1999.

[14] M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Operations research*, p. 254–265, 1987.

[15] “Extended SOLOMON’s VRPTW instances,” http://www.fernuni-hagen.de/WINF/touren/inhalte/probinst.htm. [Online]. Available: http://www.fernuni-hagen.de/WINF/touren/inhalte/probinst.htm

[16] M. Deza and E. Deza, *Encyclopedia of distances*. Springer Verlag, 2009.

[17] “VRPTW,” http://www.sintef.no/Projectweb/TOP/Problems/VRPTW/. [Online]. Available: http://www.sintef.no/Projectweb/TOP/Problems/VRPTW/