Column Generation with Bi-directional Dynamic Programming for Vehicle Routing and Scheduling Problem with Semi-Soft Time Windows

Narath BHUSIRI[1], Eiichi TANIGUCHI[2] and Ali G. QURESHI[3]

[1] Member of JSCE, Dept. of Urban Management, Kyoto University
(C1-2-334 Bldg., Nishikyo-ku, Kyoto-city 615-8540, Japan)
E-mail:bhusiri.narath@at3.ecs.kyoto-u.ac.jp
[2] Member of JSCE, Professor, Dept. of Urban Management, Kyoto University
(C1-2-335 Bldg., Nishikyo-ku, Kyoto-city 615-8540, Japan)
E-mail: taniguchi@kiban.kuciv.kyoto-u.ac.jp
[3] Member of JSCE, Lecturer, Dept. of Urban Management, Kyoto University
(C1-2-334 Bldg., Nishikyo-ku, Kyoto-city 615-8540, Japan)
E-mail: aligul@ums.mbox.media.kyoto-u.ac.jp

Vehicle Routing Problem with Semi-Soft Time Windows (VRPSSTW) is a new extension of Vehicle Routing Problem (VRP). In the VRPSSTW, vehicles are not strictly forced to serve customers only within time windows. It is also possible to arrive earlier than time windows (waiting at no cost until time windows begin) or to arrive later than time windows (with taking penalty costs into account). An exact column generation algorithm involving new elementary shortest path problem with resource constraints and late arrival penalties (ESPPRCLAP) as subproblem is used to solve the VRPSSTW to optimality. This paper compares the results of using mono-directional and bi-directional dynamic algorithms for solving the ESPPRCLAP.

Key Words : *column generation, semi-soft time windows, vehicle routing problem, exact algorithm*

1. INTRODUCTION

Vehicle Routing Problem (VRP), defined as NP-hard problem, is an effective route optimizing tool in city logistics[1),2)]. The VRP is required to determine a set of minimum cost routes for an identical set of vehicles to serve a set of known demand customers. Each route starts and ends at the central depot and visits a subset of customers along its route without vehicle capacity violation. Each customer must be served once.

Different variants of the VRP have been studied by researchers, for instance, VRP with hard time windows (VRPHTW)[3),4)] that strictly requires vehicles to visit at any customer location within a specified time interval (time windows or $[a_i, b_i]$: a_i and b_i represent the earliest and latest possible service start time at each customer, respectively). Another variant called VRP with soft time windows (VRPSTW)[5)] is considered by relaxing time windows. It is possible to visit customers beyond time windows $[a_i, b_i]$ with some early or late penalty costs.

Recently, a new variant of the VRP named vehicle routing problem with semi-soft time windows (VRPSSTW)[1)6)] has been extended from the VRPHTW and the VRPSTW. Vehicles are allowed to wait without any cost at customer locations until the earliest service start time (a_i) in case of early arrival or vehicles have to take linear penalty terms into account if vehicles arrive later than time windows (see Fig. 1). The latest possible service start time of each customer is extended from b_i to b_i'. The VRPSSTW takes advantages of offering more efficient use of vehicles (compared to the VRPHTW) and dealing with less complex cost structure (compared to the VRPSTW).

Both exact and heuristic approaches have been proposed to solve the VRPSSTW. Focusing on exact algorithm presented first by Qureshi et al.[6)], a column generation scheme based on a new subproblem, namely elementary shortest path problem with resource constraints and late arrival penalties (ESPPRCLAP), has been successfully developed.

However, the authors reported that only up to 50-customer instances of some problem types in Solomon's benchmark[4] have been exactly solved with reasonable computational time. This is because the complexity of subproblem increases when relaxing time windows[7].

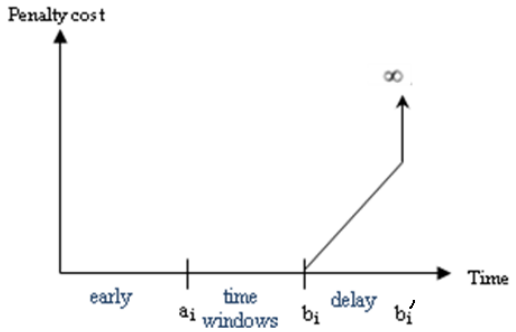

Fig.1 Linear late arrival penalty function for the VRPSSTW.

This paper aims to improve the ESPPRCLAP pricing subproblem in exact column generation to efficiently solve large problem instances in Solomon's benchmark set to optimality. Instead of using mono-directional dynamic programming in subproblem[1),6)], the bi-directional dynamic programming algorithm developed by Righini and Salani[8),9)] is applied in this paper for handling with the ESPPRCLAP subproblem.

2. COLUMN GENERATION

Column generation (or Dantzig-Wolfe decomposition or branch-and-price algorithm) decomposes the VRPSSTW problem into 1) a set covering master problem 2) the ESPPRCLAP as its subproblem. To calculate the extended latest allowable service start time at any customer (b_i', refer to Fig. 1) as well as the comprehensive VRPSSTW mathematical model can be found in Qureshi et al.[1),6)].

The set covering master problem aims at finding a set of least cost routes that services all customers. While the ESPPRCLAP subproblem generates the feasible routes subject to semi-soft time windows and capacity constraints. The set covering master problem can be stated as below;

$$\min \quad \sum_{p\in P} c_p y_p \tag{1}$$

$$\text{s.t.} \quad \sum_{p\in P} a_{ip} y_p \geq 1 \qquad \forall i \in C \tag{2}$$

$$y_p \in \{0,1\} \qquad \forall p \in P \tag{3}$$

where C denotes a set of customers, P is a set of feasible routes, c_p is cost of route $p \in P$, y_p is a binary variable having value 1 if route $p \in P$ is selected and 0 otherwise and a_{ip} is the number of times route $p \in P$ visits customer $i \in C$. The objective (1) is to minimize the overall cost of routes. The overall costs include fixed vehicle cost, traveling cost and late arrival penalty cost (if necessary).

The pricing subproblem consists of determining a set of feasible routes with negative reduced cost. The reduced cost of route is the sum of the reduced cost of all arcs in the route and the reduced cost of each arc can be computed using (4)

$$\bar{c}_{ij} = c_{ij} - (\pi_i / 2) - (\pi_j / 2) \qquad \forall i \in V \tag{4}$$

let V be the vertex set including depot (vertex 0) and customer set (with vertices 1,2,...,n), c_{ij} is the cost of arc *(i,j), i,j* $\in V$ and π_i is dual variables corresponding to (2) in master problem. Please be noted that dual variable at depot vertex is initially 0 (π_0=0).

(1) ESPPRCLAP subproblem

The general idea of bi-directional dynamic programming[8),9)] to solve the ESPPRCLAP subproblem is that labels are simultaneously generated from both forward and backward path directions subject to semi-soft time windows and capacity constraints. Forward path direction starts from initial depot and move forward to customer *i*. This represents the path from depot to *i*. While backward path direction starts from terminal depot and move backward to customer *j* representing path from *j* to depot.

Applying critical resource rule[8),9)] in order to reduce a number of labels generated, path extension in both directions are only extended when critical resource consumption does not exceed a half of the total available resource. In the ESPPRCLAP, critical resource is the time (*T*) resource, where *T* can be found as the maximum possible arrival time at terminal depot (T = min $\{b_0, (\max_{i \in C} \{b_i' + \Theta_i + tt_{i0}\})\}$) , let Θ_i be service time at $i \in C$ and tt_{i0} be traveled time from customer *i* to terminal depot).

Finally, partial paths from both directions are joined if satisfying all feasibility conditions. The steps of bi-directional dynamic programming algorithm for solving the ESPPRCLAP can be briefly outlined as follow;

a) Forward path extension

Forward time windows [a_i, b_i'] are associated with each vertex $i \in V$. Note that at depot vertex, b_0=b_0'. Three involved resources are consumed during path extension indicating by *r* (time), q_{ac} (demand) and *S*

(unreachable vertices). All resources are initially set to 0. When path is extended from i to j, these resources are updated as follow

$$\tau^* = \max\{\tau + \Theta_i + tt_{ij},\ a_j\} \quad (5)$$

$$q_{ac} = q_{ac} + q_i \quad (6)$$

Also to update S resource, some vertices are set to 1 if they have already been visited or they are becoming unreachable vertices due to violating constraints. Forward path extension (from i to j) is feasible only if $\tau \le b_j'$, $\tau \le T/2$, $a_j \le T/2$ and q_{ac} is not larger than vehicle capacity. Cost updating along forward path extension is done similar to Qureshi et al.[1),6)].

b) Backward path extension

Backward time windows $[A_i, B_i']$,calculated by adding forward time windows with service time, represent the range of time at each vertex $i \in C$ when service can be terminated. All resources (q_{ac}^{bw}, S^{bw}) except t^{bw} (time) are initialized and updated by the same rules as forward extension. The particular updated rule of t^{bw} when path extended from j to i is as follow;

$$t^{bw*} = \max\{t^{bw} + \Theta_j + tt_{ij},\ T - B_i'\} \quad (7)$$

Backward path extension from j to i is feasible only if $t^{bw} \le T - A_i$, $t^{bw} \le T/2$, $B_i' \le T/2$ and q_{ac} does not exceed vehicle capacity. Late arrival penalty cost corresponded to vertex j has to be considered in case of $T - B_j \le t^{bw} \le T - A_j$.

To avoid generating labels that cannot lead to the optimal solution, the same dominance rule developed by Feillet et al.[10)] is implemented on both forward and backward path extensions.

c) Joining of forward and backward path extensions

Forward and backward partial paths must be joined together to become complete paths. The feasibility conditions that have to be satisfied during the joining are: 1) no vertex can be visited by both forward and backward paths 2) $q_{ac} + q_{ac}^{bw}$ is no larger than vehicle capacity and 3) $\tau + t^{bw} \le T$.

3. RESULTS AND DISCUSSIONS

In this section, the exact solutions of the VRPSSTW using column generation algorithm with mono-directional dynamic programming (hereafter referred as case 1) and using column generation with bi-directional dynamic programming (hereafter referred as case 2) are summarized in Table 1. Due to arbitrary limits and in order to simplify the problems, b_i' in both cases are particularly relaxed to 10 minutes for every customer instead of using maximum b_i' based on formulation. All solutions were run on MATLAB (version R2009) and Solomon's benchmark[4)] R1 type of instances were selected to test the algorithms.

In Table 1, a few 25-customer, 50-customer and 75-customer instances of Solomon R1 type have been solved to optimality for both case 1 (represented by white rows) and case 2 (represented by gray rows). Column[2] represents the number of branch and bound nodes. Column[3] gives lower bound obtained at the root node. Column[4] shows the optimal objective value while column[5] gives number of vehicles needed. Column[6] shows the number of iterations when the optimal integer solution is reached. Column[7] gives the number of labels generated at root node. In case 2, both forward and backward labels are generated (forward labels /backward labels) while forward labels are only generated in case 1. Column[8] and [9] report computational time in seconds and the percentage of gap between[3] and [4] respectively. Note that R104_50* and R105_75* in case 1 are the best integer solutions obtained from more than 5 hours of computational time.

A comparison of both cases shows that case 1 requires less computational time when solving small size (25 and 50 customers) of R101, R102 and R105 instances. All these Solomon instances have narrow time windows and the complexity of problem depends on the width of time windows. Thus, the subproblems in case 1 can be rapidly and easily handled while the joining step in subproblem of case 2 needs slightly more computational time.

In contrast, R103 and R104 instances that have very wide time windows result in difficulty of handling subproblems. Using bi-directional dynamic algorithm (case 2) spends shorter computational time to generate feasible paths in subproblem especially for 50-customer of R103 and R104 instances. Case 2 also requires less computational time to solve large size (75 customers) of some R1 instances. In addition, the gap between lower bound obtained at the root node and the optimal solution in case 2 is smaller (or equal) than case 1. This implies that lower bound obtained in case 2 is better.

4. CONCLUSIONS

This paper attempted to handle the ESPPRCLAP subproblem in the VRPSSTW by using bi-directional dynamic programming. This paper also presented the

solutions tested on Solomon R1 instances of using column generation based bi-directional dynamic programming compared to using column generation based mono-directional dynamic programming.

Table 1 Summary of the VRPSSTW exact solutions

Instance [1]	BB [2]	LB [3]	Z [4]	K [5]	Run [6]	Label [7]	Time (s) [8]	GAP (%) [9]
R101_25	2	5554	5841	7	10	891/-	1.1	4.9
R101_25	2	5554	5841	7	6	105/96	1.4	4.9
R101_50	1	7958	7958	9	11	6693/-	6.8	0
R101_50	1	7958	7958	9	8	385/299	8.6	0
R101_75	29	11686	11931	14	176	30015/-	324.9	2.1
R101_75	8	11927	11931	14	34	843/713	253.3	0.03
R102_25	3	4561	4986	6	20	4737/-	5.9	8.5
R102_25	2	4561	4986	6	8	438/649	14.0	8.5
R102_50	15	6507	6939	8	88	40953/-	255.6	6.2
R102_50	16	6516	6939	8	37	4878/ 3051	281.6	6.1
R102_75	15	9406	9570	11	109	104866/-	1257.4	1.7
R102_75	14	9414	9570	11	45	8475/ 7058	1165.3	1.6
R103_25	1	3445	3445	4	17	17678/-	24.2	0
R103_25	1	3445	3445	4	7	1595/ 1532	22.1	0
R103_50	13	5508	6046	7	111	225008/-	3718.3	8.9
R103_50	13	5615	6046	7	44	18492/ 12656	382.3	7.1
R104_25	2	2997	3400	4	31	86375/-	285.8	11.9
R104_25	13	3036	3400	4	19	3601/ 3152	199.0	10.8
R104_50*	1	-	39775	50	12	308328/-	50279	-
R104_50	2	4639	5098	6	7	86618/ 54932	1840.6	9.0
R105_25	3	3886	4250	5	18	2992/-	3.7	8.6
R105_25	2	3886	4250	5	7	280/213	9.6	8.6
R105_50	41	6421	6917	8	156	18950/-	132.6	7.2
R105_50	23	6443	6917	8	45	1669/ 853	184.4	6.9
R105_75*	52	8731	10331	12	301	72018/-	1142.5	15.5
R105_75	38	8723	9582	11	92	3971/ 3529	369.1	9.0

The obtained results show that solving large size of instances as well as solving instances with very wide time windows by column generation with mono-directional one need more computational time requirement. This is because a number of labels generated in subproblem is dramatically high when size of instances increase. The solution approach to optimality also needs many iterations and very large size of branch and bound tree. Using column generation with bi-directional dynamic programming has significant advantages over these cases. It leads to the reduction of labels generated in subproblem and computational time.

5. FUTURE WORK

This paper studied a variant of route optimizing tool, named the VRPSSTW. The overall cost as well as the number of vehicles used is optimized (minimized). The order of customers along the route is also optimally done. However, only single depot is represented in the system. Considering to be more realistic, more than one depot will be concerned (multi-depot vehicle routing problem) and the number of depots and their locations will be optimized (location routing problem).

6. REFERENCE

1) Qureshi, A. G., Taniguchi, E. and Yamada, T. : Exact solution for vehicle routing problem with semi soft time windows and its application, *Procedia Social and Behavioral Sciences,* Vol. 2, pp. 5931-5943, 2010.
2) Taniguchi, E., Thompson, R. G., Yamada, T. and Duin, R. V. : *City logistics: Network modeling and intelligent transport systems,* Pergamon, Oxford, 2001.
3) Kohl, N., Desrosiers, J., Madsen, O. B. G., Solomon, M. M., Soumis, F. : 2-path cuts for the vehicle routing problem with time windows, *Transportation Science,* Vol. 33, pp. 101-115, 1999.
4) Solomon, M. M. : Algorithms for the vehicle routing and scheduling problems with time window constraints, *Operation Research,* Vol. 35, pp. 254-264, 1987.
5) Ioannou, G., Kritikos, M. and Prastacos, G. : A problem generator-solver heuristic for vehicle routing with soft time windows, Omega; *The International Journal of Management Science,* Vol. 31, pp. 41-53, 2003.
6) Qureshi, A. G., Taniguchi, E. and Yamada, T. : An exact solution approach for vehicle routing and scheduling problems with soft time windows, part E45, *Transportation Research,* pp.960-977, 2009.
7) Desrochers, M., Desrosiers, J. and Solomon M. M. : A new optimization algorithm for the vehicle routing problem with time windows, *Operation Research,* Vol. 40, pp. 342-354, 1992.
8) Righini, G. and Salani, M. : New dynamic programming algorithms for the resource constrained elementary shortest path problem, *Networks; An International Journal,* Vol. 51, pp. 155-170, 2008.
9) Righini, G. and Salani, M. : Symmetry helps: Bounded bi-directional dynamic programming for the elementary shorest path problem with resource constraints, *Discrete*

Optimization, Vol. 3, pp. 255-273, 2006.

10) Feillet, D., Dejax, P., Gendreau, M. and Gueguen, C. : An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems, *Networks,* pp. 216-229, 2004.