negative reduced cost route. When column generation achieves optimality at each node of the branch-and-bound tree, i.e. no negative reduced cost route can be found by the SP, z_{RMP} of the current RMP fundamentally becomes z_{LB} (Chabrier, 2006). If the current optimal z_{RMP} contains integer solution, it will be set as the new global upper bound (z_{GUB}) in case it is less than the previous one. After entire nodes of the branch-and-bound tree are searched, the current z_{GUB} represents the optimal solution.

4.1 A Two-Stage Pricing Subproblem Solution Technique

The feasibilities of routes generated in the SP rely on constraints (7)-(17). When generating such routes, some complexities might particularly be taken into consideration in the SP including 1) Penalty function is not a non-decreasing function, 2) Possible arrival time at each customer is restrained by definitions of maximum early arrival time and late arrival time, 3) Vehicle capacity along its route is not monotonous, and 4) Repetition of customers visited in the same route is not allowed (elementary requirement). Therefore, to tackle the SP, a two-stage solution technique is then developed. More specifically, a set of candidate feasible routes is generated in the first stage without considering optimal departure times from depot, which are left to be found in the second stage. An optimal departure time is the starting time of the vehicle to leave depot in which overall costs of the route are minimized. In the following, a two-stage solution technique for the novel SP is briefly explained.

4.1.1 First stage: generating a set of candidate routes

In the first stage, a labeling algorithm, which is an extended version of Dell'Amico *et al*. (2006), Fagerholt (2001), and Feillet *et al*. (2004), is employed to generate a set of candidate routes with negative reduced costs. The key feature in this stage is that violating customer's hard time windows is still possible within $[a'_i, b'_i]$, yet penalty is temporarily not considered. Indeed, a vehicle needs waiting at no cost if it arrives early and the delayed service can also occur without penalty. The reduced cost of the route is the sum of the reduced costs of all arcs traversed in the route. The reduced cost $\bar{c}_{ij}$ of arc $(i,j) \in A$ disregarding penalty is computed as follows:

$$\bar{c}_{ij} = c_{ij} - \mu_i \qquad \forall i \in V \tag{21}$$

In the process, a partial path m is characterized by a label L_i^m, which contains useful components such as $last(L_i^m)$: current visited vertex $i \in V$, $\bar{c}(L_i^m)$: the reduced cost of the path m starting from depot, $dep(L_i^m)$: departure time from depot, $pre(L_i^m)$: predecessor vertex, $no(L_i^m)$: label number, $\tau(L_i^m)$: time resource, $q(L_i^m)$: load resource, $diff(L_i^m)$: accumulated load-difference resource, $wt(L_i^m)$: accumulated waiting time, $dt(L_i^m)$: accumulated delayed time, and $v_h(L_i^m), \forall h \in V$: unreachable resources considering at vertex $i \in V$. Unreachable resources, which consist of $|V|$ extra resources, help to forbid repetition of vertex visited in the path to satisfy elementary requirement. Each unreachable resource can be defined either reachable (taking value zero) or temporarily unreachable (0.5) or unreachable (one). The definitions of reachable, temporarily unreachable, and unreachable resources will be given later.

At initialization, the partial path m is constructed at depot, and it is labeled by L_0^m. As the

optimal departure time from depot will be determined in the second stage, an "as early as possible" departure time is instead considered for the path to particularly ensure that any possible departure time will not be neglected. Components $dep(L_0^m)$ and $\tau(L_0^m)$ are thus set to the earliest possible departure time, i.e. max(a_0, a_r-t_{0r}), where $r \in C$ is the prospective first customer of the path m. Please be noted that waiting at first customer of the path is never allowed. Other components including $\bar{c}(L_0^m)$, $q(L_0^m)$, $diff(L_0^m)$, $wt(L_0^m)$, and $dt(L_0^m)$ are all set to zero. The unreachable resource $v_0(L_0^m)$ is initiated as one due to already visited; the remaining unreachable resources $v_h(L_0^m), \forall h \in C$ can take values either zero or 0.5 or one depending on feasibility conditions, which will be described later.

Labeling algorithm is executed by extending partial paths from their current vertices to all possible successors. Note that the terminating depot (artificial vertex), which all partial paths must return to in order to complete their extensions (i.e. a complete path with negative reduced cost represents a candidate route $p \in P$), is always a possible successor for any vertex on any partial path at any time. The feasibility criteria of path extension in the labeling algorithm must assure that some possible schedules are not excluded as it could significantly impact on an absence of the optimal solution. Given the partial path m, its extension from current vertex $i \in V$ to successor vertex $j \in C$ is considered feasible if all equations (22)-(25) are satisfied.

$$b_i' + t_{ij} \geq a_j' \tag{22}$$

$$\tau(L_i^m) + t_{ij} \leq b_j' \tag{23}$$

$$q(L_i^m) + d_j + \max\{0, (u_j - d_j) + diff(L_i^m)\} \leq Q \tag{24}$$

$$v_j(L_i^m) < 1 \tag{25}$$

Focusing on equation (22), it is particularly developed instead of usual inequality $\tau(L_i^m) + t_{ij} \geq a_j'$. This is to retain all possible schedules at customers for further processing in the seconds stage as explained. Since all paths start from the depot at the "as early as possible" departure times, hence time resources at all vertices along the paths are the least ones, which are not necessarily the optimal ones owing to early arrival penalties. Due to the definition of maximum possible arrival time at vertex $i \in C$ within $[a'_i, b'_i]$ exists, use of inequality $\tau(L_i^m) + t_{ij} \geq a_j'$ might lead to an absence of some significant extensions. Instead, use of equation (22) attempts to keep all possible schedules for further processing in the second stage and to prevent the absence of the optimal solution. Once again, the optimal departure time from depot would be found in the second stage.

Equation (24) is also a particular criterion for the VRPSTWSPD subproblem. The vehicle capacity along its route is becoming non-monotonous as both pickup and delivery demands are simultaneously served at each visit, and such fluctuated capacity also plays a significant role in path extensions. For better understanding on this issue, an illustrative example is shown in Figure 2. Given that a problem considered consists of one depot, two customers with known demands, and a single vehicle with fixed capacity of 100. The resulting routing plans of this problem can be displayed in two cases. Both cases seem to be feasible but case 1 is, in fact, not a feasible plan. Once the vehicle leaves depot, total loads of the vehicle equal 100, and visiting customer no. 1 prior to customer no. 2 results definitely in an excess vehicle capacity. Use of equation (24) attempts to prevent such path extension of case 1.

Case 1

Case 2

Figure 2. Examples of routing plan

Every time the partial path m is successfully extended from current vertex $i \in V$ to its successor vertex $j \in C$, the associated label is changed to L_j^m. Components of L_j^m must also be updated as follows:

$$\tau(L_j^m) = \max\{\tau(L_i^m) + t_{ij}, a_j\} \tag{26}$$

$$\bar{c}(L_j^m) = \bar{c}(L_i^m) + \bar{c}_{ij} \tag{27}$$

$$wt(L_j^m) = wt(L_i^m) + \max\{0, a_j - (\tau(L_i^m) + t_{ij})\} \tag{28}$$

$$dt(L_j^m) = dt(L_i^m) + \max\{0, (\tau(L_i^m) + t_{ij}) - b_j\} \tag{29}$$

$$q(L_j^m) = q(L_i^m) + d_j + \max\{0, diff(L_i^m) + (u_j - d_j)\} \tag{30}$$

$$diff(L_j^m) = \min\{0, diff(L_i^m) + (u_j - d_j)\} \tag{31}$$

At some vertex $j \in C$, if factor $diff(L_i^m) + (u_j - d_j)$ based on equation (30) gives positive value, i.e. due to accumulated pickup demands are currently more than accumulated delivery demands, extra loads must be taken into account in load resource $q(L_j^m)$. That is to say, the vehicle needs to leave some spaces for such extra loads. After that, the accumulated load-difference resource of vertex $j \in C$, $diff(L_j^m)$, is set to zero again before further processing as in equation (31).

To update unreachable resources at vertex $j \in C$, $v_j(L_j^m)$ becomes one due to already visited. Some other vertex $h \in C$ would be unreachable ($v_h(L_j^m)$=1) if either it has already been visited or its extension from $j \in C$ to $h \in C$ does not satisfy equations (22)-(24). In particular, if extension from $j \in C$ to $h \in C$ does not just satisfy only equation (22), it is then defined as temporarily unreachable ($v_h(L_j^m)$=0.5). As time resource is a non-decreasing resource, it might be possible to include vertex $h \in C$ to the partial path m afterwards. In all other cases, vertex $h \in C$ would definitely be reached from vertex $j \in C$ ($v_h(L_j^m)$=0).

When two or more partial paths lead to the same current visited vertex, they must be compared and evaluated. All dominated partial paths are eliminated and no longer considered for extensions. This strategy is very useful to reduce storage space and computational time of labeling algorithm. The dominance criteria can be summarized as follows. Given that two

partial paths m and n end at vertex $i \in C$, labels associated to these partial paths are L_i^m and L_i^n, respectively. The partial path m dominates the partial path n if and only if $\bar{c}(L_i^m) \le \bar{c}(L_i^n)$, $\tau(L_i^m) \le \tau(L_i^n)$, $q(L_i^m) \le q(L_i^n)$, $wt(L_i^m) \le wt(L_i^n)$, $dt(L_i^m) \le dt(L_i^n)$, $v_h(L_i^m) \le v_h(L_i^n), \forall h \in V$, and at least one of these inequalities is strict. As the reduced cost calculation in this stage is not an actual calculation, i.e. all penalties are disregarded, two additional inequalities, i.e. $wt(L_i^m) \le wt(L_i^n)$ and $dt(L_i^m) \le dt(L_i^n)$, are then requisite in the dominance rules to precisely evaluate the partial paths.

4.1.2 Second stage: finding the optimal departure times of all candidate routes

Each candidate route obtained in the first stage must be processed to find an optimal departure time of the route. In this stage, all penalties concerned are now active. Given a candidate route $p \in P$ and the first customer $r \in C$ of the route, all possible departure times in range $[(\max(a_0, a_r - t_{0r})), (b'_r - t_{0r})]$ must be taken into consideration. As time windows and travel times are both integer, size of possible departure times that need to be considered is $|(b'_r - t_{0r}) - (\max(a_0, a_r - t_{0r}))|$.

The route p begins at depot; setting initial routing cost to zero, and time resource to the earliest departure time, i.e. $\max(a_0, a_r - t_{0r})$. Next, the vehicle leaves the depot to visit a given set of customers (all sequences were already fixed from the previous stage). Every time the vehicle traverses between two points, cost and time resource of the route p must be updated. The updates of cost and time resource follow equation (3) and equation (26), respectively. After serving all customers, the vehicle returns to the terminal depot. Then, the procedure starts again by setting initial cost of the route p to zero, and time resource to the next possible departure time. All corresponding departure times are proceeded in increasing order. If some possible departure time cannot satisfy soft time window at some vertex $i \in C$ due to arriving before a'_i, this departure time and its calculations will then be eliminated, and the procedure will start again with the next possible departure time. On the contrary, if some possible departure time results in arriving later than b'_i at some vertex $i \in C$, the procedure on the route p will immediately terminate, and all remaining departure times will not be considered. After all, the departure time that leads to cost minimization is selected. If two or more departure times lead to the same cost, the one with minimum waiting time is taken as one of our objectives. Eventually, all candidate routes must be optimally processed to determine their departure times and their corresponding overall costs before feeding them to the RMP.

5. GENETIC ALGORITHM

Genetic algorithm (GA), which is one of effective metaheuristics, has been widely applied to combinatorial optimization problems such as the VRP. Interesting reader is referred to Braysy and Gendreau (2005) and Caserta and Vob (2010) for comprehensive survey of metaheuristics. The GA works for a number of generations (iterations) in which characteristics of individuals (or solutions) in the current population are formed and placed to the new population via several genetic operators.

At each generation of the proposed GA, a two-stage operation is performed. In the first stage, all routes are primarily generated without considering optimal departure times from depot, i.e. considering to depart as early as possible. Furthermore, violating customer's hard time windows within $[a'_i, b'_i]$ is feasible without considering penalty. Later, in the second stage, after executing all genetic operators, each individual in the current population is then decoded

and each route in the individual is optimally processed to determine its departure time from depot using the same procedure as in the second stage of the pricing subproblem algorithm. Subsequently, the corresponding overall costs of all routes in each individual including fixed vehicle costs, travel costs, and penalties are recalculated, and an inverse of such overall costs provides a fitness value of the individual. It is worth pointing out that the two-stage operation in the proposed GA is implicitly similar to the two-stage subproblem algorithm of the branch-and-price approach, where a set of candidate routes is generated in metaheuristic fashion instead of exact fashion.

After termination of the proposed GA, the best individual found during entire search is finally taken, representing the (nearly) optimal solution. The proposed GA and its operators can be summarized as follows.

5.1 Individual Representation

An individual in the population is encoded by a two-row representation, where the first row represents an integer string (called chromosome) of length $|C|$, and the second row indicates a set of assigned vehicles. As displayed in Figure 3, an illustrative instance with 5 customers is routed. Each integer (called gene) in the chromosome represents a customer node; depot is always omitted. In this manner, after leaving depot, vehicle 1, respectively, visits customer 5 and 1 before returning to depot. Vehicle 2 visits depot, customer 2, 3, and depot in sequence, while vehicle 3 is a single customer route of customer 4.

Chromosome	5	1	2	3	4
Vehicle reference	1	1	2	2	3

Figure 3. Solution representation

5.2 Construction Algorithm

To start the proposed GA, an initial population of D individuals must be created (D is fixed to 100 individuals in this study). First half of the initial population is obtained using an extended insertion heuristic with residual capacity criterion (ψ_{RC}) of Dethloff (2001), where the algorithm must incorporate additional soft time window constraints. While the second half is constructed from an effective greedy algorithm with reinsertion of single customer routes. Every time a customer is added to (or removed from) the existing route, feasibilities on capacity constraints and soft time window constraints must be checked. The extended insertion-based heuristic generally gives better quality of individuals, while the greedy algorithm results in more diversity.

In particular, for more advanced exploration of the solution space, the proposed GA considers a new search strategy over all generations. Such strategy is to allow generating intermediate infeasible routes with taking value M into account. An intermediate infeasible route is a route that violates soft time window constraint at some vertex $i \in C$ due to arriving earlier than a'_i. Given that a vehicle $k \in K$ leaves vertex $h \in V$ to vertex $i \in C$, which causes intermediate infeasible arrival, i.e. $a'_i - p_{ik} > 0$, the travel cost involved is calculated by $c_{hi} + M(a'_i - p_{ik})$.

5.3 Selection Operator

In this operator, individuals in the current population are selected to be parent (father and

mother) individuals. A tournament selection (Ghoseiri and Ghannadpour, 2010) is employed for this purpose. Two copies of the population are made. In the first copy, all individuals are ranked in order of decreasing their fitness values. Two consecutive individuals are compared and the one with higher fitness value is chosen to be father individual. In the second copy, all individuals are arbitrarily ranked and processed similarly. Finally, a set of father individuals with size $|D/2|$ and a set of mother individuals with size $|D/2|$ are ready for mating.

5.4 Crossover Operator (recombination, with a rate of 80%)

Characteristics of father and mother chromosomes are shared and carried on into offspring individuals during crossover. The best cost route crossover (BCRC) of Ombuki *et al*. (2006) is called to create two offspring from a pair of parents. Given that a father individual (F) and a mother individual (M) are selected for mating. To create the first offspring, one route in F is randomly chosen, all customers appeared in the selected route of F are then removed from M. Next, all removed customers in M are, one-by-one randomly, reinserted to the existing routes of M in their best possible locations. Note that some removed customer might be reinserted to a new route if possible insertion to all existing routes of M cannot be found. To create the second offspring, a route in M is in turn randomly selected and the same procedure is called. Finally, a set of D offspring individuals is thus obtained after crossover and they will become candidates for the new population.

5.5 Mutation Operator (with a rate of 20%)

Mutation operator attempts to prevent too early convergence of the solution by enlarging the search region. Two mutation strategies of Alvarenga *et al*. (2007) are applied to a randomly selected subset of offspring individuals. Both mutation strategies are equally likely to be applied. The first strategy is a similar customer exchange. In each selected offspring individual, the operator randomly selects a route and a vertex $i \in C$ associated to the route. Then, the operator searches for a vertex $j \in C$ located in other routes in which the difference of these two customers, i.e. $|a_i - a_j|$, is minimal, and tries to exchange their locations. The second strategy is a random customer migration, where a randomly chosen vertex $i \in C$ from one route is randomly migrated to other existing route.

5.6 Intensification (with a rate of 30%)

After mutation, a fixed number of offspring individuals is randomly selected. A 2-opt* local search of Potvin and Rousseau (1995) is then applied to improve the quality of these offspring.

5.7 Fitness Value

The fitness value must be measured at the end of every generation in order to evaluate the performance of all individuals generated in the current population, which further influences creation of the next population. The evolution of the GA relies on these fitness values. The fitness value of each individual equals the inverse of overall costs of all corresponding routes in the individual as described previously.

5.8 Replacement and Elitism

A set of *D* offspring generated through genetic operators explained above is eventually set to be the individuals of the new population. The generations of GA are continually repeated until terminating conditions are met. The proposed GA terminates when either 200 iterations are run or 20 consecutive populations result in similar individuals. To preserve the quality of solution over generations, 2% of the best individuals (highest fitness values) found in the current population are kept and brought directly to the new population.

6. RESULTS AND DISCUSSIONS

Our numerical experiments were conducted on an Intel Core i7 CPU, 2.67 GHz processor, and 4 GB of memory, and all codes were implemented in MATLAB (R2009a). Test instances for evaluating the proposed solution algorithms are composed of Solomon's VRPTW benchmark set (Solomon, 1987). For evaluating the proposed GA, 10 experiments were performed on each instance and the one with best solution produced was selected.

In Solomon's benchmark, six different types of problem are contained; however, only the RC1 type was picked for testing. Generally, the RC1 type consists of eight different instances. All information including coordinates of vertices, vehicle capacity, service time, time window of depot, and customer's hard time windows are provided in each instance. A single depot is located at the center of a two-dimensional 100*100 Euclidean space, while customer's locations are geographically dispersed in a form of mixed cluster and randomness. All instances of the RC1 type have the same coordinates of vertices, whereas they differ from each other in characteristics of customer's hard time windows. Figure 4 displays an instance of the RC1 type, named RC101.

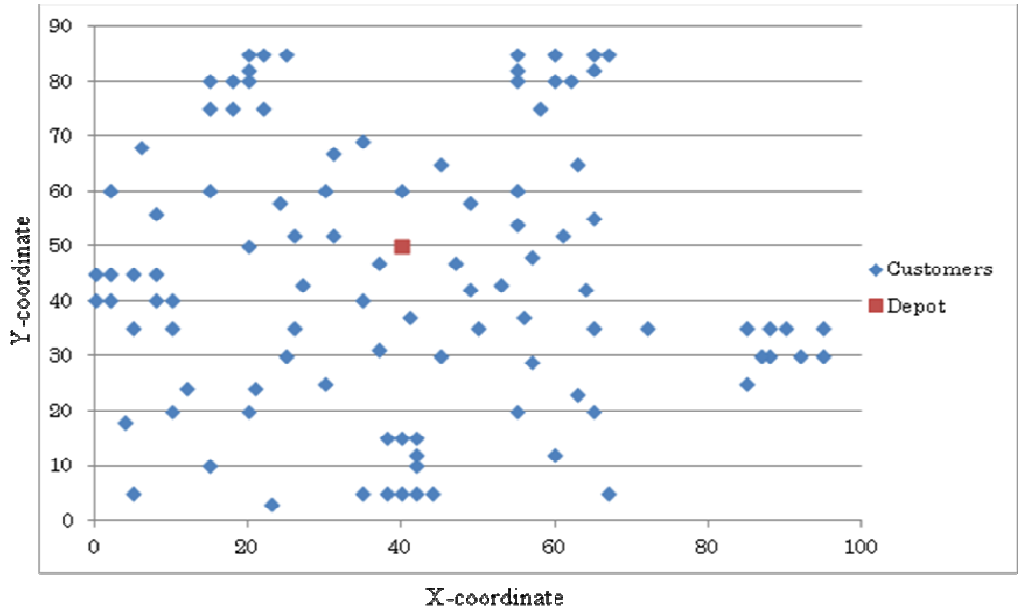

Figure 4. Scattered points of RC1 type (RC101 instance)

In particular, delivery demands and pickup demands of all customers were randomly generated between the interval [0,90]. Note that each complete instance comprises 100 customers, yet smaller sizes of the concerned instance could be considered by taking the first few customers such as 25, 40 or 75 customers. Travel cost on arc connected two points was measured using Euclidian distance calculation and was rounded down to nearest integer if

fractional. Travel time on arc was set equal to the sum of travel cost on the arc and service time at the starting point. Following data reported in Qureshi *et al*. (2009), vehicle operating cost (VOC) was set to 14.02 yen/minute and fixed vehicle cost was taken as 10417.5 yen/vehicle. Both values were gathered from surveys of Japanese logistics firms. For the sake of simple calculations, however, the VOC was converted to as one unit of travel cost equals one unit of travel time (i.e. VOC=1). Fixed vehicle cost was consequently scaled to the same level, i.e. 743 unit cost/vehicle. Early arrival penalty and late arrival penalty were set to three times and five times of the VOC, respectively. Due to arbitrary limits and simplicity of problems, a soft time window at vertex $i \in C$, i.e. $[a'_i,b'_i]$ was particularly extended up to 10 units from the given hard time window $[a_i,b_i]$ instead of using maximum a'_i and b'_i based on Eq.(1)-(2).

Table 1 and 2 provide numerical results of the branch-and-price algorithm and the proposed GA, respectively, for solving benchmark instances. In Table 1, the most-left columns show instance name with size of customers considered. Column "BB" represents number of nodes explored in the search tree. Column "LB" gives lower bound value obtained when column generation ended at the root node. Column "Z" is the optimal cost, while columns "WC" and "DC" specify, respectively, early arrival penalties and late arrival penalties. Column "K" gives the number of vehicles needed in the optimal solution. Column "Run" shows total number of iterations required to reach the optimal solution. Column "Label" shows an average number of labels generated at the root node. Column "CPU" reports computational time (in second) required starting from the root node to reach the optimal solution.

Table 1. Summary of exact solutions using the branch-and-price approach

Instance		BB	LB	Z	WC	DC	K	Run	Label	CPU
Name	Size									(s)
RC101	25	12	3360.4	3505	0	20	4	85	4279	47.2
RC102	25	11	3740.2	4213	0	0	5	101	16779	108.7
RC103	25	15	3609.2	4208	0	0	5	154	45261	334.3
RC104	25	14	2974.8	3379[a]	0	0	4	240	141801	1279.7
RC105	25	8	3947.1	4261	0	0	5	73	9589	69.2
RC106	25	15	4123.9	4258	0	5	5	88	13056	60.8
RC107	25	21	3849.2	4203[a]	0	0	5	106	41122	198.2
RC108	25	25	4375.6	4998[a]	0	0	6	145	37320	225.6
RC101	40	15	5893.8	6155	0	40	7	109	6462	129.5
RC102	40	17	5785.4	6056[a]	0	0	7	190	27760	664.4
RC103	40	15	5557.5	6045[a]	0	0	7	166	79128	1327.1
RC104	40	28	4727.9	5266[a]	0	5	6	366	299453	3962.4
RC105	40	11	6245.0	6847	6	0	8	121	14460	273.2
RC106	40	20	6479.4	6824[a]	0	5	8	146	18377	294.1
RC107	40	23	6312.4	6768[a]	0	0	8	124	54263	351.4
RC108	40	20	6363.9	6759[a]	0	0	8	163	112220	848.5

[a] The best integer solution found after 10 hours running of the branch-and-price approach.

The separated details of construction algorithm and the proposed GA are given in Table 2. Column "Total CPU" combines both CPU of construction algorithm and the GA. Column "Δt" describes the percentage decrease in computational time between exact solution and approximate solution, while "Δz" shows percentage gap between the optimal cost and the

best cost found by the proposed GA. The terminology "N/A" informs that a comparison of exact solution and approximate solution is not available as size of instance is too large for the branch-and-price approach, thus neither best integer solution nor lower bound obtained at the root node is found.

Table 2. Summary of approximate solutions using the proposed GA

Instance		Construction algorithm			GA			Total CPU	Δt	Δz
Name	Size	Z	K	CPU(s)	Z	K	CPU(s)	(s)	(%)	(%)
RC101	25	4370	5	4.6	3505	4	32.7	37.3	-21.0	0
RC102	25	4273	5	4.3	4213	5	30.5	34.8	-68.0	0
RC103	25	4291	5	5.2	4208	5	50.5	55.7	-83.3	0
RC104	25	3409	4	6.9	3379	4	73.3	80.2	-93.7	0
RC105	25	4689	5	4.7	4264	5	56.8	61.5	-11.1	+0.1
RC106	25	4997	6	4.5	4258	5	48.0	52.5	-13.7	0
RC107	25	4267	5	4.3	4203	5	46.5	50.8	-74.4	0
RC108	25	5027	6	4.3	5000	6	45.4	49.7	-78.0	+0.04
RC101	40	7162	8	10.5	6177	7	116.6	127.1	-1.9	+0.4
RC102	40	7049	8	9.7	6057	7	86.1	95.8	-85.6	+0.02
RC103	40	6930	8	12.4	6029	7	189.3	201.7	-84.8	-0.3[b]
RC104	40	5235	6	13.3	5171	6	138.8	152.1	-96.2	-1.8[b]
RC105	40	7922	9	12.6	6870	8	136.7	149.3	-45.4	+0.3
RC106	40	7152	8	9.3	6817	8	125.6	134.9	-54.1	-0.1[b]
RC107	40	6865	8	9.0	6752	8	148.4	157.4	-55.2	-0.2[b]
RC108	40	6886	8	11.4	6757	8	108.3	119.7	-85.9	-0.03[b]
RC101	75	13437	15	30.3	12120	14	276.2	306.5	N/A[c]	N/A[c]
RC102	75	12460	14	29.2	11242	13	231.2	260.4	N/A[c]	N/A[c]
RC103	75	12302	14	27.4	11045	13	369.8	397.2	N/A[c]	N/A[c]
RC104	75	10406	12	33.7	10109	12	805.8	839.5	N/A[c]	N/A[c]
RC105	75	13302	15	35.2	12006	14	514.1	549.3	N/A[c]	N/A[c]
RC106	75	13032	15	31.4	12106	14	326.9	358.3	N/A[c]	N/A[c]
RC107	75	12870	15	26.7	11782	14	596.4	623.1	N/A[c]	N/A[c]
RC108	75	11116	13	33.8	10157	12	857.3	891.1	N/A[c]	N/A[c]

[b] The percentage gap gives negative value, i.e. the solution cost of the GA is less than that of the branch-and-price approach since the branch-and-price approach fails to solve the instance to optimality within the setting time limit. The solution reported is just the best solution found after 10 hours of computational time.

[c] No comparison between exact solution and approximate solution is available as neither the best integer solution nor lower bound value could be produced by the branch-and-price approach.

Both the branch-and-price approach and the proposed GA were effective when performing on instances with 25-customer. The branch-and-price approach was able to solve almost all of the small-sized instances to optimality at reasonable computational time. For RC104, RC107, and RC108 with 25-customer, which could not optimally be solved, the solutions reported are just the best integer solutions found after 10 hours of computational time, i.e. not the optimal ones. In comparison to the exact solutions, the proposed GA could produce the solutions with the same number of vehicles used, while the solution costs obtained by the proposed GA of some instances were slightly high such as in RC105 (0.1%),

and RC108 (0.04%). Use of the proposed GA resulted in the decrease of computational time for all of the small-sized instances.

When involving instances with larger size and/or more complex, the branch-and-price approach often failed to solve them to optimality within the setting time limit of 10 hours of computational time. Due to the facts that a huge number of labels are generated in the SP, more column generation iterations are required, and a large number of nodes in the branch-and-bound tree are to be explored, the exact approach typically spent much longer computational time. Only RC101 and RC105 with 40-customer instances were solved to optimality. Both algorithms still gave the same number of vehicles used. Note that for some instances with 40-customer such as RC103, RC104, RC106, RC107, and RC108, even the best integer solutions produced by the branch-and-price approach within 10 hours of computational time were worse than those obtained by the proposed GA due to the complexity of problems.

For all instances with 75-customer, neither optimality nor lower bound nor the best integer solution obtained at the root node of the search tree could be acquired by use of the branch-and-price approach. Therefore, the results produced by the proposed GA are just given in Table 2 without any comparison. However, based on the performance of the proposed GA on the small-sized and medium-sized instances, it is expected that these results performed on larger-sized instances are also close to the optimal ones.

7. CONCLUSIONS

This paper has introduced the VRPSTWSPD, which represents a class of distribution and routing systems of convenience store industry. The VRPSTWSPD theoretically combines characteristics of two existing problems in the VRP family, i.e. the VRPSTW and the VRPSPD. The 3-index model formulation of the VRPSTWSPD has been first developed in this paper, taking into account several relevant constraints such as soft time window constraints and capacity constraints. Branch-and-price exact-based approach has been proposed to optimally solve the VRPSTWSPD. Decomposition of the VRPSTWSPD has led to the MP and the newly developed SP. Complexities of the SP rely on all of the following: 1) non-decreasing functions of cost and penalty, 2) non-monotonic vehicle capacity and 3) elementary requirement. To cope with the SP, the two-stage subproblem solution algorithm has also been proposed to generate a set of candidate routes in exact environment.

As NP-hard the VRPSTWSPD is, an efficient GA metaheuristic-based approach has been developed. The proposed GA considers an advanced search strategy to expand the exploration of solution space. Such strategy allows generating intermediate infeasible solutions with high penalties. Various genetic operators have been presented in order to enhance the performance of the proposed GA. The exact approach is superior to the metaheuristic approach due to the fact that only the former can give lower bounds. On the other hand, the GA has advantage over the exact approach when tackling large-scale network, where its usage would absolutely provide good solution to the problem in reasonable computational time. Comparisons of these two solution algorithms on small-sized benchmark instances signify that the proposed GA is remarkably efficient. The results indicate that the proposed GA worked well with minor tolerance. It is worth mentioning that the exact approach is extremely important although it is quite respected only to small-sized instances. Use of the exact solution as reference to assess performance of the proposed GA offers the high confidence level to apply the proposed GA to handle real-life instances, which are considerably larger and more complex.

REFERENCES

Ai, T.J., Kachitvichyanukul, V. (2009) A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36, 1693-1702.

Alvarenga, G.B., Mateus, G.R., de Tomi, G. (2007) A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers & Operations Research*, 34, 1561-1584.

Ando, N., Taniguchi, E. (2006) Travel time reliability in vehicle routing and scheduling with time windows. *Networks and Spatial Economics*, 6, 293-311.

Baldacci, R., Mingozzi, A., Roberti, R. (2012) Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research*, 218, 1-6.

Balakrishnan, N. (1993) Simple heuristics for the vehicle routing problem with soft time windows. *Journal of the Operational Research Society*, 44(3), 279-287.

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

Braysy, O., Gendreau, M. (2005) Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science*, 39(1), 119-139.

Caserta, M., Vob, S. (2010) Metaheuristics: intelligent problem solving. In Maniezzo, V., Stutzle, T., Vob, S. (eds.), *Matheuristics, Annals of Information Systems*, 10, 1-38. Springer US, ISBN 978-1-4419-1305-0.

Chabrier, A. (2006) Vehicle routing problem with elementary shortest path based column generation. *Computers & Operations Research*, 33, 2972-2990.

Chiang, W.C., Russell, R.A. (2004) A metaheuristic for the vehicle-routing problem with soft time windows. *Journal of the Operational Research Society*, 55, 1298-1310.

Dell' Amico, M., Righini, G., Salani, M. (2006) A Branch-and-Price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2), 235-247.

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

Fagerholt, K. (2001) Ship scheduling with soft time windows: An optimisation based approach. *European Journal of Operational Research*, 131, 559-571.

Feillet, D., Dejax, P., Gendreau, M., Gueguen, C. (2004) An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, 44, 216-229.

Figliozzi, M.A. (2010) An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C*, 18, 668-679.

Fu, Z., Eglese, R., Li, L.Y.O. (2008) A unified tabu search algorithm for vehicle routing problems with soft time windows. *Journal of the Operational Research Society*, 59, 663-673.

Gajpal, Y., Abad, P. (2009) An ant colony system for vehicle routing problem with simultaneous delivery and pickup. *Computers & Operations Research*, 36, 3215-3223.

Gendreau, M., Guertin, F., Potvin, J.Y., Taillard, E. (1999) Parallel tabu search for a real-time vehicle and dispatching. *Transportation Science*, 33(4), 381-390.

Ghoseiri, K., Ghannadpour, S.F. (2010) Multi-objective vehicle routing problem with time

windows using goal programming and genetic algorithm. *Applied Soft Computing*, 10, 1096-1107.

Hashimoto, H., Ibaraki, T., Imahori, S., Yagiura, M. (2006) The vehicle routing problem with flexible time windows and travelling times. *Discrete Applied Mathematics*, 154, 2271-2290.

Ibaraki, T., Imahori, S., Kubo, M., Masuda, T., Uno, T., Yagiura, M. (2005) Effective local search algorithms for routing and scheduling problems with general time-window constraints. *Transportation Science*, 39(2), 206-232.

Ioachim, I., Gelinas, S., Soumis, M., Desrosiers, J. (1998) A dynamic programming algorithm for the shortest path problem with time windows and linear node costs. *Networks*, 31(3), 193-204.

Ioannou, G., Kritikos, M., Prastacos, G. (2003) A problem generator-solver heuristic for vehicle routing with soft time windows. *Omega; The International Journal of Management Science*, 31, 41-53.

Laporte, G. (2009) Fifty years of vehicle routing. *Transportation Science*, 43(4), 408-416.

Liberatore, F., Righini, G., Salani, M. (2011) A column generation algorithm for the vehicle routing problem with soft time windows. *4OR: A Quarterly Journal of Operations Research*, 9(1), 49-82.

Lubbecke, M.E., Desrosiers, J. (2005) Selected topics in column generation. *Operations Research*, 53(6), 1007-1023.

Min, H. (1989) The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A*, 23(5), 377-386.

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

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

Ngaochay, T., Walsh, J. (2011) Paths to success for 7-Eleven in Thailand. *Information Management and Business Review*, 3(1), 1-7.

Ombuki, B., Ross, B.J., Hanshar, F. (2006) Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, 24, 17-30.

Parragh, S.N., Doerner, K.F., Hartl, R.F. (2008) A survey on pickup and delivery problems: Part 1: Transportation between customers and depot. *Journal für Betriebswirtschaft*, 58(1), 21-51.

Potvin, J.Y., Rousseau, J.M. (1995) An exchange heuristic for routing problems with time windows. *Journal of the Operational Research Society*, 46, 1433-1446.

Qureshi, A.G., Taniguchi, E., Yamada, T. (2009) An exact solution approach for vehicle routing and scheduling problems with soft time windows. *Transportation Research Part E*, 45, 960-977.

Qureshi, A.G., Taniguchi, E., Yamada, T. (2010) Exact solution for vehicle routing problem with semi soft time windows and its application. *Procedia Social and Behavioral Sciences*, 2, 5931-5943.

Qureshi, A.G., Taniguchi, E., Yamada, T. (2012) Exact solution for vehicle routing problem with soft time windows and dynamic travel time. *Asian Transport Studies*, 2(1). 48-63.

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

Solomon, M.M. (1987) Algorithms for the vehicle routing and scheduling problems with time

window constraints. *Operation Research*, 35(2), 254-265.

Subramanian, A., Cabral, L.D.A.F. (2008) An ILS based heuristic for the vehicle routing problem with simultaneous pickup and delivery and time limit. Proceedings of the 8th European Conference on Evolutionary Computation in Combinatorial Optimization, Naples, Italy, March 26-28.

Subramanian, A., Drummond, L.M.A., Bentes, C., Ochi, L.S., Farias, R. (2010a) A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37, 1899-1911.

Subramanian, A., Ochi, L.S., Uchoa, E. (2010b) New lower bounds for the vehicle routing problem with simultaneous pickup and delivery. In Festa, P. (eds.), *Experimental Algorithms: Lecture Notes in Computer Science*, 6049, 276-287. Springer, Berlin-Heidelberg.

Tagmouti, M., Gendreau, M., Potvin, J.Y. (2007) Arc routing problems with time-dependent service costs. *European Journal of Operational Research*, 181, 30-39.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F., Potvin, J.Y. (1997) A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2), 170-186.

Taniguchi, E., Kakimoto, Y. (2003) Effects of e-commerce on urban distribution and the environment. *Journal of the Eastern Asia Society for Transportation Studies*, 5, 2355-2366.

Taniguchi, E., Shimamoto, H. (2004) Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation Research Part C*, 12, 235-250.

Taniguchi, E., Thompson, R.G., Yamada, T., Duin, R.V. (2001) *City logistics: Network modelling and intelligent transport systems*. Pergamon, Oxford.

Tasan, A.S., Gen, M. (2012) A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering*, 62, 755-761.

The U.S. Agricultural Trade Office Japan, U.S. Embassy, Tokyo (2011) *Japan Retail Food Sector Report 2011*, GAIN Report Number: JA1524. (Online) Available: http://gain.fas.usda.gov/Recent%20GAIN%20Publications/Retail%20Foods_Tokyo%20ATO_Japan_12-21-2011.pdf (accessed November 6, 2012).

Toth, P., Vigo, D. (2002) *The vehicle routing problem: SIAM Monographs on Discrete Mathematics and Applications*, SIAM, Philadelphia.

Villeneuve, D., Desrosiers, J., Lubbecke, M.E., Soumis, F. (2005) On compact formulations for integer programs solved by column generation. *Annals of Operations Research*, 139, 375-388.

Wang, H.F., Chen, Y.Y. (2012) A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering*, 62, 84-95.

Westphal, S., Krumke, S.O. (2008) Pruning in column generation for service vehicle dispatching. *Annals of Operations Research*, 159, 355-371.