7e Conférence Internationale de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril 2008 – Paris- France
« Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination.».

A NEW HEURISTIC ALGORITHM FOR THE VRPTW

YA LIU

ICD-LOSI, Université de Technologie de Troyes
12 rue Marie Curie BP2060
10010 Troyes Cedex
Ya.liu@utt.fr

CHENGBIN CHU

ICD-LOSI, Université de Technologie de Troyes
12 rue Marie Curie BP2060
10010 Troyes Cedex
chu@utt.fr

RÉSUMÉ : This paper attempts to use an idea already applied successfully in a cutting stock problem, to solve approximately a classical vehicle routing problem with time windows (VRPTW) which is well known to be NP-hard. This idea is based on dynamic programming by aggregating states to avoid the explosion of the number of states. Because of the similarities of a series of set partitioning problems, including cutting stock problems and vehicle routing problems (VRP), our objective is to develop a general method based on this idea to solve a series of set partitioning problems. The VRPTW is an application of this idea. Computational results show that our algorithm is efficient for a dataset with narrow time windows, compared to algorithms specially dedicated to this problem.

MOTS-CLÉS: *VRPTW, set partitioning, dynamic programming, knapsack problem*

1. INTRODUCTION

The vehicle routing problem with time windows (VRPTW) is one of the attractive topics in logistic optimization in last decades. It is known to be NP-hard in computational complexity theory. It can be described as the problem of designing routes for serving a set of geographically scattered demand points. In these routes, each vehicle starts and ends at the depot, visiting scattered points, subject to capacity constraints. Each point is visited exactly by one vehicle within a given time interval. In this paper, we treats the given time interval as a hard time window. If a vehicle arrives at a customer earlier, it must wait. In addition, due dates cannot be violated.

Early research focuses mainly on exact technique and almost all the algorithms use one of the following three principles: Dynamic Programming, Lagrange Relaxation-based method and Column Generation.

Dynamic programming approach is first presented by Kolen, Rinnooy Kan and Trienekens (1987). Instances with up to 15 customers are solved. Fisher, Jornsteen, and Madsen (1997) use a K-tree approach followed by Lagrange relaxation. The most successful approach up to now is the column generation method proposed by Desrosiers, Soumis and Desrochers (1984) which yields a tight lower bound.

Over the last few years, much progress has been made in the development of approximate algorithms of VRPTW. The traditional heuristic approaches are route construction and route improvement methods. Route construction method consists of selecting nodes until a feasible solution is created. The first paper using this method is route-first cluster-second scheme proposed by Solomon (1986). It is an extension of the savings heuristic of Clarke and Wright (1964).The concept of route improvement method is iteratively improving the initial feasible solution by exploring neighbouring ones. Edge-exchange algorithms expressed as r-Opt are the most used, where r arcs are removed and replaced by r other arcs. Potvin and Rousseau (1995) compared different edge exchange methods, like 2-opt, 3-opt and Or-opt.

Metaheurustics form another branch of heuristics. Compared with traditional heuristic approaches, they allow deterioration of the current solution to generate future better solutions in the search process. Glover (1986) introduces Tabu search. Garcia et al. (1994) make the first application to VRPTW problem. Holland (1975) proposes the genetics idea to solve complex problems. And Thangiah et al. (1991) first apply the genetic algorithm to VRPTW problem.

In this paper, we develop a new heuristic algorithm for VRPTW. This algorithm comes from the dynamic programming idea. In general, it gives a global view of the overall optimization problem, in contrast to other myopic methods where the solution is constructed very locally in a step-by-step manner. The final aim of our research is to solve a series of set partitioning problems, with this dynamic programming idea. And this paper is an application used in VRPTW background to check the performance of the idea. In this particular background, our algorithm has also some advantages. Firstly, heterogeneous fleets are easy to be considered. Secondly, compared with exact techniques, it can solve very complex VRPTW. Compared with metaheuristics, it is not parameter dependent while the performance of metaheuristics is often related to the value of parameters and per-

sonal experiences play an important role in the determination of these values.

This paper is organized as follows. Section 2 describes in detail this new heuristic method. Section 3 focuses on solving subproblem used as subroutines for the global problem. In order to evaluate the performance of our algorithms, computational results are reported in Section 5.

2. THE GENERAL METHOD

2.1. Problem Description

The VRPTW can be described as follows.

$$MIN \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij} \sum_{k=1}^{N} x_{ijk}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{N} x_{ijk} = 1 \qquad i \neq 0 \tag{1}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{N} x_{ijk} \leq M_k \qquad i = 0 \tag{2}$$

$$\sum_{i=0}^{n}\sum_{k=1}^{N} x_{ijk} = 1 \qquad j \neq 0 \tag{3}$$

$$\sum_{j=0}^{n}\sum_{k=1}^{N} x_{ijk} = \sum_{i=0}^{n}\sum_{k=1}^{N} x_{ijk} \qquad i = 0 \tag{4}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{n} x_{ijk} d_j \leq P_k \qquad k = 1,2...N \tag{5}$$

$$\sum_{i,j \in S} x_{ijk} \leq |S| - 1, \text{ for all } S \in 2^{v/\{0\}} \qquad k = 1,2...N \tag{6}$$

$$T_i + U_i + t_{ij} - L(1 - x_{ijk}) \leq T_j \tag{7}$$

$$readytime_j \leq T_j \leq duetime_j \tag{8}$$

We suppose the depot is a virtual customer point indexed 0.The notations are explicated in table1.

The objective function states the traveling cost should be minimized. Constraints (1), (3) ensure each customer is visited exactly once. Constraint (2) states for each vehicle type, the vehicle quantity selected to serve all customers is no more than the quantity available. Constraints (4) ensures the vehicle leaves depot and returns to the depot in the end. Constraints (5) means that the vehicle is loaded no more than its capacity P. (6) ensures the connectivity of the solution. (7), (8) are time window constraints.

Then, we formulate the VRPTW problem as a set partitioning model. The set partitioning is one of the widely used models in the field of combinatorial optimization. Many real-life problems such as vehicle routing, facility location and airline crew scheduling can be formulated as a set partitioning problem. It is concerned with partitioning a given set of resources to process a set of activities. A cost is incurred in the processing of a subset of activities by a resource in a sequence.

M_k	number of vehicles of type k
n	number of customers to be served
d_j	demand of customer j
c_{ij}	travelling cost from customer i to customer j
P_k	The capacity of vehicle k
U_i	the service time at customer i by vehicle
t_{ij}	the travelling time from customer i to customer j
T_i	the vehicle arriving time at customer i
$readytime_i$	the earliest beginning service time of customer i
$duetime_i$	the latest ending service time of customer i
x_{ijk}	if vehicle k visits customer j immediately after customer i, x_{ijk} =1. Otherwise, x_{ijk} =0
L	A very large scalar
N	types of vehicles

Table1. List of notations

In our problem, a route is analogous to a set formed by a number of customers as activities and one vehicle as resource. Our problem is to identify each route by knowing which type of vehicle is assigned in this route and which customers are served by this vehicle. Let R be the set of all feasible routes. For a given route, δ_{ir} is a constant that takes the value 1 if route $r \in R$ visits customer i and 0 otherwise. σ_{jr} takes the value 1 if a route r involves a vehicle of type j and 0 otherwise. c_r is the cost of route r. Finally, let x_r be a binary variable equal 1 if route r is used and 0 otherwise. The set partitioning problem selects a minimal cost of routes satisfying the VRPTW constraints:

$$\min \sum_{r \in R} c_r x_r$$

$$\sum_{r \in R} \delta_{ir} x_r = 1 \qquad \forall i \in \{1,2,...n\} \tag{9}$$

$$\sum_{r \in R} \sigma_{jr} \leq M_j \qquad \forall j \in \{1,2,...N\} \tag{10}$$

$$x_r \in \{0,1\}, \qquad r \in R \tag{11}$$

Constraint (9) ensures that each customer is visited exactly once. (10) states for a given type ,the number of vehicles used can't exceed the available number. For a large size set partitioning problem, it is impossible to enumerate all feasible routes. We are trying to find several sets which cover all customers and minimize the total travel cost. Column generation is usually used to solve set partitioning problem. It yields an excellent lower bound but the solution generated is not always

feasible, so a branch-and-bound algorithm is embedded to get integer results. In contrast, our algorithms can obtain a feasible solution directly.

2.2. The algorithms

The problem can be solved using a dynamic programming idea. This idea is proposed by Chu and Antonio (1999) and successfully applied to a cutting stock problem. There are some similarities between the vehicle routing problem and cutting stock problem. Both of them belong to set partitioning problems. The cutting stock problem is to partition the output rods into subsets and assign input rods to each of the subsets. We believe that the idea used in cutting stock problem can also be used successfully in vehicle routing problem with time windows. In order to demonstrate this algorithm clearly, a complete list of notation is given in Table2.

.

B_0	set of vehicles available, $B_0 = (M_1, M_2, ..., M_N)$
C_0	set of customers to be served, $C_0 = (m_1, m_2, ..., m_n)$
B	subset of B_0
C	subset of C_0
$\gamma(B,C)$	the minimum cost of using a vehicle subset B serving a customer subset C
$h(B-B^{'},C-C^{'})$	the minimum cost of using a single vehicle described by $B-B^{'}$ to visit customer set $C-C^{'}$
$\gamma'(\alpha,\beta)$	the minimum cost of serving β customers using α vehicles
$U(\alpha,\beta')$	the set of remaining vehicles corresponding to the minimum cost $\gamma'(\alpha,\beta')$.
$V(\alpha,\beta')$	the set of remaining customers to be served corresponding to the minimum cost $\gamma'(\alpha,\beta')$.
$\bar{U}(\alpha,\beta,\beta')$	the vehicle set used corresponding to the function $h'(\alpha,\beta,\beta')$
$\bar{V}(\alpha,\beta,\beta')$	the customers set used corresponding to the function $h'(\alpha,\beta,\beta')$
$d_{\min}$	minimum demand of customers
$capacity_{\max}$	maximum vehicle capacity
$\eta_{\max}$	the maximum number of customers served by one vehicle

Table 2. List of notations

Let B_0 , C_0 denote the set of all vehicles available and the set of all customers respectively. According to the problem described in Section2.1, B_0 is an N-vector whose i^{th} entry is M_i meaning M_i vehicles are available for type i ,so $B_0 = (M_1, M_2, ..., M_N)$ .Similarly, $C_0 = (m_1, m_2, ..., m_n)$ whose i^{th} entries equals to 1 meaning the i^{th} customer to be served . Any subset of vehicles is also an N-entry that can be represented by B whose i^{th} entry is the quantity of vehicles of type i (i = 1, 2, …, N). If B' is a subset of B, then we have B' ≤ B which means that for any i ∈ {1, 2, …, N}, the i^{th} entry of B' is less than or equal to the i^{th} entry of vector B. In a similar way, any subset of customers is an n-entry whose i^{th} entry is equal to 1 if the i^{th} customer is to be served and 0 otherwise. As mentioned above, the set partitioning problem is concerned with partitioning a given set of resources to process a set of activities. Our objective is to partition the customers set C_0 as activity set and to assign resource from B_0 to each of the subsets to minimize the travel cost.

$\gamma(B,C)$ is the minimum cost of using a vehicle subset B serving a customer subset C .The definition $\gamma(\phi,\phi) = 0$ provides the boundary condition for the dynamic programming procedure. The recursive equation follows:

$$\gamma(B,C) = \min_{\substack{B' \le B \\ C' \le C \\ e_N^T(B-B')=1}} \{\gamma(B',C') + h(B-B', C-C')\} \qquad (12)$$

where $h(B-B^{'},C-C^{'})$ is the minimum cost of using all vehicles belonging to set $B-B^{'}$ to visit all customers belonging to set $C-C^{'}$, this equation subjects to $e_N^T(B-B')=1$, e_N is a vector with N ones, T denotes the transposition operators. As $e_N^T(B-B')=1$ is mentioned, only one vehicle is available from set $B-B^{'}$ to serve all customers in set $C-C^{'}$ .Compared with the shortest path problem where dynamic programming approach are widely used, the $\gamma(B,C)$ is analogous to the shortest path cost from starting state $(0,0)$ to state (B,C); $h(B-B^{'},C-C^{'})$ is analogous to the edge cost from one state to another. The shortest distance can be realized by finding transferred states recursively until all the customers can be visited at the final state. So we are only interested in the minimum cost $\min_{B \le B_0} \gamma(B,C_0)$, the corresponding optimal solution can be found in a backward manner. In this algorithm, all possible combinations of subset B, C must be considered. The complexity of this algorithm is $\prod_{i=1}^{N}(M_i+1)\times 2^n$. For large size problem, it is impossible to use this algorithm.

To simplify this, we identify all combinations of subset B , C using the number of vehicles and customers. We replace state (B,C) by (α,β) where α,β specify the number of vehicles used in subset B , the number of customers visited in subset C respectively. Define $\gamma'(\alpha,\beta)$ which is analogue to $\gamma(B,C)$, denoting the minimum cost of serving β customers using α vehi-

cles. Similarly to the recursive equation above, the new recursive equation is:

$$\gamma'(\alpha+1,\beta)=\min_{\alpha\le\beta'<\beta}\left\{\gamma'(\alpha,\beta')+h'(\alpha,\beta,\beta')\right\} \tag{13}$$

The definition of the function $h'(\alpha,\beta,\beta')$ could be the minimum cost of visiting $\beta-\beta'$ customers using one vehicle. Simply speaking, the function is the least cost of serving $\beta-\beta'$ customers by one vehicle. This subproblem is solved in detail in the next section.

Let $U(\alpha,\beta')$ and $V(\alpha,\beta')$ be the set of remaining vehicles and the set of remaining customers to be served corresponding to the minimum cost $\gamma'(\alpha,\beta')$. From state (α,β') to state $(\alpha+1,\beta)$, $\beta-\beta'$ customers must be served by one vehicle. The vehicle and customers should be chosen from $U(\alpha,\beta')$ and $V(\alpha,\beta')$ respectively. We define $\bar{U}(\alpha,\beta,\beta')$ and $\bar{V}(\alpha,\beta,\beta')$ respectively, as the vehicle set and the set of customers used corresponding to the function $h'(\alpha,\beta,\beta')$. Let β^* be the value of β' that gives the minimum value in (13).We have:

$$U(\alpha+1,\beta)=U(\alpha,\beta^*)-\bar{U}(\alpha,\beta,\beta^*) \tag{14}$$

$$V(\alpha+1,\beta)=V(\alpha,\beta^*)-\bar{V}(\alpha,\beta,\beta^*) \tag{15}$$

The specific steps of this algorithm are stated:

(1) Scan the demands of all customers, find the minimum demand $d_{\min}$, $d_{\min}=\min\left\{d_i\,|\,i=1,2,...n\right\}$.

Define the maximum vehicle capacity $capacity_{\max}$.

.So, the maximum number of customers served by one vehicle $\eta_{\max}$ is:

$$\eta_{\max}=\frac{capacity_{\max}}{d_{\min}}$$

(2) Initialization:

$$U(0,0)=B_0, V(0,0)=C_0$$
$$\gamma'(0,0)=0$$

(3) From the recursive equation, we have:

$$\gamma'(1,1)=h'(0,1,0),$$
$$\gamma'(1,2)=h'(0,2,0),\ldots$$
$$\gamma'(1,\eta'_{\max})=h'(0,\eta'_{\max},0)$$

Where $h(0,1,0),h(0,2,0)...h(0,\eta_{\max},0)$ are computed from $U(0,0)$, $V(0,0)$. And we get $U(1,1),V(1,1),U(1,2),V(1,2),...U(1,\eta_{\max}),V(1,\eta_{\max})$ .The equation continues:

$$\gamma'(\alpha,\beta)=\min\left\{\gamma'(\alpha-1,\beta')+h'(\alpha-1,\beta,\beta')\right\}$$

The condition is:

$$0\le\beta-\beta'\le\eta_{\max},\beta'\ge\alpha-1.$$

So, $\max\left(\beta-\eta_{\max},\alpha-1\right)\le\beta'\le\beta$. The recursive equation stops when $\beta=e_n^TC_0$.

(4) Find the optimal solution in a backward manner to identify each route by considering the best paths, starting from the best final state.

3 SOLVING SUBPROBLEM

3.1 The formulation of the subproblem

Now, we will focus on solving the function $h'(\alpha,\beta,\beta')$ as subproblem. It can be described as selecting $\beta-\beta'$ customers to be served by one vehicle from a set of customers within a specified time interval to minimize the cost. Let $m=\beta-\beta'$. Let T_i be the vehicle arriving time at customer *i*, if the arriving time of customer i is earlier than its beginning service time $readytime_i$, the vehicle must wait until the beginning time, if the arriving time is later than its end time $duetime_i$, the customer can not be serviced. U_i is the service time at customer *i* by vehicle, t_{ij} is the travelling time from customer *i* to customer *j*. c_{ij} is the travelling cost from customer *i* to *j*. The objective is to minimize the total distance starting from the depot, visiting each customer selected and returning to the depot. Mathematical model of this subproblem is formulated as follows.

$$\min\sum_i\sum_j c_{ij}x_{ij}$$

St:

$$\sum_{i=1}^{n}x_{0i}=1 \tag{16}$$

$$\sum_{i=1}^{n}x_{i0}=1 \tag{17}$$

$$\sum_{j=0}^{n}x_{ji}=\sum_{j=0}^{n}x_{ij}=y_i \tag{18}$$

$$\sum_{i=1}^{n}y_id_i\le P \tag{19}$$

$$\sum_{i=1}^{n}y_i=m \tag{20}$$

$$\sum_{i,j\in S}x_{ij}\le|S|-1\quad\text{For all } S\in 2^{V/\{0\}} \tag{21}$$

$$T_i + U_i + t_{ij} - K(1 - x_{ij}) \le T_j \quad (22)$$

$$readytime_j \le T_j \le duetime_j \quad (23)$$

$$x_{ij} \in \{0,1\} \quad (24)$$

$$y_i \in \{0,1\} \quad (25)$$

The objective function states the cost should be minimized. Variable x_{ij} takes value 1 if that vehicle visits customers *j* from customer *i* and 0 otherwise. Variable y_i =1 means that the customer *i* is visited by the vehicle, y_i =0 means the opposite case. Constraints (16), (17) ensure that the vehicle leaves the depot and returns to the depot finally. Constraint (18) states the flow constraints and defines y_i . Constraints (19) means that the vehicle is loaded no more than its capacity P. Constraints (20) states that m customers must be assigned to the vehicle. (21) ensures the connectivity of the solution. (22), (23) are time window constraints, K is a large scalar.

3.2 The algorithm

We transform the problem into a knapsack problem and solve it by using classical dynamic programming. We suppose a vehicle like a hitch-hiker having to fill up his knapsack by selecting a given number of objects from various possible objects.

First, we define middle time as the middle of the time interval. Customers are scheduled in the increasing order of their middle time. In the ordered list, customer positioning in the list has its own label. The label set is $Z = \{z_1, z_{2,} \ldots z_n\}$, with z_i meaning the label of the customer in the i^{th} position. Let $\pi_k(b)$ be the minimum cost of selecting b customers from customers in the first k^{th} positions of the list ordered. The dynamic programming recursion equation is as follows:

$$\pi_k(b) = \min\left\{\pi_{k-1}(b), \min f(y_1^*, y_2^*, y_3^* \ldots y_{b-1}^*, z_k)\right\} \quad (15)$$

$(y_1^*, y_2^* \ldots\ldots y_{b-1}^*)$ is the optimal solution corresponding to $\pi_{k-1}(b-1)$, where y_j^* (j=1,2……b-1) is the label of customer selected. The function $f(y_1^*, y_2^*, y_3^* \ldots y_{b-1}^*, z_k)$ is the travel distance starting from the depot, visiting customers labelled $(y_1^*, y_2^*, y_3^* \ldots y_{b-1}^*, z_k)$, and returning to the depot. In fact, solving $\min f(y_1^*, y_2^*, y_3^* \ldots y_{b-1}^*, z_k)$ is the same as solving a travelling salesman problem. The problem is described as finding the shortest route to visit a collection of customers and return to the depot considering time windows. In this paper, we apply a branch- and-bound algorithm to solve this problem. The basic method is to break up the set of all tours into smaller and smaller subsets and to calculate a lower bound for each of them. The bounds guide the partitioning of the subsets and identify an optimal solution eventually. In order to find a feasible solution quickly, we use depth first rule for branching. The node with the smallest lower bound has the highest priority in branching. We use the lower bound proposed by Little et al (1963).

4 AN EXAMPLE

In order to demonstrate this heuristic algorithm clearly, we take a small example.

We have two vehicle types in the depot to serve customers; the first type has only one vehicle available with its capacity 40, the second type has two vehicles available with its own capacity 60. So, $B_0 = (1,2)$. Nine customers scatter geographically with demands 20, 25, 20, 22, 21, 22, 20, 24, 20 respectively. So, $C_0 = (1,1,1,1,1,1,1,1,1)$, $capacity_{\max} = 60$, $d_{\min} = 20$, $\eta_{\max} = 3$, $\gamma'(0,0) = 0$, $U(0,0) = B_0$, $V(0,0) = C_0$.
Following recursive equation (13),

$$\gamma'(1,1) = \min\{\gamma'(0,0) + h'(0,1,0)\}$$
$$\gamma'(1,2) = \min\{\gamma'(0,0) + h'(0,2,0)\}$$
$$\gamma'(1,3) = \min\{\gamma'(0,0) + h'(0,3,0)\}$$

$h'(0,1,0)$, $h'(0,2,0)$, $h'(0,3,0)$ is computed by the algorithm described in section 3.2 with condition $U(0,0) = B_0$, $V(0,0) = C_0$ .For example, $h'(0,2,0)$ means the minimum cost of serving two customers by one vehicle .The vehicle and customers should be chosen from the vehicle set $U(0,0)$ and customers set $V(0,0)$. Then $U(1,1)$, $V(1,1)$, $U(1,2)$, $V(1,2)$, $U(1,3)$, $V(1,3)$ are obtained by equation(14),(15). The equation continues,

$$\gamma'(2,1) = \min\{\gamma'(0,0) + h'(0,1,0)\}$$

$$\gamma'(2,2) = \min\begin{Bmatrix}\gamma'(0,0) + h'(0,2,0)\\ \gamma'(1,1) + h'(1,2,1)\end{Bmatrix}$$

$h'(0,2,0)$ is the minimum cost of serving two customers in set $U(0,0)$ by one vehicle in set $V(0,0)$ .And $h'(1,2,1)$ is the minimum cost of serving one customer in set $U(1,1)$ by one vehicle in set $V(1,1)$.
Until $\gamma'(3,9)$, the recursion equation stops, the corresponding 'optimal solution' can be found in a backward manner from $\gamma'(3,9)$.

5 EVALUATIONS

Solomon (1987) generated six VRPTW data sets which are commonly accepted as the main benchmark for VRPTW algorithms. They are made up of 100 customers.

Homberger and Gehring(1999) developed test problems for large size VRPTW problem in which the number of customers ranges from 200 to 1000. We test our algorithm in the datasets of C1. C1 means the customers geographically clustered. Table3 compares some results using the described algorithm with the best known solutions obtained by Homberger and Gehring(1999).

We define narrow time window and wide time window as: Time interval ration=customer's time interval /the depot's time interval. If the time interval ration is more than 20%, we consider this time window is wide time window. If in one dataset, more than 50% customers have wide time windows, we consider the dataset a wide time window case. Table4 compares the result between the narrow time window cases and wide time window cases in C1.

Frankly speaking, the algorithm is effective for dataset with narrow time windows. For the wide time window cases, results are not as competitive as the best known solutions. We attribute this shortcoming to the solution of the subproblem.Further effort should be devoted to solving the subproblem with wide time windows more effectively. The signification of this subproblem study is explained as: Although the vehicle routing problems have been intensively studied, the researches mainly focused on the tradition models, for example capacitated vehicle routing problem, multi-depot vehicle routing problem. They are interested in exploiting a number of combinatorial properties, improving a series of algorithms from these models. From the practical point of view, many industrial situations are even more complicated based on the classical vehicle routing problems. The subproblem demonstrated above is one of them. The given number of customers selected is added as constraint in traditional VRPTW problem. In practice, this model has many applications. For example, if the depot is not only a site to deliver products to customers, but also provide a service. So the service ability is limited by the human resource. Like a repair centre, the repairman can only service a given number of customers because of technological constraints. So the further study of this subproblem continues.

6 CONCLUSION

This paper uses the dynamic programming idea to solve the VRPTW problem. We propose a new heuristic algorithm based on this idea. The aim of our research is to develop a general method which can be applied to a series of set partitioning problems especially large size instances. Compared with column generation method, this approach is easy to generate a feasible near optimal solution. Choosing the VRPTW problem is an attempt to verify this idea. And the results make us consider some other application background using this idea. We believe that this approach will be paid attention and become a useful method to solve a series of set partitioning problems. Also, the application in VRPTW problem can be improved if a better method is found to solve the subproblem.

Future work will be dedicated to apply this new heuristic method to other set partition problems, like cutting stock problem and operation room problem.

Customer number	Data-set	Number of vehicles	Total distance	Computation Time(CPU seconds)	Gap
100	C101	10	828.94	63.7	0%
	C102	10	842.61	97.8	1.65 %
	C105	10	828.94	60.7	0%
	C107	10	828.94	62.5	0%
200	C1_2_1	20	2709.67	729	0.19 %
	C1_2_5	20	2709.66	639.8	0.28 %
	C1_2_7	20	2709.67	635.5	0.32 %
400	C1_4_1	40	7195.5	5951	0.61 %
	C1_4_5	40	7602.95	5836	6.3%
	C1_4_7	40	7609.945	5935.5	6.4%

Table3. Computational results1

Time window type	Average gap
Narrow	0.4125%
Wide	21%

Table4. Computational results2

REFERENCES

CHU C.,and J. Antonio, 1999. Approximation Algorithms to Solve Real-Life Multicriteria Cutting Stock Problems.*Operations Research*, Volume 47, Issue 4 , pp. 495 – 508.

Clarke G., J. W. Wright, 1964.Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, Vol. 12, No. 4), pp. 568-581.

Desrosiers J., Soumis F., and Desrochers M., 1984.Routing with time windows by column generation.*Networks*, Vol.14, Issue 4, pp. 545-565.

Fisher M. L., Ornsteen K.O.J., and Madsen O. B. G., 1997.Vehicle routing with time windows: Two optimization algorithms. *Operations Research*, 45(3):488-492.

Garcia B.-L., Potvin J.-Y.,and Rousseau J.-M.,1994. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints. *Comput. Oper. Res*. 21 1025–1033.

Glover F., 1986.Future paths for integer programming and links to artificial intelligence.*Comput. Oper Res* 13 533–549.

Holland J. H., 1975.Adaptation in Natural and Artificial Systems.*University of Michigan Press, Ann Arbor, MI*.

Kolen A.W.J, Rinnooy Kan A.H.G, and Trienekens H.W.J.M, 1987.Vehicle routing with time windows. *Operations Research,* Volume 35, Issue 2, pp. 266 – 273.

Little J. D. C., Murty K.G., Sweeney D.W., and Karel C., 1963.An Algorithm for the Traveling Salesman Problem.*Operations Research*, Vol. 11, No. 6, pp. 972-989.

Potvin J.-Y.,and Rousseau J.-M., 1995.An exchange heuristic forrouteing problems with time windows. *J.Oper .Res.Soc.* 46 1433–1446.

Solomon M.M., 1986.On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks* 16 161–174.

Thangiah, S.R.,Nygard K.E., and Juell P. L.,1991. GIDEON: A genetic algorithm system for vehicle routing with time windows. *Proc. 7th IEEE Conf. Artificial .Intelligence Appl., IEEE Computer Society Press, Los Alamitos*, 322–328.