An efficient column generation heuristic for vehicle routing with multiple use of vehicles for a rental business

Takuya HIROTA*, Susumu MORITO* and Kento HARA*

* Department of Industrial and Management Systems Engineering, School of Creative Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku-ku, Tokyo, 169-8555, Japan
E-mail: morito@waseda.jp

Abstract
Optimization of daily vehicle routes with deliveries and pickups as well as multiple use of vehicles for a rental business is considered. The problem is formulated as a variant of generalized set covering problems in which variables correspond to workdays of vehicles. Two variants of column generation heuristic algorithms are developed: one to obtain near-optimal solutions with moderate amount of CPU time together with the associated lower bound, and the other to get solutions with maximum of 8% GAP within 30 CPU seconds. The speed-up of the algorithm is achieved by approximately solving subproblems which exploits information of optimal dual prices associated with the restricted LP master problem. The column generation heuristic algorithms are applied successfully not only to single-depot problems but also to multi-depot problems.

Key words: Vehicle routing, Multiple use of vehicles, Column generation heuristic, Elementary shortest paths with resource constraints, Dual price

1. Introduction

This paper addresses cost effective transportation for a company renting prefabricated unit houses often used at construction sites, event sites, disaster areas and others for temporary housing. These housing units can be assembled or disassembled quickly with ease. Customers are classified into two groups: those to whom rental products should be delivered, and those from whom products should be picked up at the completion of the rental period. Since the company delivers to or picks up from customers rented housing units, generation of their daily vehicle routes becomes a vehicle routing problem (VRP) in the general category of “VRP with backhauls” (See, e.g., Toth and Vigo, 2002). Because of the size of the products and the limited capacity of vehicles, individual vehicles often make several “trips” (as defined precisely later) from a depot in a day, thus leading to “VRP with multiple use of vehicles” or sometimes called “multitrip VRP” (Mingozzi et al., 2013). There are relatively a few VRP studies considering multiple use of vehicles and heuristic algorithms are used in most of the cases to obtain solutions.

The problem is formulated as a variant of generalized set covering problems in which variables correspond to workdays of vehicles. Two variants of column generation heuristic algorithms are developed and applied successfully not only to single-depot problems but also to multi-depot problems. These two variants differ in their way to solve column generation subproblems. One variant which is called the complete algorithm obtains near-optimal solutions with moderate amount of CPU time together with the associated lower bound and achieves high quality solutions to both single-depot and multi-depot problems. The other variant is called the accelerated algorithm and approximately solves subproblems exploiting information of optimal dual prices associated with the restricted LP master problem. The accelerated algorithm allows to obtain solutions with maximum of 8% GAP within 30 CPU seconds to both types of the problems.

Azi et al. (2010) address the problem with time windows and multiple use of vehicles in anticipation of its applications in the home delivery of perishable goods. They introduce a branch-and-price approach where lower bounds are computed by solving the linear programming relaxation of a set packing formulation, using column generation. Their network representations of column generation subproblems are utilized in this paper. Inoue et al. (2014) consider the
same case as this study and present an integer programming formulation together with a solution approach which solves integer programs repeatedly by augmenting “time-compatibility constraints.” Their solution approach produces solutions within 30 minutes, but the company wants to obtain solutions in a few minutes. Their formulation also has a problem that the transportation cost is only approximately evaluated. On the other hand, we evaluate cost exactly and our accelerated algorithm produces solutions faster than theirs.

2. Vehicle routing for a rental business

We consider optimization of daily vehicle routes for a company renting prefabricated unit houses. Our goal is to generate a schedule minimizing the total cost of transportation that the company pays to truck companies. A vehicle trip is defined to be an elementary circuit originating from and ending at a depot. We also define a sequence of trips where a vehicle performs in a day as “workday.”

2.1. Basic assumptions of the single-depot problem

Basic assumptions (except for Section 5) are listed below:

(1) There exists a single depot from which products are delivered and to which products are returned.
(2) Products are delivered to new rental customers, and are picked up from customers upon completion of rental.
(3) There exist several product types and the numbers and the types of rented products are given for each delivery and pickup.
(4) Generally there exist customer requested time windows for delivery and pickup.
(5) Each vehicle could make several trips within its working hours of a day thus forming a workday.
(6) There exist two types of vehicles, large and small, and only limited number of vehicles are available for each type.
(7) Housing units can be loaded on vehicles either in a “disassembled” (or “fold”) mode or in an “assembled” (or “unfold”) mode.
(8) The numbers of products that each types of vehicles can carry is limited, and is dependent on the product type as well as the loading mode.
(9) The numbers of deliveries and pickups on a particular trip of a vehicle are limited to two and one, respectively.
(10) Pickup should follow delivery(s) in each trip.
(11) Different types of products cannot be loaded on the same vehicle.
(12) Travel time and time needed for “operations” at customer’s sites are all given constants.
(13) The problem calls for a vehicle schedule that minimizes the cost of transportation as calculated based on the agreement between the company and truck companies.

In Section 5, we relax the first assumption of single depot to consider multi-depot problems.

2.2. Cost calculation

We now describe how the cost of transportation is calculated at the company. Note that the cost is not a simple linear function of actual distance of the trips. The calculation depends on the sequence of customer visits and also on the number of trips for each vehicle. The cost is generally calculated for each customer based on the distance from depot to the customer and also on the type of vehicle used. The cost of a visit can be read from a given table of tariff.

For the first (round) trip from a depot, the first customer visit of the trip will be charged 100% of the designated cost, whereas the second and possibly third visits of the trip are charged only 60% of the cost. We call this type of cost calculation for the first trip of a workday Type 1 cost calculation. For the second and possibly third (round) trips, on the other hand, the trip’s first visit will be charged 80% (rather than 100%) of the cost, whereas the second and possibly the third visits of the trip are charged 60% of the designated cost. We call this type of cost calculation for the second and third trip of a workday Type 2 cost calculation.

The above means that the cost is less if more customers are visited in a single (round) trip, and also if more (round) trips are made for a vehicle (Inoue et al., 2014, Morito et al., 2016).

3. Optimization by mathematical programming

3.1. Typical problem size and solution approach

In our single-depot case, the number of customers in a day including deliveries and pickups is between 40 and 70.
In addition, there exist at most 3 customers in a trip and at most 3 trips in a workday. Because of the limited parameter sizes of the problems, the number of possible trips is expected to be manageable in size, and we thus opt to enumerate in advance all possible trips meeting constraints. We then make workdays by combining these trips, and select a set of workdays which meet all customer demands with the smallest possible cost. However, the number of possible workdays is huge so we apply column generation.

3.2. Generation of all feasible trips

We first generate all feasible trips satisfying a given set of constraints. We consider the following 5 trip patterns.

I depot ÷ delivery ÷ depot
II depot ÷ pickup ÷ depot
III depot ÷ delivery ÷ pickup ÷ depot
IV depot ÷ delivery ÷ delivery ÷ depot
V depot ÷ delivery ÷ delivery ÷ pickup ÷ depot

In the case of patterns IV and V which involve two deliveries, there may exist two trips which consist of identical set of customers with different order of visits. Because of time windows of customers, there may be idle time between the two deliveries, waiting for the start time of time window of the latter delivery. Comparing such idle times of the two alternatives, we exclude the trip from consideration which has longer idle time.

The steps to generate all feasible trips are given below.

Step 1 Enumerate all of the back and forth trips(I, II).

Step 2 Enumerate “delivery ÷ delivery” and “delivery ÷ pickup” trips which satisfy all constraints.

Step 3 Enumerate “delivery ÷ delivery ÷ pickup” trips if the enumerated “delivery ÷ delivery” trips can be expanded to include a pickup.

Step 4 If a trip visits at least two customers (i.e., either one of patterns III, IV, or V) and violates either one of the two constraints below, the trip is excluded:
  (1) the distance between customers is over 50km,
  (2) the total time required is over 13 hours.

Step 5 If there exist trips which have the same combination of customers with different order of visits, the trips which have longer total times are excluded.

3.3. Route graph

Based on the generated trips, we construct a “route graph” (Azi et al., 2010) in which a trip corresponds to a node and a directed arc is drawn between two nodes if the associated trips can be performed in the order specified by the direction of arc without violating time windows.

In a sample route graph shown in Fig. 1, trip node 0120 represents a trip in which a vehicle departs from the depot (denoted as 0), visits customer 1 and customer 2 in this order, and then returns to the depot. Costs assigned to arcs of the route graph will be described below in Section 3.6. Although there exist arcs connecting the dummy start node with all trip nodes, not all of them are shown in Fig.1 to make the figure simple. The thick solid line in Fig.1 describes one of the solutions of the pricing subproblem described below in Section 3.6, which corresponds to the workday consisting of three trips, depot ÷ customer1 ÷ depot ÷ customer3 ÷ customer2 ÷ depot ÷ customer3 ÷ depot.

![Route graph](image_url)
3.4. Integer programming formulation

We give the notations used in the IP formulation.

Sets
The set of all customers including both delivery and pickup is \( V \). The set of vehicle types is \( L \). The set of workdays for a vehicle of type \( l \in L \) is \( W^l \).

Constants
The cost of workday \( w \) for vehicle of type \( l \) is \( c^l_w \). The number of times customer \( i \) is visited in workday \( w \) is \( a^i_w \). The minimum number of times to visit customer \( i \) using vehicle of type \( l \) in order to meet demand of customer \( i \) is \( b^l_i \). The number of available vehicles of type \( l \) is \( k^l \).

Variables
If workday \( w \) for a vehicle of type \( l \) is selected, \( x^l_w \) is 1, 0 otherwise.

The integer programming master problem which we call IPM is then formulated as follows.

\[
\min \sum_{l \in L} \sum_{w \in W^l} c^l_w x^l_w \quad \text{(1)} \\
\text{s.t.} \quad \sum_{w \in W^l} a^i_w x^l_w \geq b^l_i \quad \forall i \in V, \forall l \in L \quad \text{(2)} \\
\quad \sum_{w \in W^l} x^l_w \leq k^l \quad \forall l \in L \quad \text{(3)} \\
\quad x^l_w \geq 0 \text{ and integer} \quad \forall l \in L, \forall w \in W^l \quad \text{(4)}
\]

Here, (1) is minimization of the total route cost, (2) is a constraint of the minimum visiting times for each customer, (3) indicates that the number of selected workdays is less than or equal to the number of available vehicles.

Since there exist no constraints that involve both of the two types of vehicles, the problem is separable by vehicle type and thus we solve the integer programming master (IPM) independently for each vehicle type.

3.5. Column generation

As the number of columns can be huge, the columns are progressively introduced into IPM to obtain a series of restricted integer programming master programs (RIPMs). The linear relaxation of each RIPM which we call RLPM is then solved. The dual variables associated with the optimal solution of a given RLPM are used to define a pricing subproblem which identifies workdays with a negative reduced cost, if any. These workdays are then added to the current RLPM to obtain the next RLPM, which is solved again to obtain new dual variables. This iterative procedure is repeated until no more workdays with negative reduced cost can be found. At this point, an optimal solution for the LP relaxation of IPM, LPM is obtained.

Since we identify a sequence of customers in a trip and also since we know if the trip is the first trip or subsequent trip of the workday, we can calculate the exact cost the company pays to truck companies.

To initiate column generation procedure, initial columns will be needed. Initial columns of workdays are generated while an initial feasible solution of the problem is created by assigning trips, in the ascending order of cost of a trip per customer in the trip, to vehicles. When a trip is added to a vehicle, the time-compatibility of the trip with the trips already assigned to the vehicle (if any) will be checked.

3.6. Pricing subproblem

In the pricing subproblem, we find workdays of the minimum reduced cost using the route graph as in Fig.1. We give the notations used in the pricing subproblem.

Sets
The set of nodes of the route graph is \( V^T \). The set of customers in route \( s \) is \( V_s \). The set of arcs of the route graph is \( A^T \).
### Constants

Each route node \( r \) in this graph has two time windows which are derived from the time windows of the customers served in that route, namely, \([\bar{t}_r, \bar{t}_r]\) and \([\bar{t}_{r+1}, \bar{t}_{r+1}]\), where \( \bar{t}_r^0 \) and \( \bar{t}_r^0 \), \( \bar{t}_{r+1}^0 \) and \( \bar{t}_{r+1}^0 \) are the earliest departure time, latest departure time, earliest arrival time and latest arrival time, respectively. \( A_t \) is a dual variable of constraint (2), whereas \( \mu \) is a dual variable of constraint (3). \( c_{rs} \) is transportation cost associated with arc \((r, s)\) connecting trip nodes \( r \) and \( s \), whereas \( \sigma_s \) is a setup time for loading at trip node \( s \). \( m \) is the maximum number of trips. \( M \) is an arbitrary large constant.

### Variables

\( X_{rs} \) indicates if arc \((r, s)\) is selected or not in the route graph. \( T_r \) corresponds to the departure time of route \( r \).

The problem is then formulated as follows.

\[
\min \sum_{(r, s) \in A^T} (c_{rs} - \sum_{i \in U_r} A_i) X_{rs} - \mu \tag{5}
\]

subject to

\[
\sum_{(r, s) \in A^T} X_{rh} - \sum_{(h, s) \in A^T} X_{hs} = 0 \quad \forall h \in V^T \tag{6}
\]

\[
\sum_{r \in V^T} X_{0r} = 1 \tag{7}
\]

\[
\sum_{r \in V^T} X_{r(i+1)} = 1 \tag{8}
\]

\[
\sum_{r \in V^T} X_{rt} \leq m + 1 \quad \forall r \in V^T \tag{9}
\]

\[
T_r + \sigma_s + (\bar{t}_{r+1} - \bar{t}_r) - M(1 - X_{rs}) \leq T_s \quad (r, s) \in A^T \tag{10}
\]

\[
\bar{t}_r^0 \leq T_r \leq \bar{t}_r^0 \quad r \in V^T \tag{11}
\]

\[
X_{rs} \in \{0, 1\} \quad (r, s) \in A^T \tag{12}
\]

\[
T_r \geq 0 \quad \forall r \in V^T \tag{13}
\]

Here (5) is minimization of the reduce cost, (6), (7), (8) are flow conservation constraints that describe the individual routes, (9) is a constraint that the maximum number of trips is \( m \) for each vehicle (\( m=3 \) in our case), (10) defines vehicle arrives at the route node \( r \) and can depart from the route node \( s \) only after loading, (11) is a time window constraint of the departure time.

Pricing subproblems are often solved as a resource constrained shortest path problem on the route graph. An arc of the graph is drawn between trip nodes if the corresponding trips are time-wise compatible. Generally, cost of an arc of the route graph corresponds to the cost of the trip associated with the destination node of the arc minus the dual price of the trip, which is calculated as the sum of dual prices corresponding to the customers visited by the trip.

Remember that the cost calculation of a trip depends whether the trip is the first trip of a workday or the second or third trip of a workday as described in Section 2.2. In the route graph, the first trip of a workday means that the associated node is connected to a dummy start node, and thus cost of an arc connecting a dummy start node with a trip node is determined based on the Type 1 calculation as described in Section 2.2. Any destination node of an arc connecting two trip nodes are always either the second or the third trip of a workday, and thus the associated cost must be determined based on the Type 2 calculation.

For example, the arc cost connecting a dummy start node to node 0120 in Fig.1 is the cost of trip 0120 as calculated by the Type 1 calculation minus sum of dual prices associated with customers 1 and 2. On the other hand, the arc cost connecting trip 010 and trip 0320 is the cost of trip 0320 as calculated by the Type 2 calculation minus sum of dual prices associated with customers 3 and 2.

### 3.7. Solving the pricing subproblem

A standard procedure to solve the resource-constrained shortest path problem of the pricing subproblem is the label correcting algorithm (Feillet et al., 2004). However, it was found that using only the labeling algorithm in a straightforward way requires too much time, and thus we opt to use the labeling algorithm in conjunction with solutions of the pricing subproblems (5)-(13) obtained by a commercial solver. Steps of the algorithm are described below where each step moves to the next step when no column of the negative reduced cost is found.

**Step 1** \( \theta \% \) of the total nodes are selected in ascending order of the cost of nodes which is the cost of associated trip...
minus the associated dual price, and the rest of the nodes are eliminated from the route graph together with arcs incident to the deleted nodes. The pricing subproblem is solved using the resultant graph with the additional condition that the maximum number of trips in a workday is two.

**Step 2** Find a trip with negative reduced cost. That is, find a workday with negative reduced cost consisting of a single trip.

**Step 3** Solve the pricing problem (5)-(13) by a commercial IP solver.

Step 3 gives the optimal solution of the LP relaxation of IPM, and thus provides a lower bound on the optimal value of IPM. We call the algorithm which performs all of these three steps the complete algorithm. As will be shown in Section 4.1., the complete algorithm necessitates moderate amount of CPU time. One of our study goals is to find a reasonable solution rapidly, say, within 60 seconds so that company could use the scheduling system in more-or-less an interactive mode. For this purpose, we consider a variant of algorithms in which only Step 1 and Step 2 are performed, which we call the accelerated algorithm. Note that this variant only solves the LP relaxation approximately, and thus does not guarantee the lower bound of IPM.

4. **Computational experiments**

We evaluate performance of solutions obtained by the complete and accelerated algorithms by comparing our results with those obtained by Inoue et al. (2014) and also with actual schedules created by scheduling personnel of the company. The accuracy of the accelerated algorithm will be evaluated based on the lower bound obtained by the complete algorithm.

The computing platform for all experiments is a Core i7 3.10 GHz with 16 GB of RAM, using AMPL-Gurobi (6.0.4) to solve the RLMPs in column generation and the IPM with only those columns generated by column generation to obtain a feasible solution and thus an upper bound of the original problem. The accuracy of solutions is evaluated by the duality gap

\[ \text{GAP} = \frac{\text{upper bound} - \text{lower bound}}{\text{lower bound}} \times 100(\%) \].

Table 1 summarizes the six real data sets used in our experiments, together with the number of trips generated.

| Problem size | No. | # Customers | # Vehicles | # Trips |
|--------------|-----|-------------|------------|---------|
|              |     | Delivery    | Large      | Small   | Large | Small |
| 1            | 41  | 14          | 23         | 44      | 339   | 490   |
| 2            | 38  | 8           | 30         | 44      | 289   | 327   |
| 3            | 37  | 25          | 27         | 45      | 621   | 1,109 |
| 4            | 27  | 14          | 25         | 39      | 113   | 145   |
| 5            | 40  | 9           | 30         | 50      | 534   | 641   |
| 6            | 35  | 10          | 30         | 50      | 506   | 539   |

4.1. **Performance of the complete algorithm**

Table 2 summarizes the results obtained by the complete algorithm. The numbers of columns (=workdays) generated during solving the LP master (LPM) are generally small and are substantially less than the total number of feasible workdays which could be millions. Despite the column generation heuristic which solves IP master with only generated columns, the resultant solutions are very close to optimal as indicated by small GAPs. We also note that almost 100% of the CPU time is spent in solving the LP master (LPM), and time to solve IP master with generated columns is basically negligible.

The complete algorithm yields high quality solutions with the GAP of less than 1% except for data No.6. The complete algorithm beats with good margin the Inoue’s approach in terms of the costs of the solutions in all data sets, but the CPU times of the complete algorithm are roughly same as or more than Inoue’s. CPU times in Table 2 do not include time to generate all feasible trips, which only takes a few seconds. Manual schedules made by scheduling personnel are available only for four data sets. For three out of these four data sets, the complete algorithm yields better results. It turns out, however, that, for data No.2 cost of the manual schedule is lower than the cost obtained by the complete algorithm. The possible explanations behind this “reverse” phenomenon might be 1) the manual schedule generated by scheduling personnel actually violated some of the constraints, and 2) even the complete algorithm is only a column generation heuristic.

Comparing Tables 1 and 2, and also recalling that the problems are solved independently for each type of vehicles, small and large, one can observe that the problems with 400 or less trips (= nodes) can be solved within 100 seconds.
As the number of trips exceeds 400, however, the number of feasible workdays rapidly increases, and so does CPU time. In fact, data No.3 took almost 4,000 CPU seconds. The objective of the complete algorithm is to find less costly vehicle schedules, and thus the approach could be practically usable if moderate CPU time of, say 1 hour is available.

Table 2  Comparison of schedules generated by the two methods and also manually in practice

| DataNo. | # Columns | Total cost | Lower bound | GAP(%) | CPU(sec) | Total cost | CPU(sec) | Total cost | Improvement ratio(%) |
|---------|-----------|------------|-------------|--------|----------|------------|----------|------------|----------------------|
| 1       | 372       | 1,656,000  | 1,655,000   | 0.06   | 688      | 1,814,500  | 529      | 1,770,000  | 6.44                 |
| 2       | 267       | 1,880,000  | 1,877,750   | 0.12   | 111      | 2,046,000  | 58       | 1,858,000  | -1.18                |
| 3       | 438       | 2,009,000  | 2,008,600   | 0.02   | 3,931    | 2,078,500  | 497      | 2,022,500  | 0.67                 |
| 4       | 177       | 1,752,500  | 1,751,500   | 0.06   | 41       | 1,762,500  | 20       | 1,815,500  | 3.47                 |
| 5       | 409       | 1,564,500  | 1,560,390   | 0.26   | 921      | -          | -        | -          | -                    |
| 6       | 321       | 1,272,000  | 1,254,500   | 1.39   | 964      | -          | -        | -          | -                    |

4.2. Performance of the accelerated algorithm

Table 3 shows the results of the accelerated algorithm when $\theta=5\%$. The lower bounds shown in Table 3 are the lower bounds of the original problem, which are the optimal values of the LP relaxation of IPM obtained by the complete algorithm. Although there is a difference of the CPU time due to a difference of the data size, we are able to find the solution with the GAP of at most 6% and average of 2.74% within 1 minute.

Table 3  Performance of the accelerated algorithm when $\theta=5\%$

| Data No. | Total cost | Lower bound | GAP(%) | CPU(sec) |
|----------|------------|-------------|--------|----------|
| 1        | 1,688,500  | 1,655,500   | 2.02   | 4        |
| 2        | 1,899,500  | 1,877,750   | 1.16   | 3        |
| 3        | 2,115,000  | 2,000,600   | 5.72   | 57       |
| 4        | 1,768,000  | 1,751,500   | 0.94   | 4        |
| 5        | 1,615,500  | 1,560,390   | 3.53   | 19       |
| 6        | 1,293,000  | 1,254,500   | 3.07   | 6        |

Figure 2 shows the relationship between the CPU time and the number of trip nodes in the route graph. The CPU time generally increases as the number of trip nodes increases. Behavior of the graph for data No.6 is somewhat different from others, but increase of CPU time as the number of nodes increases is only a general trend and is not an absolute one. When the number of nodes is around 30, we are able to solve the problem within 10 seconds, whereas the CPU time becomes over 100 seconds as the number of nodes approaches 50 depending on the data set used. The CPU time clearly increases rapidly as the number of nodes increases.

![Relationship between the CPU time (in seconds) and the number of trip nodes in the route graph](image)

Table 4 shows the results when the number of nodes is set to 30 in order to solve the problems within the target time of 10 seconds. The average CPU time is 11 seconds with the average GAP of 2.58%.
Table 4 Performance of the accelerated algorithm when the number of nodes in the route graph is set to 30

| Data No. | Nodes | Large | Small | Total cost | GAP (%) | CPU (sec) |
|----------|-------|-------|-------|------------|---------|-----------|
| 1        | 339   | 490   | 1,682,500 | 1.63 | 6 |
| 2        | 289   | 327   | 1,899,500 | 1.16 | 10 |
| 3        | 621   | 1,109 | 2,124,000 | 6.17 | 9 |
| 4        | 113   | 145   | 1,758,000 | 3.53 | 13 |
| 5        | 534   | 641   | 1,615,500 | 3.53 | 13 |
| 6        | 506   | 539   | 1,287,500 | 2.63 | 13 |

Table 5 shows details of column generation processes for the complete algorithm. Specifically, the table shows the number of columns and the associated CPU time during each phase of the entire procedure. Naturally, the largest proportion of columns is generated in Step 1. Note that multiple number of columns with negative reduced cost are generated in each iteration of Step 1. The majority of CPU time, on the other hand, will be consumed in Step 3 to solve the column generation subproblem with a commercial solver. Note that only one column with the “minimum” reduced cost will be generated in each iteration.

Table 5 Details of column generation processes (the complete algorithm)

| Data No. | Initialization | Step 1 | Step 2 | Step 3 | IP | Total |
|----------|----------------|--------|--------|--------|----|-------|
|          | # Columns | CPU(s) | # Columns | CPU(s) | # Columns | CPU(s) | # Columns | CPU(s) | # Columns | CPU(s) | # Columns | CPU(s) |
| 1        | 50         | 2      | 223     | 9      | 25        | 1      | 74        | 657    | 2          | 372    | 688 |
| 2        | 66         | 1      | 150     | 2      | 25        | 1      | 26        | 106    | 1          | 267    | 111 |
| 3        | 62         | 5      | 201     | 36     | 65        | 1      | 110       | 3,887  | 2          | 438    | 3,931 |
| 4        | 52         | 1      | 79      | 2      | 8         | 0      | 38        | 38     | 0          | 177    | 41 |
| 5        | 45         | 2      | 278     | 29     | 19        | 1      | 67        | 887    | 2          | 409    | 921 |
| 6        | 33         | 2      | 166     | 18     | 57        | 1      | 65        | 942    | 1          | 321    | 964 |

5. Extension to multi-depot problems

The current practice of the company is to assign customers to depots first, and then generate vehicle routes for each depot independently. The company is considering to switch to a system in which customer assignment and vehicle routes are determined at the same time for a group of depots in a certain area, in anticipation that the new system would further reduce transportation cost.

The new system would necessitate to solve the multi-depot extensions of the problem. The algorithm presented earlier for single-depot problems can easily be extended to multi-depot problems. In the multi-depot extension, product types that could be handled by a given depot is generally a subset of all product types which are given in advance. The number of vehicles of each type available at a given depot is known. Trips of each vehicle must start and end at the depot the vehicle belongs to.

As before, all possible trips are enumerated in advance, this time for each depot. The rest of the column generation procedure is same as the single depot case, except that the pricing subproblems are solved independently for each depot.

Table 6 Problem size of multi-depot extension

| Data No. | Depot ID | # Customers Delivery | # Customers Pickup | # Vehicles Large | # Vehicles Small | # Trips Large | # Trips Small |
|----------|----------|----------------------|--------------------|-----------------|-----------------|--------------|--------------|
| 1        | A        | 28                   | 2                  | 58              | 58              | 39           | 615          |
|          | B        | 0                    | 9                  | 0               | 0               | 0            | 0            |
|          | C        | 0                    | 0                  | 0               | 0               | 0            | 0            |
|          | D        | 0                    | 0                  | 0               | 0               | 0            | 0            |
|          | E        | 0                    | 0                  | 0               | 0               | 0            | 0            |
| 2        | A        | 27                   | 2                  | 11              | 11              | 63           | 615          |
|          | B        | 0                    | 0                  | 0               | 0               | 0            | 0            |
|          | C        | 0                    | 0                  | 0               | 0               | 0            | 0            |
|          | D        | 0                    | 0                  | 0               | 0               | 0            | 0            |
|          | E        | 0                    | 0                  | 0               | 0               | 0            | 0            |
| 3        | A        | 25                   | 2                  | 15              | 15              | 15           | 256          |
|          | B        | 0                    | 0                  | 0               | 0               | 0            | 0            |
|          | C        | 0                    | 0                  | 0               | 0               | 0            | 0            |
|          | D        | 0                    | 0                  | 0               | 0               | 0            | 0            |
|          | E        | 0                    | 0                  | 0               | 0               | 0            | 0            |
Table 7 Comparison of schedules generated by three different methods

| No. | Data | Our method (Complete algorithm) | Our method (Accelerated algorithm) | Manual depot assignment + solution of single-depot problems |
|-----|------|---------------------------------|-----------------------------------|----------------------------------------------------------|
|     |      | Total cost | Lower bound | GAP (%) | CPU (sec) | Total cost | GAP (%) | CPU (sec) | Total cost | Improvement ratio 1(%) | Improvement ratio 2(%) |
| 1   |      | 2,867,500  | 2,863,880   | 0.13    | 7,668     | 2,963,000 | 3.46    | 29        | 3,116,500  | 7.99                  | 4.93                  |
| 2   |      | 3,011,500  | 3,010,000   | 0.06    | 839       | 3,082,500 | 2.41    | 16        | 3,185,500  | 5.45                  | 3.23                  |
| 3   |      | 2,449,500  | 2,445,500   | 0.10    | 2,229     | 2,628,000 | 7.46    | 12        | 2,644,000  | 7.40                  | 0.60                  |

Table 6 summarizes input information for 3 real data sets with 5 depots in a certain region. Table 7 shows results of the complete algorithm and the accelerated algorithm, together with results when the customers’ depot assignment actually given by the company is used and then vehicle routes are generated by solving the single-depot problem for each depot. Remember that the company currently performs customers’ depot assignment first and then vehicle routing in a sequential manner. In the accelerated algorithm, \(\theta\) is set to 5%.

The complete algorithm takes non-trivial amount of CPU time even though GAPs are reasonably small. Data No.1 which is the largest among 3 tested data sets with 96 customers takes more than 7,000 seconds. If we forget about accurate lower bound with the accelerated algorithm with \(\theta = 5\%\), however, all 3 instances are solved within 30 seconds with average GAP of 4.44\% (based on the lower bounds obtained by the complete algorithm).

The right-most block of Table 7 shows the total costs when customers are assigned to the depots by the planning personnel in actual practice, and then the single-depot problem is solved for each depot, together with how our methods reduce total costs. “Improvement ratio 1 (2)” shows the reduction of total cost in percentage when the complete (accelerated) algorithm is applied. For example, for data No.1, the complete algorithm generates vehicle schedule whose cost is 7.99\% less, whereas the accelerated algorithm 4.93\% less.

Table 8 Comparison of schedules generated by two different algorithms

| Data No. | \(\theta\) (%) | Complete algorithm | Accelerated algorithm |
|----------|----------------|--------------------|-----------------------|
|          |                | Total cost (\%) | Lower bound (\%) | GAP (%) | CPU (sec) | Total cost (\%) | GAP (%) | CPU (sec) | Total cost (\%) | Improvement ratio 1(%) | Improvement ratio 2(%) |
| 1        | 5              | 2,867,500         | 2,863,880          | 0.13    | 7,668     | 2,963,000        | 3.46    | 29        | 3,116,500        | 7.99                  | 4.93                  |
| 2        | 10             | 2,867,500         | 2,863,880          | 0.13    | 6,958     | 2,951,500        | 3.06    | 95        | 3,082,500        | 2.41                  | 16                    |
|          | 20             | 2,867,500         | 2,863,880          | 0.11    | 7,875     | 2,937,500        | 2.57    | 1,955     | 3,011,800        | 0.06                  | 839                   |
| 2        | 5              | 3,011,500         | 3,010,000          | 0.06    | 1,083     | 3,078,000        | 2.26    | 56        | 3,055,500        | 1.51                  | 501                   |
|          | 10             | 3,011,500         | 3,010,000          | 0.05    | 1,361     | 3,087,500        | 1.98    | 124       | 3,011,300        | 0.04                  | 1,361                 |
| 3        | 5              | 2,449,500         | 2,445,500          | 0.10    | 2,174     | 2,628,000        | 7.46    | 12        | 2,448,000        | 6.97                  | 42                    |
|          | 10             | 2,449,500         | 2,445,500          | 0.08    | 2,722     | 2,593,500        | 6.05    | 495       | 2,447,500        | 0.08                  | 2,722                 |

Finally, Table 8 summarizes results for different values of \(\theta\). Note that CPU time of the accelerated algorithm increases rather quickly as \(\theta\) is increased.

6. Conclusion

In this paper, we proposed a column generation approach for a vehicle routing problem of a rental business with multiple use of vehicles. We were able to find near-optimal solutions for most of test data sets using CPU time of roughly one hour using the complete algorithm. With the present approach, we are able to reflect accurately the very specific way of calculating transportation cost at the company. Moreover, we were able to speed up computations substantially in the accelerated algorithm by reducing the size of the route graph based on information obtained from dual prices of RLPM, and also by sacrificing the lower bound information. Solutions with the average GAP of 2.58\% and the maximum GAP of roughly 8\% were obtained within 30 seconds for single-depot problems. The proposed column generation heuristic algorithms are found to be effective not only to single-depot problems but also to multi-depot problems.

Acknowledgements

This work was supported by JSPS Kakenhi Grant No.26350437.
References

Azi, N., Gendreau, M. and Potvin, J.-Y., An exact algorithm for a single-vehicle routing problem with time windows and multiple routes, European Journal of Operational Research, Vol.178 (2007), pp.755-766.
Azi, N., Gendreau, M. and Potvin, J.-Y., An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles, European Journal of Operational Research, Vol.202 (2010), pp.756-763.
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, Vol.44 (2004), pp.216-229.
Inoue, T., Nakahara, R., Hirota, T. and Morito, S., Optimization of vehicle routes with delivery and pickup for a rental business: a case study, Abstracts of the 2014 Fall National Conference of Operations Research Society of Japan (2014), pp.42-43. (in Japanese)
Mingozzi, A., Roberti, R. and Toth, P., An exact algorithm for the multitrip vehicle routing problem, INFORMS Journal on Computing, Vol.25 (2013), pp.193-207.
Morito, S., Inoue, T., Nakahara, R. and Hirota, T., Optimization of vehicle routes with delivery and pickup for a rental business: a case study, in Lübbecke, M., Koster, A., Letmathe, P., Madlener, R., Peis, B. and Walther, G., eds., Operations Research Proceedings 2014, pp.407-413, Springer, Switzerland (2016).
Toth, P. and Vigo, D., VRP with backhauls, in Toth, P. and Vigo, D., eds., The Vehicle Routing Problem, pp.195-224, SIAM, Philadelphia (2002).