A heuristic algorithm for the periodic vehicle routing problem with flexible delivery dates

Hang DONG*, Yosuke TAKADA*, Wei WU** and Mutunori YAGIURA*
* Graduate School of Informatics, Nagoya University
  Furocho, Chikusaku, Nagoya 464-8601, Japan
  E-mail: dong.hang@nagoya-u.jp
** Graduate School of Integrated Science and Technology, Shizuoka University
  3-5-1 Johoku, Naka-ku, Hamamatsu, Shizuoka, 432-8561, Japan

Received: 11 December 2019; Revised: 7 April 2020; Accepted: 22 May 2020

Abstract
The periodic vehicle routing problem (PVRP) is a generalization of the vehicle routing problem (VRP) to a planning period of more than one day. A fleet of vehicles needs to deliver goods periodically to each customer so that the delivery dates meet the request of the customer. This paper introduces a new model of PVRP, the periodic vehicle routing problem with flexible delivery dates, which covers a wider range of applications and is able to answer more diverse customer requirements. We propose an algorithm based on iterated local search (ILS), which is a metaheuristic approach that involves an iterative application of a local search algorithm and the use of perturbation as a diversification mechanism. Computational results show that our proposed method performed competitively in solving the standard PVRP. Moreover, we confirm that our new model can reduce the cost for more general instances.

Keywords: Periodic vehicle routing problem, Flexible delivery dates, Iterated local search, Metaheuristics

1. Introduction

The vehicle routing problem (VRP) has been intensively studied in order to solve problems associated with distribution or collection. However, many real-life logistical problems are inherently periodic: each customer requires a number of deliveries during a planning period of several days or more, with an appropriate interval between every two consecutive deliveries. Examples of such periodic applications are elevator maintenance, garbage collection, vending machine restocking, and recurring deliveries by online shopping sites such as Amazon.com. Let us take garbage collection as an example. The garbage collector is in charge of a set of customers who need a collector to come twice during a week. The set of customers is so large that it is not feasible to visit them all in the same day; hence the collector needs to decide for each customer on which two days to visit them, and on each day to decide the order by which the customers will be visited. A method commonly used is to separate the problem into two phases: the first phase to decide the dates to visit customers, and the second phase to solve a vehicle routing problem for each day. However, optimizing the entire repetitive operation can reduce costs compared with scheduling by such two-phase methods. Such a problem, which tries to optimize these two phases at the same time, is called the periodic vehicle routing problem (PVRP) and was first introduced by Beltrami and Bodin (1974). This problem can be considered as a generalization of the conventional VRP, which aims to find an optimal set of daily routes for a planning period.

This paper considers a PVRP with flexible delivery dates. This is different from the standard PVRP, which usually has a small number of candidates for delivery date combinations. With more flexibility on the day assignment, our proposed model covers a wider range of applications and it is able to deal with more diverse requests for customer schedules. Returning to the example of garbage collection scheduling, it is covered by the standard PVRP. Each customer has regular combinations of dates to have garbage collected (e.g., twice every week, with the allowable collection date combinations of Monday/Thursday or Tuesday/Friday) because it is important for residents to have fixed days to have garbage collected.
However, there are cases in which more flexibility is desirable, for example, a monthly elevator maintenance assignment for which January 3rd/February 3rd and January 3rd/February 5th might both be allowable.

For this PVRP with flexible delivery dates, we propose an iterated local search (ILS) algorithm. We also propose an improvement method based on dynamic programming to reassign customers from one allowable date combination to another. Through computational experiments, we confirm that our proposed method performed competitively in solving the standard PVRP. Additionally, the results of experiments on the PVRP with flexible delivery dates show that by allowing flexibility on the delivery dates, it will be able to reduce the distribution cost and serve customer deliveries on their willing dates.

2. Problem Description

2.1. Periodic vehicle routing problem

In the periodic vehicle routing problem, a given set of \( n \) customers is periodically supplied by a fleet of \( m \) vehicles operating from a depot for a given planning period, where each customer \( i \) requires \( h_i^{\text{max}} \) visits in the whole planning period. The objective is to design optimal routes for this fleet of vehicles for each day in the period. The total cost to be minimized includes the cost associated with the distances traveled, vehicle capacity, and so on.

Periodic vehicle routing problems have several ways to define the allowable delivery schedules, such as asking customers to specify several allowable delivery date combinations or to specify an interval \( r_i \) and imposing a constraint that requires two successive deliveries to be spaced by exactly \( r_i \) days. In our model, we define allowable delivery schedules in a more flexible way.

2.2. PVRP with flexible delivery dates

We describe our periodic vehicle routing problem with flexible delivery dates. From this section onwards, we call a vehicle a “worker.” The delivery or service to a customer on a given day is called a “task,” with a task processing time spent at the customer’s location. The time from when the worker departs from the depot until it returns to the depot, including the travel time and the task processing time is called “working time.”

The problem is to assign each customer to one of its allowable sets of date combinations, and to create daily routes such that the total cost incurred by the working time of all workers is minimized, under the following constraints:

(a) For each day, the route of a worker starts from the depot and ends at the depot after completing all the visits if the worker is on duty that day.

(b) For each day, the number of on-duty workers does not exceed the given number of workers \( m \).

(c) Each worker has a limited capacity. That is, for each day, the total demand of customers assigned to a worker does not exceed the worker capacity.

(d) The total working time of a worker in a day does not exceed the duration of on-duty hours.

(e) Customer \( i \) must get \( h_i^{\text{max}} \) deliveries during the whole planned period, and the \( h_i \)th delivery \( (h \leq h_i^{\text{max}}) \) must be done on a day in \( \{u_{ih}, \ldots, v_{ih}\} \) \((u_{ih} \leq v_{ih})\).

(f) The interval between the \((h - 1)\)th and the \( h \)th deliveries for a customer \( i \) must be at least \( a_{ih} \) days and at most \( b_{ih} \) days.

To describe the PVRP with flexible delivery dates more precisely, we define the following notations:

\(-\)

\(T: \) the set of days in the planning period, \( T = \{1, 2, \ldots, T^{\text{max}}\} \),

\(N: \) a depot and the set of \( n \) customers, \( N = \{0, 1, 2, \ldots, n\} \) (0 denotes the depot, and 1, 2, \ldots, \( n \) denote \( n \) customers),

\(M: \) the set of \( m \) workers, \( M = \{1, 2, \ldots, m\} \),

\(l_{ij}: \) travel time between customers (or depot) \( i \) and \( j \),

\(e_i: \) service time of customer \( i \) (for the depot, \( e_0 = 0 \)),

\(d_i: \) the demand of customer \( i \) to be delivered on day \( t \),

\(w_i: \) the value representing unwillingness or inconvenience of customer \( i \) to be visited on day \( t \). In particular, when customer \( i \) totally refuses to be visited on day \( t, w_i \) is \( +\infty \),

\(Q_i: \) capacity of worker \( k \) on day \( t \),

\(h_i^{\text{max}}: \) the number of deliveries required by customer \( i \) during the entire period of \( T^{\text{max}} \) days \((h_i^{\text{max}} \leq T^{\text{max}})\).
The objective of the problem we consider can be formulated as follows:

\[
\min \sum_{k \in M} \sum_{t \in T} G_{kt} + \sum_{(i,t) \in U} W_{it},
\]  

(1)

where

\[
G_{kt} = g_{kt}C_{kt} + (\sigma_{kt}^{\text{over}} - g_{kt}) \max \{0, C_{kt} - r_{kt}\}
\]

\forall k \in M, \forall t \in T,

(2)

We denote by \( \sigma_{kt} \) the delivery route of worker \( k \in \{1, 2, \ldots, m\} \) on day \( t \in \{1, 2, \ldots, r_{t}^{\text{max}}\} \), where \( \sigma_{kt}(s) = i \) signifies that the \( s \)th visit of worker \( k \) on day \( t \) is to customer \( i \). We define \( |\sigma_{kt}| \) as the total number of customers visited by worker \( k \) on day \( t \) (not including the depot). Then, for all the workers, on all the days, we must have \( \sigma_{kt}(0) = 0 \) and \( \sigma_{kt}(|\sigma_{kt}| + 1) = 0 \) to satisfy constraint (a), which requires each worker to start from the depot and at the end of the day to return to the depot (or never leave the depot for those not on duty that day). In addition, we define \( x_{ikth} \) by

\[
x_{ikth} = \begin{cases} 1 & \text{if the } h \text{th delivery of customer } i \text{ is performed by worker } k \text{ on day } t, \\ 0 & \text{otherwise}, \end{cases}
\]

that is, if \( x_{ikth} \) takes value 1, \( \sigma_{kt}(s) = i \) holds for an \( s \).

The objective of the problem we consider can be formulated as follows:

\[
\min \sum_{k \in M} \sum_{t \in T} G_{kt} + \sum_{(i,t) \in U} W_{it},
\]

(1)

where

\[
G_{kt} = g_{kt}C_{kt} + (\sigma_{kt}^{\text{over}} - g_{kt}) \max \{0, C_{kt} - r_{kt}\}
\]

\forall k \in M, \forall t \in T,

(2)
that the positions in the existing routes for a set of appropriate delivery dates so as to minimize the increase in cost.

If the delivery dates and the orders of other remaining tasks in the routes do not change, re-insert those deleted tasks in the best neighborhoods.

3.1.1. Reoptimize one customer (reOptOne)

We consider six neighborhoods in the local search: the four standard ones, 2-opt, 2-opt*, cross exchange, and relocate exchange (interested readers can refer to Bráysy and Gendreau (2005) for more details); and two other neighborhoods, called reoptimize one customer (reOptOne) and reoptimize all customers (reOptAll), both of which use a dynamic programming (DP) technique. We omit the explanations of the standard neighborhoods and focus on these two DP-based neighborhoods.

3. Proposed method

In this section, we propose an iterated local search algorithm for the PVRP with flexible delivery dates. This algorithm involves the iterative application of a local search algorithm and the use of perturbation, called a kick, as a diversification mechanism.

3.1. Local search

Local search starts from an initial solution and iteratively replaces the current solution \( \sigma \) with a better solution \( \sigma' \) in its neighborhood of candidate solutions \( N(\sigma) \) by applying local changes until no better solution can be found in the neighborhood.

We consider six neighborhoods in the local search: the four standard ones, 2-opt, 2-opt*, cross exchange, and relocate exchange (interested readers can refer to Bráysy and Gendreau (2005) for more details); and two other neighborhoods, called reoptimize one customer (reOptOne) and reoptimize all customers (reOptAll), both of which use a dynamic programming (DP) technique. We omit the explanations of the standard neighborhoods and focus on these two DP-based neighborhoods.

\[
C_{kt} = \sum_{i=1}^{|\sigma_k|+1} (l_{\sigma_k(i)} + c_{\sigma_k(i)}) \quad \forall k \in M, \forall t \in T, \quad (3)
\]

\[
W_{it} = u_{it} \sum_{k \in M} \sum_{h=1}^{h_{\text{max}}} x_{ikh} \quad \forall (i, t) \in U_{\text{soft}}. \quad (4)
\]

Function (1) minimizes the total cost, which consists of the total value of the workers’ wage (2) and the total inconvenience of customers (4). Equation (2) is worker \( k \)'s one-day wage on day \( t \), which includes a normal wage and over time pay. Equation (3) presents worker \( k \)'s working time on day \( t \). The constraints (b)–(f) can be expressed by (5)–(9), respectively:

\[
\sum_{k \in M} \min \left\{ \sum_{i \in N \setminus \{0\}} \sum_{h=1}^{h_{\text{max}}} x_{ih}, \right\} \leq m \quad \forall t \in T, \quad (5)
\]

\[
\sum_{i \in N \setminus \{0\}} \sum_{h=1}^{h_{\text{max}}} d_{ih} x_{ih} \leq Q_d \quad \forall k \in M, \forall t \in T, \quad (6)
\]

\[
C_{kt} \leq t_{\text{max}}^{\sigma_k} \quad \forall k \in M, \forall t \in T, \quad (7)
\]

\[
\sum_{t = t_{\text{min}}}^{t_{\text{max}}} \sum_{k \in M} \sum_{h=1}^{h_{\text{max}}} x_{ikh} = 1 \quad \forall i \in N \setminus \{0\}, \forall h \in \{1, 2, \ldots, h_{\text{max}}^{\sigma_k}\}, \quad (8)
\]

\[
a_{ih} \leq \sum_{k \in M} \sum_{t = t_{\text{min}}}^{t_{\text{max}}} l x_{ikth} - \sum_{k \in M} \sum_{t = t_{\text{min}}}^{t_{\text{max}}-1} l x_{ikth-1} \leq b_{ih} \quad \forall i \in N \setminus \{0\}, \forall h \in \{2, 3, \ldots, h_{\text{max}}^{\sigma_k}\}. \quad (9)
\]
where

\[ T_{th}(t) = \{ t - b_{th}, \ldots, t - a_{th} \} \cap \{ u_{ih-1}, \ldots, v_{ih-1} \}, \]

\[ \Delta_{th} = \min_{k \in M, c \in \{ 1, \ldots, |\sigma_k| \}} \Delta_{kh}, \]

and \( \Delta_{kh} \) represents the cost increase caused by inserting customer \( i \) to the \( h \)th position of the route \( \sigma_k \). The set \( T_{th}(t) \) represents the set of possible delivery days for the \((h-1)\)th delivery of customer \( i \) provided that the \( h \)th delivery is on day \( t \). Then we can obtain the minimum cost increase for delivering customer \( i \) exactly once in each frequency period during the entire scheduling period by

\[ F_i = \min_{\{u_{ih-1}, \ldots, v_{ih-1}\}} f_{ih}(t). \]

The minimum cost increase of inserting customer \( i \) to a specific day \( t \) can be computed in \( O(\sum_{h=1}^{m} |\sigma_h|) = O(n) \) time, and the total computation time to prepare \( \Delta_{th} \) for all \( t \) is \( O(n_{max}) \). For each \( i \), let \( \tilde{c}_{i}^{\text{sum}} = \sum_{h=1}^{m} (v_{ih} - u_{ih} + 1) \) and \( \xi_{ih} = \max_{\{1, \ldots, n_{max}\}} \min\{b_{ih} - a_{ih}, v_{ih-1} - u_{ih-1}\} + 1 \). We only need to compute \( f_{ih}(t) \) for those combinations of \( h \) and \( t \) such that \( t \in \{u_{ih}, \ldots, v_{ih}\} \), and the number of such combinations is \( \xi_{ih}^{\text{sum}} \). For each of such combinations, \( f_{ih}(t) \) can be computed in \( O(\max(T_{ih}(t))) = O(\xi_{ih}^{\text{max}}) \) time, provided that the values of \( f_{ih-1}(t') \) for all \( t' \in T_{ih}(t) \) and \( \Delta_{th} \) are available. Hence the total computation time to compute \( f_{ih}(t) \) for all \( h \) and \( t \) is \( O(\xi_{ih}^{\text{sum}} \xi_{ih}^{\text{max}}) \). The time to compute \( F_i \) is

\[ O(n_{max} + \xi_{ih}^{\text{sum}} \xi_{ih}^{\text{max}} + 1) = O(\pi_{i}^{\text{max}}). \]

The total computation time of the above DP is therefore \( O(n_{max} + \xi_{ih}^{\text{sum}} \xi_{ih}^{\text{max}}) \) for each customer \( i \). In the worst case, we have \( \tilde{c}_{i}^{\text{sum}} = \pi_{i}^{\text{max}} + \xi_{ih}^{\text{max}} \) and \( \xi_{ih}^{\text{max}} = \pi_{i}^{\text{max}} \), but these happen only in pathological situations such that \( v_{ih} \approx u_{ih} \approx \pi_{i}^{\text{max}} \) for all \( h \) and \( b_{ih} - a_{ih} \approx \pi_{i}^{\text{max}} \) for an \( h \), which seem to be too flexible and unrealistic. In practical situations, it is natural to have two consecutive delivery-day periods almost separated (i.e., \( v_{ih-1} - c < a_{ih} \) holds for a small constant \( c \) for \( h = 2, 3, \ldots, \pi_{i}^{\text{max}} \)) and to have limited flexibility between two consecutive delivery days (i.e., \( b_{ih} - a_{ih} \) is a small constant). In such cases, we have \( \tilde{c}_{i}^{\text{sum}} = O(\pi_{i}^{\text{max}} + \pi_{i}^{\text{max}}) = O(\pi_{i}^{\text{max}}) \) and \( \xi_{ih} = O(1) \), and the above computation time for computing \( f_{ih}(t) \) for all \( h \) and \( t \) becomes \( O(\xi_{ih}^{\text{sum}} \pi_{i}^{\text{max}}) = O(\pi_{i}^{\text{max}}) \).

### 3.1.2. Reoptimize all customers (reOptAll)

This is an operation that applies reOptOne to all the customers one by one in a random order.

#### 3.2. Kick

Whenever the local search stops at a locally optimal solution, at which no improved solutions are available in its neighborhood, we perturb the solution obtained and apply local search again to the perturbed solution. Such a perturbation is called a kick.

For the kick, we perform a double bridge operation, as illustrated in Figure 2, to a part of the current solution. As a double bridge operation only changes the order of tasks in each route but does not affect the days or the workers that each task is assigned to, we apply the reOptOne operation to reassign several tasks.

First we give a small kick, and if after a kick, the local search cannot find a better solution, we regress to the solution before the kick and perform a bigger kick on it again (unless the previous kick was of the biggest size). Otherwise (if the local search finds a better solution), we move to the new solution and apply to it a kick of the smallest size. The size of a kick is controlled by the number of double bridge and reOptOne operations. We use four sizes, and the smallest kick is to apply a double bridge operation to each of randomly chosen \( \max\{m_{\pi_{i}^{\text{max}}}/5, 1\} \) routes and then to apply reOptOne operations to \( \lfloor n/20 \rfloor \) tasks. In the remaining three bigger kicks, the numbers of the two operations are set to \( \max\{[2m_{\pi_{i}^{\text{max}}}/5, 1], \lfloor n/10 \rfloor \}, \max\{[2m_{\pi_{i}^{\text{max}}}/5, 1], \lfloor n/20 \rfloor \} \) and \( \max\{[3m_{\pi_{i}^{\text{max}}}/5, 1], \lfloor n/4 \rfloor \} \).
The evaluation function $\text{eval}(\sigma)$ for the local search is defined to be the sum of five parts:

$$
\sum_{k \in M} \sum_{t \in T} G_{kt} + \sum_{(i,t) \in U_{\text{set}}} W_d + \alpha \cdot \left[ \sum_{k \in M} \sum_{t \in T} \max \{0, C_{kt} - r_{kt}^{\max} \} \right]
+ \beta \cdot \left[ \sum_{(i,t) \in U_{\text{hard}}} \sum_{h=1}^{h_{\max}} x_{ikt} \right] + \gamma \cdot \left[ \sum_{k \in M} \sum_{t \in T} \max \left\{0, \sum_{i \in N \setminus \{0\}} \sum_{h=1}^{h_{\max}} d_{it} x_{ih} - Q_t \right\} \right]
+ \delta \cdot [\text{total deviation from the allowable delivery intervals}].
$$

The weights $\alpha$, $\beta$, $\gamma$ and $\delta$ are adjusted dynamically. Whenever a local search phase finishes, if a solution that does not violate the constraints was not found during the phase, we increase the weights by 1.2 times; otherwise, we decrease the weights by 0.8 times for those constraints that are not violated by the obtained locally optimal solution. Since these parameters $\alpha, \beta, \gamma$ and $\delta$ are adaptively controlled during iterations, there is no need to manually adjust them for each instance, that is, the only parameters we need to manually adjust are the above mentioned 1.2 and 0.8 to increase/decrease the weights, and the same parameter setting of 1.2 and 0.8 are used for all the instances throughout the computational experiments in this paper. For the initial values of $\alpha, \beta, \gamma$ and $\delta$, we set them all to be equal to 800 for all the instances.

The framework of the algorithm is described in Algorithm 1.

**Algorithm 1** Iterated Local Search Algorithm for PVRP

```
Initialize $\sigma$ by applying insert operations of reOptOne to customers one by one.
Let $\sigma_{\text{current}} := \sigma$ and incVal := $+\infty$.
while stopping condition is not satisfied do
    repeat
        Let $N_{UV}$ be the neighborhood of $\sigma$, that is, the set of all solutions obtainable by applying to $\sigma$ a neighborhood operation of reOptOne, reOptAll, 2-opt, 2-opt*, relocate or cross exchange.
        Let impFound := 0.
        repeat
            Randomly choose a solution $\sigma'$ from $N_{UV}$, and then let $N_{UV} := N_{UV} \setminus \{\sigma'\}$.
            if $\sigma'$ is feasible and its objective function value is less than incVal then
                Let $\sigma_{\text{incumbent}} := \sigma'$ and let incVal be the objective function value of $\sigma'$.
            end if
            if eval($\sigma'$) < eval($\sigma$) then
                Let $\sigma := \sigma'$ and impFound := 1.
            end if
        until impFound $\neq \emptyset$ or impFound = 1.
    until impFound = $\emptyset$.
Update penalty weights.
if eval($\sigma$) $\leq$ eval($\sigma_{\text{current}}$) then
    Let $\sigma_{\text{current}} := \sigma$.
end if
Let $\sigma$ be the solution generated by applying a kick to $\sigma_{\text{current}}$.
```

3.4. Efficient implementations

3.4.1. Neighbor list

We need to evaluate the costs of many neighboring solutions in the local search. Instead of naively evaluating them
all, we create a neighbor list in advance, before the first call to local search, and we use it to reduce the neighborhood size, as described below. The neighbor list stores the $N^{\text{size}} (\leq n)$ customers that are closest to each customer in nondecreasing order of distance. Thus the total neighbor list size is $nN^{\text{size}}$.

3.4.2. Efficient implementation of local search

This efficient implementation is taken from Bentley (1992). To explain it, we use the 2-opt* operation as an example. Other neighborhoods, such as those for 2-opt, relocate, and cross exchange, can also be reduced by similar procedures, so we omit detailed explanations of them.

We focus on four tasks $A_1$, $A_2$, $B_1$, and $B_2$, and consider a 2-opt* operation that removes two edges $e_1 = (A_1, A_2)$ and $e_2 = (B_1, B_2)$ from two distinct routes and reconnects the tasks by two edges $e_3 = (A_1, B_2)$ and $e_4 = (B_1, A_2)$, as illustrated in Figure 3.

Denoting the length of edge $e$ by $l(e)$, this operation generates an improved solution if and only if $l(e_1) + l(e_4) < l(e_3) + l(e_2)$, and a necessary condition for this to happen is $l(e_3) < l(e_1)$ or $l(e_4) < l(e_2)$. The condition $l(e_3) < l(e_1)$ means that task $B_2$ is closer to task $A_1$ than $A_2$. Hence, for task $A_1$, we only need to consider tasks that are closer to it than task $A_2$. We can generate all such tasks by scanning the neighbor list of $A_1$ from the top until we encounter a task whose distance from $A_1$ is not smaller than that of $A_2$ (unless $N^{\text{size}}$ is too small and the last element of the list is reached before we encounter such a task). The case for $l(e_4) < l(e_2)$ is similar. By using the neighbor list, the neighborhood size is decreased from $O(pn^2)$ to $O(nN^{\text{size}})$.

4. Computational results

All algorithms presented in this paper were implemented in C and tested on a computer with a dual-core 64-bit 3.3 GHz Intel Core i5 processor and 8 GB 1867 MHz DDR3 RAM.

4.1. Benchmark instances

We use existing benchmark instances to evaluate the influence of neighbor list size and to conduct comparisons with existing methods. All such instances were taken from neo.lcc.uma.es, which comprise two data sets. The first one contains 32 instances introduced by Eilon et al. (1971) (instances p1–p10), Russell and Igo (1979) (instance p11), Russell and Gribbin (1991) (instances p12 and p13) and Chao et al. (1995) (instances p14–p32). The other one contains 10 instances (instances pr01–pr10) generated by Cordeau et al. (1997). Among these 42 instances, we did not adopt several instances (p02, p03, p05, p08, and p10–p13) as test instances in our computational experiments because our model has a different way of defining the allowable delivery schedules. For example, in instance p02, a number of customers have allowable delivery date combinations $\{1, 3\}$, $\{2, 4\}$, and $\{3, 5\}$ for total of five days, which cannot be treated in our model.

4.2. Evaluation of the influence of neighbor list size

We evaluate how the neighbor list size influences the computational results. All experiments in Section 4.2 were run with time limits of $V/4$ for the values of $V$ in the fifth column of Table 1. The “deviation” in this section means the average percent difference between our result and the best known solution.
Analyzing the differences of results between an implementation without a neighbor list, and those with neighbor list sizes 10, 15, or 20, we observed that most of the best values were obtained by an implementation with a neighbor list. The average deviation for the 34 instances using the algorithm without a neighborhood list was 2.57%, and deviations for the algorithm with neighbor list sizes of 10, 15, and 20 were 2.51%, 2.48%, and 2.49%, respectively.

These results indicate that, if we set $N_{\text{size}}$ to 10, the neighbor list restricts the candidates actually checked in the neighborhoods 2-opt, 2-opt*, cross exchange and relocate exchange in such a way that some improved solutions are overlooked. On the other hand, if we do not use a neighbor list, the time for local search increases, and the overall performance becomes worse. Based on these considerations, we decided to set the neighbor list size to 15 in our algorithm and used this setting throughout the computational experiments in the remainder of this paper.

### 4.3. Evaluation of proposed method

To evaluate our proposed method, we compare the experimental results of our method and those of the variable neighbor search (VNS) algorithm reported by Hemmelmayr et al. (2009). Table 1 shows the comparison. As described in Sections 3.3 and 4.2, the results of our algorithm in the table were obtained using the same parameter settings, which include the parameters for controlling the weights in the penalty function and neighbor list size $N_{\text{size}}$, for all instances.

The first four columns of the table are the descriptions of benchmark instances: instance name, the numbers of customers and vehicles, and planning period. The following two columns are the results of the VNS: column “VNS running time V (s)” shows the average running time of 10 runs in seconds on a PC with a 3.2 GHz CPU, and column “VNS gaps (%)” shows the percentage deviation between the best known solution value reported by Baldacci et al. (2011), Chao et al. (1995), Cordeau et al. (1997), Hemmelmayr et al. (2009), Vidal et al. (2012) and the average solution value of 10 runs of the VNS algorithm with 10$^7$ iterations for each run. The last six columns are the results of our proposed algorithm: each “gap” value is the percentage deviation between the best known solution value and the average solution value of ten runs of our ILS algorithm when the time limit for each run was set to the value in the second row (i.e., $V/8$, $V/4$ or $V/2$ seconds with $V$ the value in the fifth column); columns “iterations” show the iteration count of the outermost loop conducted until the algorithm was terminated, where each value is the average of ten runs.

According to the data from SPEC.org, we infer that our computer is approximately four times faster than their computational environment. Hence, we compare our ILS algorithm with the VNS method by setting the time limit to $V/4$. We also show the results when the running time was set to $V/8$ and $V/2$, respectively, to observe how running time influences the performance of our algorithm considering the fact that various factors, including implementation details and CPU, may affect the running time of algorithms. The numbers in bold indicate that the results of our ILS algorithm are better than or equal to those of the VNS algorithm.

Hemmelmayr et al. (2009) divided the instances into two groups according to average distance (p27–p32 have larger average distances, and the others have smaller average distances), and they tested the VNS algorithm on those instances with two sets of parameters. From the comparison between the sixth column (VNS gaps) and the ninth column (the gaps of our ILS with a time limit of $V/4$) in Table 1, we see that our algorithm performs well on instances that have a small total number of routes, which is given by the product of the number of workers $m$ and days $t$. In contrast, the performance is slightly worse for instances with a large total number of routes. One conceivable reason for this is that our algorithm is designed for PVRP with flexible delivery dates rather than standard PVRP with only several delivery date combinations and, during the local search, our algorithm always searches for a feasible date combination among a large number of possible combinations, most of which are infeasible for the tested instances of the standard PVRP. Nevertheless, the differences in percentage deviation are quite small and are less than 5% in the worst case. These results would indicate that our method is reasonably competitive, even though our method is designed for PVRP with flexible delivery dates and is not specially tailored for the standard PVRP.

The seventh, ninth and eleventh columns (i.e., the gaps of our ILS with time limits of $V/8$, $V/4$ and $V/2$) show that our algorithm obtains better solutions when more computation time is given; however, the differences in the gaps are small, and the above-mentioned conclusion is not much affected by such differences in computation time.

### 4.4. Evaluation of the new model with flexible delivery dates

To evaluate our new model, we generated a set of instances for PVRP with flexible delivery dates based on the existing PVRP benchmark instances pr01–pr10, which were generated by Cordeau et al. (1997).

Our instances use the data of the original instances for the following values: the maximum duration of a route (maximum working time of a worker), the maximum load of a vehicle, the number of customers, geometric locations of
customers, service duration, and demand. Because the planning periods of original instances (4 days or 6 days) only allow customers to have a limited number of frequency patterns, in order to increase the variety of customers, we extended the planning periods of original instances (4 days or 6 days) only allow the variety of customers, we extended the planning periods of original instances (4 days or 6 days) only allow customers to have a limited number of frequency patterns, in order to increase the variety of customers, we extended the planning periods of original instances (4 days or 6 days) only allow customers to have a limited number of frequency patterns.

Table 1: Solution quality of the algorithms VNS and ILS with respect to the deviation of the average solution values from the best known solutions.

| VNS | Instance | n | m | $m^{\text{max}}$ | VNS running time V (s) | VNS gaps(%) | V/8 (s) gaps(%) | V/4$^{\ast}$ (s) gaps(%) | V/2 (s) gaps(%) |
|-----|----------|---|---|-----------------|---------------------|-------------|-----------------|-----------------|-----------------|
|     |          |   |   |                 | 0.00                | 2051.1      | 0.00            | 4115.3          | 0.00            |
| p01 | 50       | 3 | 2 | 98.3            | 1.00                | 1072.4      | 1.10            | 2179.9          | 1.10            |
| p04 | 75       | 5 | 2 | 67.2            | 1.44                | 2.23        | 1.66            | 2.23            | 1.66            |
| p06 | 75       | 1 | 10| 76.0            | 5.59                | 2.53        | 2.04            | 1884.8          | 1.98            |
| p07 | 100      | 4 | 2 | 183.2           | 0.46                | 0.86        | 0.77            | 1320.8          | 0.55            |
| p09 | 100      | 1 | 8 | 193.1           | 1.05                | 0.62        | 0.46            | 2347.1          | 0.41            |
| p14 | 20       | 2 | 4 | 37.4            | 0.00                | 0.00        | 2.04            | 1884.8          | 1.98            |
| p15 | 38       | 2 | 4 | 93.9            | 0.00                | 0.00        | 1884.8          | 1.98            | 1884.8          |
| p16 | 56       | 2 | 4 | 217.7           | 0.00                | 0.00        | 1884.8          | 1.98            | 1884.8          |
| p17 | 40       | 4 | 2 | 56.7            | 0.25                | 0.17        | 0.08            | 1473.7          | 0.00            |
| p18 | 76       | 4 | 4 | 142.5           | 0.53                | 0.39        | 0.35            | 847.5           | 0.29            |
| p19 | 112      | 4 | 4 | 258.3           | 0.35                | 0.21        | 0.21            | 583.6           | 0.21            |
| p20 | 184      | 4 | 4 | 889.1           | 0.00                | 0.00        | 272.3           | 0.00            | 272.3           |
| p21 | 60       | 6 | 4 | 72.5            | 0.45                | 0.74        | 0.63            | 709.9           | 0.51            |
| p22 | 114      | 6 | 4 | 169.6           | 0.58                | 2.23        | 1.32            | 343.9           | 0.94            |
| p23 | 168      | 6 | 4 | 341.4           | 3.37                | 3.66        | 3.56            | 296.6           | 2.89            |
| p24 | 51       | 3 | 6 | 52.2            | 0.46                | 0.29        | 0.18            | 1188.9          | 0.18            |
| p25 | 51       | 3 | 6 | 46.9            | 0.11                | 0.21        | 0.21            | 954.1           | 0.21            |
| p26 | 51       | 3 | 6 | 45.2            | 0.00                | 0.00        | 0.00            | 954.1           | 0.00            |
| p27 | 102      | 6 | 6 | 66.0            | 1.44                | 4.09        | 3.79            | 230.0           | 2.18            |
| p28 | 102      | 6 | 6 | 64.6            | 0.79                | 3.35        | 2.92            | 219.7           | 2.33            |
| p29 | 102      | 6 | 6 | 59.3            | 1.40                | 3.36        | 2.33            | 230.5           | 1.90            |
| p30 | 153      | 9 | 6 | 78.0            | 2.25                | 7.35        | 6.61            | 88.0            | 5.86            |
| p31 | 153      | 9 | 6 | 77.1            | 1.88                | 7.01        | 6.34            | 78.9            | 5.51            |
| p32 | 153      | 9 | 6 | 70.4            | 2.38                | 8.12        | 7.04            | 78.0            | 6.03            |
| pr01 | 48      | 2 | 4 | 180.3           | 0.00                | 0.05        | 1193.6          | 0.00            | 2400.7          |
| pr02 | 96      | 4 | 4 | 283.3           | 0.53                | 2.01        | 1.88            | 561.8           | 1.74            |
| pr03 | 144     | 6 | 4 | 278.2           | 1.71                | 4.41        | 4.28            | 202.3           | 3.80            |
| pr04 | 192     | 4 | 4 | 314.9           | 2.23                | 6.27        | 6.00            | 102.0           | 5.39            |
| pr05 | 240     | 10| 4 | 264.4           | 2.89                | 7.86        | 7.20            | 42.8            | 6.44            |
| pr06 | 288     | 12| 4 | 324.4           | 3.00                | 7.40        | 6.84            | 36.3            | 6.12            |
| pr07 | 72      | 3 | 6 | 246.7           | 0.22                | 0.33        | 258.4           | 0.21            | 513.4           |
| pr08 | 144     | 6 | 6 | 338.6           | 2.09                | 4.99        | 4.66            | 103.4           | 4.39            |
| pr09 | 216     | 6 | 6 | 423.9           | 2.15                | 6.67        | 6.19            | 46.9            | 5.71            |
| pr10 | 288     | 12| 6 | 376.1           | 3.34                | 7.28        | 6.45            | 14.9            | 6.06            |

average 1.26 2.78 2.48 2.14

* The gap is the percentage deviation from the average solution value of ten runs and the value of the best known solution.

* * * When considering the speeds of PCs, V/4 is estimated to be equivalent to the computation time of the VNS.

[DOI: 10.1299/jamdsms.2020.jamdsms0073] © 2020 The Japan Society of Mechanical Engineers
of \( q_i \) for instance d01 (\( t_{\max} = 12 \)) and d10 (\( t_{\max} = 60 \)) are \( [2, 3, 4, 6] \) and \( [3, 4, 5, 6, 10, 12, 15, 20] \), respectively. For each customer \( i \), the value of \( q_i \) is randomly chosen from the range. The values of lower and upper bounds \( a_i \) and \( b_i \) on the interval between two successive deliveries were randomly chosen from \( \{q_i - 1, q_i - 2, \ldots, \max\{q_i - a_{\max}, 1\}\} \) and \( \{q_i + 1, q_i + 2, \ldots, \min\{q_i + b_{\max}, 2q_i - 1\}\} \), respectively, with probability \( p \) and were set \( a_i = b_i = q_i \) with the remaining probability of \( 1 - p \), where \( p \in \{0, 0.25, 0.5, 0.75, 1\} \). \( a_{\max} \) and \( b_{\max} \) (\( a_{\max} = b_{\max} \in \{[q_i/4], [q_i/2]\} \)) are parameters to control the flexibility level of instances and are fixed for each instance.

Table 2 shows the quality of the ILS algorithm on instances with and without flexible delivery dates

| Instance | \( n \) | \( m \) | \( t_{\max} \) | \( p = 0 \) | \( p = 0.25 \) | \( p = 0.5 \) | \( p = 0.75 \) | \( p = 1 \) |
|----------|--------|--------|-------------| -----------|-------------|-------------|-------------|-------------|
| f01      | 48     | 2      | 12          | \([q_i/4] \) | \([q_i/2] \) | -2.47%      | -9.28%      | -10.59%     | -15.36%     |
| f02      | 96     | 4      | 12          | \([q_i/4] \) | \([q_i/2] \) | -2.93%      | -8.52%      | -10.96%     | -13.31%     |
| f03      | 144    | 5      | 24          | \([q_i/4] \) | \([q_i/2] \) | -4.36%      | -8.79%      | -13.19%     | -18.45%     |
| f04      | 192    | 6      | 24          | \([q_i/4] \) | \([q_i/2] \) | -4.72%      | -9.28%      | -14.27%     | -19.62%     |
| f05      | 240    | 6      | 36          | \([q_i/4] \) | \([q_i/2] \) | -5.90%      | -9.19%      | -14.88%     | -19.84%     |
| f06      | 288    | 7      | 36          | \([q_i/4] \) | \([q_i/2] \) | -6.48%      | -10.73%     | -16.67%     | -21.64%     |
| f07      | 72     | 2      | 48          | \([q_i/4] \) | \([q_i/2] \) | -5.72%      | -7.46%      | -13.96%     | -18.35%     |
| f08      | 144    | 3      | 48          | \([q_i/4] \) | \([q_i/2] \) | -6.38%      | -8.23%      | -15.29%     | -20.72%     |
| f09      | 216    | 5      | 60          | \([q_i/4] \) | \([q_i/2] \) | -5.38%      | -11.25%     | -14.92%     | -19.68%     |
| f10      | 288    | 6      | 60          | \([q_i/4] \) | \([q_i/2] \) | -6.42%      | -11.41%     | -18.40%     | -24.53%     |

In Table 2, the first four columns are the descriptions of instances: instance name, the numbers of customers and vehicles, and planning period. The next six columns are the results of our ILS algorithm on instances when the values of parameters \( p \) and \( a_{\max} = b_{\max} \) representing the flexibility level are set to different values. The fifth column “\( p = 0 \)” represents the objective value of the instance without flexibility in delivery dates. The seventh to tenth columns show the results of instances with flexibility in delivery dates with the value of parameter \( p \) chosen from \( \{0.25, 0.5, 0.75, 1\} \) and \( a_{\max} = b_{\max} \) chosen from \( \{[q_i/4], [q_i/2]\} \), where the results are shown in percentage deviation of the costs from the result of instances without flexibility in delivery dates (fifth column). A negative value signifies that a lower objective value is attained compared to the one in the fifth column. All values in the fifth and seventh to tenth columns are the average of ten runs, each run performed under a 60-second time limit, with the same parameter setting for controlling the weights in the penalty function and neighbor list size \( N^{out} \) for all generated instances.

Comparing every two lines vertically, for most of the instances, the change in the flexibility with respect to \( a_{\max} = b_{\max} \) influences the cost only slightly, with fluctuation less than 2% except for a few cases, and no significant influence was observed. Through the comparison among the results in each row, corresponding to a set of instances with the same requests of delivery but with different levels of flexibility, we see that a greater cost reduction was achieved for instances with more customers allowing flexibility on the interval between two successive deliveries.

Additionally, to test the performance of our algorithm for instances in which customers have a few infeasible delivery dates (those days that are totally refused to be visited), we conducted experiments on instances involving some \( i \) and \( t \) with \( w_{it} = +\infty \). Such instances were generated from the instances in the previous experiment (i.e., the instances whose results are shown in Table 2) by modifying the value of \( w_{it} \) to \(+\infty\) for those \( i \) and \( t \) such that \( w_{it} \) is bigger than \( 0.9(r_{it}q_{it}/40) \) except for the last day in every \( q_i \)-day period to make sure that at least one feasible delivery date combination exists for
each customer. For example, consider a customer who requires a delivery in every two days. If the first day and the fourth day are infeasible delivery dates for such a customer, then for the instance without flexibility on delivery dates (i.e., \(a_i = b_i = q_i\) for all \(i\)), it will never be possible to find a feasible solution. To avoid this, for the second day in every two-day period, we keep the value of \(w_{it}\) unchanged.

The results are shown in Table 3. As in Table 2, all results are the average of ten runs, each run performed under a 60-second time limit. Differently from Table 2, the values in the fifth and seventh to tenth columns are the deviations of the costs from the corresponding results in Table 2’s fifth column. We use ‘−’ to show that no feasible solution was obtained within the time limit.

### Table 3 Comparison of the ILS algorithm on instances with and without flexible delivery dates, involving infeasible delivery dates

| Instance | \(n\) | \(m\) | \(p_{\text{max}}\) | \(p = 0\) | \(p = 0.25\) | \(p = 0.5\) | \(p = 0.75\) | \(p = 1\) |
|----------|------|------|-----------------|---------|-------------|-------------|-------------|---------|
| f01      | 48   | 2    | 12              | 0.00%   | −2.47%      | −9.28%      | −10.59%     | −15.36%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f02      | 96   | 4    | 12              | 0.00%   | −2.93%      | −8.52%      | −10.96%     | −13.21%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f03      | 144  | 5    | 24              | 0.00%   | −4.36%      | −8.79%      | −13.19%     | −18.45%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f04      | 192  | 6    | 24              | 0.02%   | −4.11%      | −8.03%      | −13.49%     | −17.15%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f05      | 240  | 6    | 36              | 4.70%   | −4.10%      | −8.86%      | −14.67%     | −19.60%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f06      | 288  | 7    | 36              | 4.26%   | −1.48%      | −7.63%      | −12.55%     | −18.29%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f07      | 72   | 2    | 48              | −       | 0.98%       | −3.69%      | −13.40%     | −18.13%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f08      | 144  | 3    | 48              | −       | −2.09%      | −9.62%      | −14.25%     | −20.06%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f09      | 216  | 5    | 60              | −       | 0.10%       | −8.99%      | −9.52%      | −22.03%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |
| f10      | 288  | 6    | 60              | −       | −          | −9.54%      | −16.33%     | −22.48%  |
|          |      |      |                 | \([q_4/4]\) | \([q_4/2]\) | \([q_4/2]\) | \([q_4/2]\) |          |

Comparing Table 3 to Table 2, we see that for instances f01 to f04, involving several infeasible delivery dates does not influence the performance of our ILS algorithm in solving the problem. For instances f05 and f06, for the case with \(p = 0\) (i.e., when no customer allows flexibility on the interval between two successive deliveries), the results became about 4.5% worse than the results in Table 2, indicating non-negligible influence of infeasible delivery dates, while for the cases with positive \(p\), the maximum difference from the counterpart in Table 2 is less than 2.8%. For the last four instances, for the case with \(p = 0\), no feasible solution was obtained, while for the cases with positive \(p\), our ILS obtained feasible solutions for all cases except two (the last two rows of column ‘\(p = 0.25\)’), and for the cases with \(p > 0.5\), the results are similar to those in Table 2 for most of the instances. Along with the increase of \(p\), the difference between the results in the two tables is getting smaller, indicating that having more customers with flexibility in the interval between two successive delivery dates will reduce the influence of having infeasible customer-date combinations.

### 5. Conclusion and future work

In this paper, we have proposed a new model of periodic vehicle routing problem with flexible delivery dates that can cover a wide range of applications and is expected to address a more diverse set of requests for customer schedules. To solve this problem, we proposed an iterated local search algorithm.

Computational results showed that our proposed method performed competitively in solving the standard PVRP. Furthermore, an efficiency improvement mechanism, called a neighbor list, has also been implemented in this algorithm.
Experiments with varying neighbor list size confirmed that the implementation of a neighbor list improves performance. Additionally, it was confirmed that by allowing flexibility on the delivery dates, the sum of total distribution costs and unwillingness of customers to receive a delivery on an inconvenient day could be reduced. Thus for the cases in which more flexibility is allowed, it would be advantageous to take such flexibilities into consideration for cost saving.

In some applications, it would be more desirable to have the same interval between every two consecutive deliveries or to have deliveries regularly on the same day(s) of the week/month. It could be an interesting future research direction to consider a more general model that can prioritize various factors such as the regularity of deliveries while still allowing this regularity to occasionally be violated for the sake of efficiency.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Numbers JP15H02969 and JP20H02388.

References

Baldacci, R., Bartolini, E., Mingozzi, A. and Valletta, A., An exact algorithm for the period routing problem, Operations Research, Vol.59, No.1 (2011), pp.228–241.

Beltrami, E. J. and Bodin, L. D., Networks and vehicle routing for municipal waste collection, Networks, Vol.4, No.1 (1974), pp.65–94.

Bentley, J. J., Fast algorithms for geometric traveling salesman problems, ORSA Journal on Computing, Vol.4, No.4 (1992), pp.387–411.

Bräysy, O. and Gendreau, M., Vehicle routing problem with time windows, Part I: Route construction and local search algorithms, Transportation Science, Vol.39, No.1 (2005), pp.104–118.

Chao, I.-M., Golden, B. L. and Wasil, E., An improved heuristic for the period vehicle routing problem, Networks, Vol.26, No.1 (1995), pp.25–44.

Cordeau, J.-F., Gendreau, M. and Laporte, G., A tabu search heuristic for periodic and multi-depot vehicle routing problems, Networks, Vol.30, No.2 (1997), pp.105–119.

Eilon, S., Watson-Gandy, C. D. and Christofides, N., Distribution Management: Mathematical Modelling and Practical Analysis, (1971), Griffin, London.

Hemmelmayr, V. C., Doerner, K. F. and Hartl, R. F., A variable neighborhood search heuristic for periodic routing problems, European Journal of Operational Research, Vol.195, No.3 (2009), pp.791–802.

Russell, R. A. and Gribbin, D., A multiphase approach to the period routing problem, Networks, Vol.21, No.7 (1991), pp.747–765.

Russell, R. and Igo, W., An assignment routing problem, Networks, Vol.9, No.1 (1979), pp.1–17.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N. and Rei, W., A hybrid genetic algorithm for multidepot and periodic vehicle routing problems, Operations Research, Vol.60, No.3 (2012), pp.611–624.