Memetic Algorithm with Route Decomposing for Periodic Capacitated Arc Routing Problem

Yuzhou Zhang[a], Yi Mei[b,*], Ke Tang[c], Keqin Jiang[a]

[a]*School of Computer and Information, Anqing Normal University, Anqing 246133, China*
[b]*School of Engineering and Computer Science, Victoria University of Wellington, Kelburn 6012, New Zealand*
[c]*USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China*

Abstract

In this paper, the Periodic Capacitated Arc Routing Problem (PCARP) is investigated. PCARP is an extension of the well-known CARP from a single period to a multi-period horizon. In PCARP, two objectives are to be minimized. One is the number of required vehicles (nv), and the other is the total cost (tc). Due to the multi-period nature, given the same graph or road network, PCARP can have a much larger solution space than the single-period CARP counterpart. Furthermore, PCARP consists of an additional allocation sub-problem (of the days to serve the arcs), which is interdependent with the routing sub-problem. Although some attempts have been made for solving PCARP, more investigations are yet to be done to further improve their performance especially on large-scale problem instances. It has been shown that optimizing nv and tc separately (hierarchically) is a good way of dealing with the two objectives. In this paper, we further improve this strategy and propose a new Route Decomposition (RD) operator thereby. Then, the RD operator is integrated into a Memetic Algorithm (MA) framework for PCARP, in which novel crossover and local search operators are designed accordingly. In addition, to improve the search efficiency, a hybridized initialization is employed to generate an initial population consisting of both heuristic and random individuals. The MA with RD (MARD) was evaluated and compared with the state-of-the-art approaches on two benchmark sets of PCARP instances and a large data set which is based on a real-world road network. The experimental results suggest that MARD outperforms the compared state-of-the-art algorithms, and improves most of the best-known solutions. The advantage of MARD becomes more obvious when the problem size increases. Thus, MARD is particularly effective in solving large-scale PCARP instances. Moreover, the efficacy of the proposed RD operator in MARD has been empirically verified.

Keywords: Periodic capacitated arc routing problem, combinatorial optimization, metaheuristics, memetic algorithms

1. Introduction

The Capacitated Arc Routing Problem (CARP), presented by Golden and Wong [1], has many practical applications such as mail delivery, street sweep, waste collection, multicommodity network design and hub location [2], [3]. It is a classic combinatorial optimization problem, which deals with a connected undirected graph $G(V, E)$ with non-negative costs and demands on edges. All edges with positive demand (*tasks*) of the graph must be served by a

[*]Corresponding author
Email address: yi.mei@ecs.vuw.ac.nz (Yi Mei)

fleet of identical vehicles based at a special vertex called the *depot* vertex. The objective of CARP is to search for a set of routes with minimal total cost, so that each route begins and ends at the depot vertex, each task is served exactly once, and the total demand of the tasks served by each route must not exceed the vehicle capacity.

CARP is NP-hard [1], and exact methods for CARP such as branch-and-bound [4] can only deal with small and medium sized instances [5]. Heuristics and meta-heuristics are promising methods for tackling the real-world scenarios, in which the problem size is usually large. A number of constructive heuristics have been proposed for CARP, such as augment-merge [1], path-scanning [6], Ulusoy's tour splitting method [7], construct-and-strike [8], and augment-insert [9]. However, these heuristics still cannot obtain satisfying results. In contrast, by searching in the solution space, meta-heuristics manage to obtain much better results with more computational effort. These include tabu search [10], [11], [12], variable neighborhood search [13], [14], guided local search [15], [16], memetic algorithm (MA) [5], [17], [18], [19], [20], [21], bi-objective genetic algorithm [22], cooperative co-evolution [23], simulated annealing [24], and ant colony optimization [25], to name a few. A survey on the CARP can be found in [26].

Although numerous researches on the basic single-period CARP model have been carried out, it is not enough for real-world applications such as waste collection. In waste collection, there are various factors contributing to the amount of refuse production on a street, such as the region, population density, the habit, etc. As a result, the refuse accumulates at different rates on different streets, and the frequency of service varies from street to street. For example, the rural areas produce refuse more slowly than the urban and residential areas. Therefore, it would be reasonable to collect the waste for different types of areas with different frequencies, e.g. daily for the urban and residential areas but every two days for the rural areas. This leads to the Periodic CARP (PCARP) variant, which makes a schedule over a multi-period horizon.

In PCARP, assuming that there are np periods in the horizon, the number of services for a task over the entire horizon is called its *service frequency*. The basic CARP constraints (e.g. each route starts and ends at the depot vertex, and the total demand of the tasks in each route cannot exceed vehicle's capacity) must be satisfied. For each task, the service days must belong to some predefined set of service days, and its demand accumulates from its last service day. PCARP is to design an optimal plan that assigns the services of each task to appropriate periods over the horizon, and schedules a set of routes in each period.

Obviously, PCARP reduces to CARP if there is only one period. Since CARP is NP-hard [1], PCARP is NP-hard as well. Moreover, compared to CARP, PCARP has an additional objective besides the original objective (i.e., the total cost tc), which is to minimize the number of required vehicles nv, since the waste management companies attempt to minimize the investment cost (depending on the fleet size) [27]. The additional objective nv affects both the service day allocation (to have balanced number of routes over the horizon) and routing in each period, and imposes extra challenges in PCARP.

As a relatively new extended version of CARP, PCARP attracted much less attention than the traditional model, despite of the fact that it is closer to reality. To the best of our knowledge, there are only a few publications devoted to PCARP so far. In 2002, Lacomme et al. gave the definition of PCARP firstly [27]. Thereafter, Chu et al. presented lower bounds in 2003 [28], a linear programming model in 2004 [29] and several constructive heuristics in 2005 [30] for PCARP. In 2005, Lacomme et al. proposed A Memetic Algorithm (MA) (referred to as LMA hereafter), in which a problem-specific crossover operator (i.e., the Periodic Linear Order Crossover, PLOX) was designed [31]. Three heuristics were presented and embedded in a Scatter Search (SS) by Chu et al. in 2006, and a greedy constructive heuristic (i.e., the Best Insertion Heuristic, BIH) was used to improve the performance of the initial population [32]. In 2009, a new ant colony system (ACS) algorithm with a heuristic for inserting the tasks into the solutions was proposed by Kansou and Yassine [33]. However, as an important objective, nv can hardly be improved by above methods, since it is insensitive to the traditional small-step-size search operators. In 2011, Mei et al. developed a Route-Merging (RM) operator for improving nv particularly. By including the RM operator before the local search process, the MA with RM (MARM) first improves a solution in nv, and in tc by the subsequent local search [34]. In 2015, Laporte et al. investigated the CARP extensions including the PCARP [35]. Other works include [36], [37] and [38].

On the other hand, there have been a number of studies proposed on VRP (e.g. [39], [40], [41], [42], [43]) and Periodic VRP (PVRP) (e.g. [44], [45], [46], [47]), which are the node routing counterparts of CARP and PCARP. However, these works are not directly applicable to PCARP due to following reasons: (a) the objective nv was ignored in PVRP, (b) the instance size will becomes much larger, e.g. a CARP with n tasks can be converted to a VRP with $2n + 1$ vertices [48], and (c) the task demand is constant in VRP, but depends on the gap between two successive services in PCARP.

In literature, it is usually assumed that nv is more important than tc in PCARP. In other words, in PCARP, nv is considered to be the primary objective, and tc is considered to be the secondary objective. Under such a consideration, one can combine these two objectives into a single objective by the weighted sum approach and assign a sufficiently large weight to nv to ensure its priority. A previously designed problem-specific operator called the Route Merging (RM) operator has been demonstrated to be promising by improving nv and tc separately during the optimization. However, the RM operator cannot handle the tradeoff between the objectives well enough. In this paper, to overcome this drawback, we propose a new Route Decomposition (RD) operator. The RD operator improves the objective nv by decomposing routes and the objective tc by moving the tasks to appropriate positions heuristically. In contrast to the RM operator, which considers only nv, the proposed RD operator takes both nv and tc into account while still keeping the priority of nv. This way, the RD operator optimizes both tc and nv simultaneously. More specially, to reduce nv, RD selects one route from the period with the most routes to decompose. Then, to reduce tc, it moves the tasks on the selected route to new positions to minimize the additional tc value. Based on the RD operator, a novel MA with RD (MARD) is proposed. In addition, to improve search efficiency, two constructive heuristics (i.e., BIH and path-scanning) are employed to generate the initial population. The experimental studies demonstrate that MARD can obtain better solutions than the state-of-the-art algorithms on the tested benchmark sets and has a significantly better performance in terms of both nv and tc on large-scale instances.

The remainder of this paper is organized as follows. Section 2 gives the notations and problem model. In Section 3, the relationship between the two objectives is investigated and the RD operator is proposed. Section 4 introduces our new approach MARD in detail, followed by experimental studies in Section 5. Finally, the conclusions are presented in Section 6.

2. Problem Model of PCARP

In PCARP, consider an undirected connected graph $G(V, E)$ and a horizon H of np periods, where V and E are the vertex (e.g. crossroad) and edge (e.g. street) sets. There are m identical vehicles with limited capacity Q at the depot vertex $v_0 \in V$. Each edge $e \in E$ has four non-negative attributes: the demand vector $\vec{d}(e) = (d_1(e), d_2(e), \ldots, d_{np}(e))$, the service frequency $\eta(e)$, the serving cost $sc(e)$ and the deadheading cost $dc(e)$. In $\vec{d}(e)$, $d_p(e)$ indicates the demand of e in period p. The service frequency $\eta(e)$ is the number of required services during horizon. The serving cost $sc(e)$ means the cost induced by serving e, and the deadheading cost $dc(e)$ is the cost for traversing e without service. Each task t in the task set $E_R \subseteq E$ must be served in at least one period. That is, $\eta(t) \geq 1$, and $sc(t) > 0$. Each task t has an allowed period combination set $\mathcal{APC}(t)$, which consists of a number of period combinations. A period combination $PC(t) = \{p_1(t), \dots, p_{\eta(t)}(t)\} \in \mathcal{APC}(t)$ is a set of $\eta(t)$ periods, indicating the periods that t is served. Each task t is assigned two positive integer IDs t_1 and t_2, one for each direction. For ID t_i, let $inv(t_i)$ stand for its reverse direction and they are equivalent in the service frequency, serving cost, deadheading cost and demand, but it is important to note that only one of them can be served in a period. In addition, $head(t_i)$ and $tail(t_i)$ denote the head and tail vertices respectively.

For PCARP, a route R_{pk}, which is traversed by vehicle k in period p, can be represented as a sequence of tasks. That is,

$$R_{pk} = (0, t_{pk1}, t_{pk2}, \dots, t_{pkl_{pk}}, 0) \tag{1}$$

where l_{pk} is the number of tasks in R_{pk}. A dummy loop (v_0, v_0) is created to ensure the route to start and end at the depot and assigned an ID 0, i.e. $tail(0) = head(0) = v_0$. The number of routes in period p is denoted as nr_p. If the task t appears in the sequence of R_{pk} $(k = 1, \dots, nr_p)$, then $p \in PC(t)$. Otherwise, $p \notin PC(t)$. Then, a solution S to PCARP can be represented as a route set, i.e.,

$$S = \{R_{pk} \mid p = 1, 2, \dots, np;\ k = 1, 2, \dots, nr_p\} \tag{2}$$

In PCARP, we assume that the demand of each task accumulates over time in a periodic manner. That is, the remaining demand from the previous period (e.g. week) will be inherited to the next one. Therefore, given a period combination $PC(t)$ of the task t, the steady-state accumulated demand $ad(t, p)$ of task t in period p can be computed as follows:

Step 1. Initialize $i = p$, $ad(t, p) = d_i(t)$;
Step 2. $i \leftarrow i - 1$. If $i = 0$, set $i = np$;
Step 3. If $p_i(t) \notin PC(t)$, $ad(t, p) \leftarrow ad(t, p) + d_i(t)$, and go to Step 2;
Step 4. Return $ad(t, p)$.

The total cost and load of each route R_{pk} can be computed as:

$$cost(R_{pk}) = \sum_{i=1}^{l_{pk}} sc(t_{pki}) + \sum_{i=0}^{l_{pk}} \delta(tail(t_{pki}), head(t_{pk(i+1)})) \tag{3}$$

$$load(R_{pk}) = \sum_{i=1}^{l_{pk}} ad(t_{pki}, p) \tag{4}$$

where $t_{pk0} = t_{pk(l_{pk}+1)} = 0$. The function $\delta(v_i, v_j)$ is the shortest distance (with minimum deadheading cost) from the vertex v_i to v_j, which can be obtained by Dijkstra's algorithm [49]. A vehicle can serve only one route. Hence, for period p, the number of vehicles is equal to the number of routes nr_p.

Then, we can obtain two objectives $tc(S)$ and $nv(S)$ of given solution S to PCARP.

$$tc(S) = \sum_{p=1}^{np} \sum_{k=1}^{nr_p} cost(R_{pk}) \tag{5}$$

$$nv(S) = \max\{nr_p \mid p = 1, \dots, np\} \tag{6}$$

According to the above definitions, the PCARP model can be described as below:

$$\min f(S) = \alpha \cdot nv(S) + tc(S) \tag{7}$$

$$s.t. : count(t_i, p) + count(inv(t_i), p) \leq 1, t_i \in \{\text{IDs of } t\}, t \in E_R, p = 1, 2, ..., np \tag{8}$$

$$\sum_{p=1}^{np}(count(t_i, p) + count(inv(t_i), p)) = \eta(t), t_i \in \{\text{IDs of } t\}, t \in E_R \tag{9}$$

$$PC(t) \in \mathcal{APC}(t), t \in E_R \tag{10}$$

$$load(R_{pk}) \leq Q, p = 1, 2, ..., np, k = 1, 2, ..., nr_p \tag{11}$$

where $count(t_i, p)$ denotes the times that task t appears in the period p at the direction denoted by its ID t_i. In PCARP, nv is considered to be more important than tc. Therefore, the coefficient α in Eq. (7) is set to a sufficiently large value to ensure the priority of nv. Constraint (8) shows that each task t is served no more than once in one period, and its service frequency over the horizon H can be guaranteed by constraint (9). Constraint (10) indicates that the arranged period combination of each task belongs to its allowed period combination set. Constraint (11) shows that the load (total demand of the served tasks) of each route cannot exceed the capacity Q.

For the sake of convenience, all the notations used in this paper are listed in Table 1 for quick reference.

Table 1: Commonly used mathematical notations in this paper

Symbol	Description
$G(V, E)$	An undirected connected graph
V	Vertex set
E	Edge set
H	Time horizon
np	Number of periods
Q	Vehicle capacity
m	Number of vehicles available
v_0	Depot vertex
e	An edge
E_R	Task set
t	A task
$\vec{d}(e)$ ($\vec{d}(t)$)	Demand vector of edge e (task t)
$\eta(e)$ ($\eta(t)$)	Service frequency of edge e (task t)
$sc(e)$ ($sc(t)$)	Serving cost of edge e (task t)
$dc(e)$ ($dc(t)$)	Deadheading cost of edge e (task t)
n	Number of required edges (tasks)
γ	Total number of services over H
$\mathcal{APC}(t)$	Allowed period combination set of task t
$PC(t)$	A period combination of task t represented as a vector
λ_t	Number of the period combinations in $\mathcal{APC}(t)$
$\delta(v_i, v_j)$	The shortest distance from vertex v_i to v_j
S	A solution to PCARP
R_{pk}	A route traversed by vehicle k in period p
l_{pk}	Total number of served tasks in R_{pk}
$ad(t, p)$	Accumulated demand of task t in period p
$cost(R_{pk})$	Cost of route R_{pk}
$load(R_{pk})$	Load of route R_{pk}
nr_p	Number of routes in period p
$nv(S)$	Number of vehicles used in solution S over horizon H
$tc(S)$	Total cost of solution S
α	Coefficient of $nv(S)$ in the objective function
$f(S)$	Objective function

3. Route Decomposing Operator for PCARP

Based on the existing methods for CARP, researchers have presented modified algorithms for PCARP, such as LMA and SS. To solve PCARP, these algorithms extended the solution representation and search operators that were designed for CARP. However, the modified approaches can hardly deal with the objective nv of PCARP well, because they mainly focus on minimizing tc alone. For LMA and SS, the improvement of objectives largely depends on the crossover and local search operators. Analysis in [34] has shown that it is hard to get high-quality solutions with smaller nv's from their parents by crossover and traditional local search with small-step operators that move one or two tasks. To address this issue, a Route-Merging (RM) operator [34] was proposed for directly reducing nv. The RM operator keeps merging two routes in the period with the maximum number of routes until the number of routes is reduced to the pre-specified lower bound. It has been shown that the RM operator can help the search achieve a significantly smaller nv value. However, it ignores the other objective (tc) when merging the routes, and may result in a much higher tc value that makes the subsequent search less efficient.

Based on the above discussions, we aim to consider tc while reducing nv. Motivated by this, the Route Decomposition (RD) operator is developed. The basic idea is to select a route from the period with the maximal number of routes, and decompose it. During the decomposition, each task in the selected route is moved to a position in another route that leads to the minimal additional cost. This way, the RD operator not only improves nv, but also makes effort to reduce tc.

3.1. The Route Decomposing Operator

The pseudo code of the RD operator is given in Algorithm 1 . First, it examines whether there are periods with excessive number of routes, which is larger than the lower bound $lbnv$ (lines 2–4). The lower bound is calculated as follows:

$$lbnv = \left\lceil \frac{1}{np_w \cdot Q} \sum_{t \in E_R} \sum_{p=1}^{np} d_p(t) \right\rceil \tag{12}$$

where np_w is the number of working periods.

If such periods exist, then the RD operator iteratively selects the period p_{sel} in which there are the maximum number of routes and decomposes a route in it (line 5). If there is no infeasible route in the selected period, the (feasible) route with the lowest load is selected to be decomposed. Otherwise, the two routes in p_{sel} with the lowest and highest loads are randomly selected (lines 6–10). In the former case, decomposing the feasible route with the lowest load is expected to reduce the number of routes in p_{sel} with the least constraint violation. In the latter case, decomposing the infeasible route with the highest load tends to reduce the constraint violation the most. Finally, the selected route is removed and its tasks are inserted to other places by the function `move()` (lines 11–14). The movement of a single task will be described in the next section in detail.

Algorithm 1 The Route-Decomposing (RD) operator

```
Input: A solution $S$, $lbnv$;
Output: A new solution $S'$;
 1: $S' \leftarrow S$;
 2: if $\max\{nr_p\} \le lbnv$ then
 3:     return $S'$;
 4: end if
 5: $p_{sel} \leftarrow \arg\max\{nr_p \mid p = 1, \dots, np\}$;
 6: if there is no infeasible route in $p_{sel}$ then
 7:     $R_{dec} \leftarrow$ the route in $p_{sel}$ with the lowest load;
 8: else
 9:     Randomly select $R_{dec}$ between the routes in $p_{sel}$ with the lowest and highest loads;
10: end if
11: Remove $R_{dec}$ from $S'$;
12: for each task $t \in R_{dec}$ do
13:     $S' \leftarrow$ move($S', t, p_{sel}$);
14: end for
15: return $S'$;
```

3.2. Movement of a Task

The movement of a task t of solution S works is described in Algorithm 2. First, a new period combination $PC_{new}(t) \in \mathcal{APC}(t) \setminus \{PC_{curr}(t)\}$ is found by the function `selectPeriodCombination()` (line 2), which will be described in detail next. Then, all the services of t are removed from the periods that do not belong to $PC_{new}(t)$. Finally, a service of t is inserted into each period $p \in PC_{new}(t) \setminus PC_{curr}(t)$, in which the service of t is missed (lines 4–26). Here, the service insertion of t are also done in p_{sel} if $p_{sel} \in PC_{new}(t) \cap PC_{curr}(t)$, because it has been removed from p_{sel} by Algorithm 1. For inserting t into one period, all the routes and positions are examined (including an empty route) (lines 11–18), and the best *feasible* route r^* and position i^* that yield the least additional cost is selected to insert the service (lines 20–24). For the sake of convenience, such an insertion strategy is called the *best insertion* strategy hereafter.

Algorithm 2 `move`(S, t, p_{sel})

```
Input: A solution $S$ and a task $t$;
Output: A new solution $S'$;
 1: $S' \leftarrow S$;
 2: $PC_{new}(t) \leftarrow$ selectPeriodCombination$(S, t, lbnv)$;
 3: Remove all the services of $t$ from the periods that do not belong to $PC_{new}(t)$;
 4: for each period $p \in PC_{new}(t)$ do
 5:   if $(p \in PC_{new}(t) \setminus PC_{curr}(t))$ or $(p = p_{sel}$ and $p_{sel} \in PC_{new}(t) \cap PC_{curr}(t))$ then
 6:     $r^* \leftarrow 0, first\_position \leftarrow true$;
 7:     for $k = 1 \rightarrow nr_p$ do
 8:       if $ad(t,p) + load(R_{pk}) > Q$ then
 9:         continue;
10:       end if
11:       for $i = 2 \rightarrow l_{pk}$ do
12:         if $first\_position = true$ then
13:           $r^* \leftarrow r$, $i^* \leftarrow i$, $\Delta_{cost}(t,p) \leftarrow \delta(t_{pk(i-1)}, head(t)) + \delta(tail(t), t_{pki}) - \delta(t_{pk(i-1)}, t_{pki})$;
14:           $first\_position \leftarrow false$;
15:         else if $\delta(t_{pk(i-1)}, head(t)) + \delta(tail(t), t_{pki}) - \delta(t_{pk(i-1)}, t_{pki}) < \Delta_{cost}(t,p)$ then
16:           $r^* \leftarrow r$, $i^* \leftarrow i$, $\Delta_{cost}(t,p) \leftarrow \delta(t_{pk(i-1)}, head(t)) + \delta(tail(t), t_{pki}) - \delta(t_{pk(i-1)}, t_{pki})$;
17:         end if
18:       end for
19:     end for
20:     if $r^* = 0$ then
21:       Create a new route $(0, t, 0)$ and add into period $p$ of $S'$;
22:     else
23:       Insert a service of $t$ into the position $i^*$ of the route $r^*$ in period $p$ of $S'$;
24:     end if
25:   end if
26: end for
27: return $S'$;
```

In the function `selectPeriodCombination()`, a good new period combination should have the following three properties: (1) it overlaps with the current period combination $PC_{curr}(t)$ well, (2) the additional cost caused by moving the services from the current period combination to the new one (lines 4–26 in Algorithm 2) is small, and (3) the maximal number of routes in the periods that the services are to be inserted into, if larger than $lbnv$, is small. The first property reduces the number of service removal and insertion operations. The second property aims to minimize the cost. The third property tends to reduce the maximal number of vehicles over the horizon.

Based on the above intuition, we define a utility function $U(PC(t))$ for each $PC(t) \in \mathcal{APC}(t) \setminus \{PC_{curr}(t)\}$ as follows:

$$U(PC(t)) = \frac{C - \Delta_{cost}(PC(t))}{(\eta(t) - C_1(PC(t))) \cdot C_2(PC(t))} \tag{13}$$

where $\Delta_{cost}(PC(t))$ indicates the total additional cost caused by moving all the tasks from $PC_{curr}(t)$ to $PC(t)$ (inserted by the best insertion). C is a large constant (3000 in our experiments), which can ensure that $C - \Delta_{cost}(PC(t)) \geq 0$. $C_1(PC(t)) = |PC(t) \cap PC_{curr}(t)|$ is the number of common periods shared by $PC(t)$ and $PC_{curr}(t)$. $C_2(PC(t))$ is defined as follows:

$$C_2(PC(t)) = \max\{\max\{nr_p | p \in PC(t) \setminus PC_{curr}(t)\} - lbnv + 1, 1\} \tag{14}$$

From Eq. (13), we can see that a $PC(t)$ with a smaller additional cost, more periods shared with $PC_{curr}(t)$ or smaller number of routes, tends to have a larger utility function value. Thus, the $PC(t)$ with the largest utility function value is selected as the new period combination. The pseudo code of `selectPeriodCombination`() is given in Algorithm 3.

Algorithm 3 $PC_{new}(t) \leftarrow$ `selectPeriodCombination`$(S, t, lbnv)$

```
Input: A solution $S$, a task $t$, the lower bound $lbnv$;
Output: A new period combination $PC_{new}(t)$;
1: Obtain the current period combination $PC_{curr}(t)$ of $t$ from $S$;
2: if $\lambda_t = 1$ then
3:     return $PC_{new}(t) \leftarrow PC_{curr}(t)$;
4: end if
5: for $PC(t) \in \mathcal{APC}(t) \setminus \{PC_{curr}(t)\}$ do
6:     Compute the utility function $U(PC(t))$ by Eq. (13);
7: end for
8: return $PC_{new}(t) \leftarrow \arg\max\{U(PC(t)) | PC(t) \in \mathcal{APC}(t) \setminus \{PC_{curr}(t)\}$;
```

It should be noted that the capacity constraints may be violated during the route decomposition and infeasible solutions may be obtained by the RD operator. From the description of the function `move`(), it can be seen that the services of task t are not moved in the periods of $PC_{new}(t) \cap PC_{curr}(t) \setminus \{p_{sel}\}$. In each of these periods, the accumulated demand of t is updated according to $PC_{new}(t)$, and the load of the route containing t will change. If the accumulated demand of t increases, then the capacity constraints may be violated. Given the solution S, the *total overload* $tol(S)$ reflects the degree of constraint violation of S, and is calculated as follows:

$$tol(S) = \sum_{p=1}^{np} \sum_{k=1}^{nr_p} \max\{load(R_{pk}) - Q, 0\} \tag{15}$$

The task movement is similar to the λ-interchange operator proposed for VRP [40], which divides all the customers of a given solution into clusters and exchanges subsets between two given clusters. The size of the exchanged subsets cannot be larger than the parameter λ. The task movement proposed in this paper is similar to the λ-interchange operator, where one task is moved from one cluster to another, i.e. $(\lambda_1, \lambda_2) = (1, 0)$. However, it is more complex than the λ-interchange operator, since it may move multiple services of one task to different periods.

4. Memetic Algorithm with Route-Decomposing

Obviously, PCARP is a kind of permutation-based optimization problems for which lots of approaches have been presented(e.g. [50], [51], [52], [53], [54]). As a relatively new population-based meta-heuristic approach, MA was firstly introduced by Moscato in 1989 [55] and effective for the permutation-based optimization problems ([56]). It was inspired by Darwinian principles of natural evolution and Dawkins' notion of memes[57], and can be regarded as a kind of hybrid genetic algorithm (GA) with local search strategies. Due to its powerful ability of searching in combinatorial solution spaces, MA has shown to be successful in many fields (e.g. [18], [19], [20], [21], [31], [34], [58], [59], [60], [61], [62]).

The proposed MARD for PCARP uses the framework of MA and focuses on three aspects. First, the RD operator is embedded between the crossover and local search operators. Second, in the initialization stage, a few high-quality initial individuals are included to help improve the convergence of population in the subsequent search process. Last but not least, during the local search, a novel operator is constructed to select the period combination of one task. In the following, we will describe the framework of MARD and each component in detail.

4.1. Framework of MARD

The framework of MARD for PCARP is given in Algorithm 4. Firstly, a population *pop* with *psize* nonclone individuals is initialized. Then, in each generation, two parent solutions P_1 and P_2 are selected from *pop* randomly, and the Periodic Period-Based Crossover (PPBX) operator is applied to them to produce two offsprings C_1 and C_2 (line 6). Then, the RD operator is applied to both C_1 and C_2 to generate C_1' and C_2', respectively (lines 7 and 8). After that, C_1' and C_2' are further exploited by local search with a certain probability (line 10). As a result, two new solutions NS_1 and NS_2 are generated. Then, each of them is compared with the existing individuals in the current population, and inserted into the population if it is not a clone (lines 11–16). After each insertion, the population is sorted by the stochastic ranking procedure [63], and the worst individual is removed to keep the population size consistent. The stochastic ranking procedure sorts a list of solutions through a bubble-sort-like procedure. If the compared solutions are both feasible, then they are compared by $fitness$. Otherwise, they are compared by $fitness$ or constraint violation, with a certain probability respectively. After the stochastic ranking procedure, the solution with the highest ranks is considered as the worst solution.

Algorithm 4 The framework of MARD

```
Input: A PCARP instance, $psize$, $G_{\max}$, $P_{ls}$ ;
Output: A best feasible solution $BFS$ to PCARP;
 1: Initialize a population pop with psize nonclone individuals;
 2: Set $g = 0$;
 3: while $g < G_{\max}$ do
 4:    $g \leftarrow g + 1$;
 5:    Randomly select two parent solutions $P_1$ and $P_2$ from the population pop;
 6:    $(C_1, C_2) = \mathrm{PPBX}(P_1, P_2)$;
 7:    $C_1' = \mathrm{RD}(C_1, lbnv)$;
 8:    $C_2' = \mathrm{RD}(C_2, lbnv)$;
 9:    Set $NS_1 = C_1'$ and $NS_2 = C_2'$;
10:    Apply local search to $NS_1$ and $NS_2$ with a probability $P_{ls}$;
11:    for $i = 1 \to 2$ do
12:       if $NS_i$ is nonclone solution then
13:          Insert $NS_i$ into pop;
14:          Remove the worst solutions from pop by using stochastic ranking procedure;
15:       end if
16:    end for
17: end while
18: return the best feasible solution $BFS$ in pop;
```

4.2. Solution Representation and Evaluation

Like MARM([34]), we use an explicit task encoding scheme that represents a route as Eq. (1), and a solution as Eq. (2). Fig. 1 shows a simple example, in which the tasks are represented by the solid lines, and the deadheading paths are represented by the dashed lines. As mentioned before, each task is assigned two positive integer IDs, one for each direction. Then, Fig. 1 shows a PCARP solution: Period 1 $((0,1,2,3,0),(0,4,5,0))$ and Period 2 $(0,11,10,6,0)$. Given a solution S, the objectives *tc* and *nv* can be calculated according to Eq. (5) and (6) respectively, and $fitness$ is computed by Eq. (7).

Due to the explicit task encoding scheme employed in this paper, the complexity of the $fitness$ evaluation is $O(\sum_{t \in E_R} \eta(t)) = \mathrm{O}(\gamma)$, where γ is the total number of services over the horizon.

4.3. Population Initialization

To ensure the diversity of *pop*, clones are not allowed in *pop* throughout the entire search process. Starting from an empty *pop*, the initial individuals are generated and added to *pop* one by one. Since the full verification

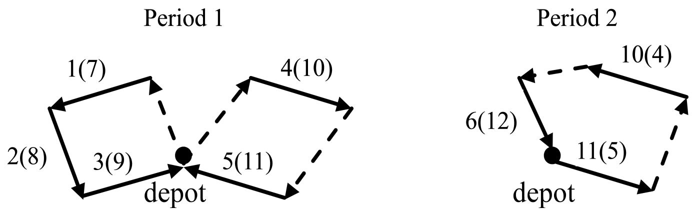

Fig.1. An simple example of the explicit task encoding scheme for PCARP solutions.

of clone individual solutions in PCARP is computationally expensive, an efficient approximate scheme is applied to two solutions S_1 and S_2, which only compares their tc, nv and tol values. S_1 and S_2 are considered to be clone to each other if they have the same values in all the three properties.

It has been showed that containing a few high-quality individuals in the initial population can help improve the convergence of the population, and this initializing scheme was used by most approaches for PCARP (e.g. [31], [32], [34]). Thus, we adopt two heuristics to produce a few good solutions during initialization. To this end, the Path Scanning(PS) [6] and Best Insertion Heuristic(BIH) [30] are adopted and extended to fit the multi-period scenario. The extended heuristics are called EPS and EBIH respectively. The extension process is described as follows:

Step 1. Create np sub-lists of tasks sl_1, sl_2, . . . , sl_{np} (one for each period), and set them to empty,

Step 2. For each task $t \in E_R$, select a period combination $PC(t)$ from $\mathcal{APC}(t)$ randomly;

Step 3. For each task $t \in E_R$, add t to corresponding period's sub-list based on the selected $PC(t)$;

Step 4. For each sub-list sl_p, apply the single-period heuristic (PS or BIH) to generate a single-period sub-solution;

Step 5. Integrate all the sub-solutions obtained by Step 4 to form a complete PCARP solution.

Due to the randomness involved in PS and BIH, one can generate multiple unique solutions by the same heuristic. As a result, in the initial population, 1/6 individuals are generated by EPS, 2/3 individuals are generated by EBIH, and 1/6 remaining individuals are generated randomly.

4.4. Crossover Operator

In this paper, a new Periodic Period-Based Crossover (PPBX) operator is developed by modifying the Periodic Linear Order Crossover (PLOX) [31]. It is described in Algorithm 5. In the beginning, r periods are randomly chosen from the horizon. Then, C_1 inherits the routes from P_1 for the selected r periods, and from P_2 for the remaining periods (lines 7–11). After that, the period combination $PC(t)$ of each task t is determined on the basis of the period combination of t in P_1 (lines 12–16). Briefly speaking, the new period combination includes all the periods in which t is served in both P_1 and C_1, and covers the periods in which t is served in both P_2 and C_1 as much as possible. Finally, for all the tasks, the services are removed from the periods that are out of the new period combination, and inserted into the missing periods in the new period combination (lines 30–38). The other offspring solution C_2 can be obtained by switching the roles of P_1 and P_2.

PPBX is similar to the Best-Cost Route Crossover (BCRC) proposed [39] in the sense that it inserts the missing tasks into the best positions. Additionally, it consists of more complex components to deal with the periodic situation, such as period selection and adjustment of period combinations of the involved tasks.

Algorithm 5 The PPBX operator

```
Input: Two parent solutions $P_1$ and $P_2$;
Output: An offspring solution $C_1$;
1: $C_1 \leftarrow \emptyset$;
2: Randomly choose $r \in \{1, \dots, np\}$ periods $H_x = \{p_{x_1}, p_{x_2}, \dots, p_{x_r}\} \subseteq H$ from the horizon $H$;
3: for each task $t \in E_R$ do
4:     Set the served periods $SP(t)$ to $\emptyset$ and period combination $PC(t)$ to that of $t$ in $P_1$;
5: end for
6: for each period $p \in H$ do
7:     if $p \in H_x$ then
8:         Add all the routes in period $p$ of $P_1$ to period $p$ of $C_1$;
9:     else
10:        Add all the routes in period $p$ of $P_2$ to period $p$ of $C_1$;
11:    end if
12:    for each task $t$ served in period $p$ of $C_1$ do
13:        if $t$ is served in period $p$ of $P_1$ then
14:            $SP(t) \leftarrow SP(t) \cup \{p\}$;
15:        end if
16:    end for
17: end for
18: for each period $p \in H$ do
19:    for each task $t$ served in period $p$ of $P_2$ do
20:        if $SP(t) = \emptyset$ then
21:            Set $PC(t)$ to that of $t$ in $P_2$, $SP(t) \leftarrow SP(t) \cup \{p\}$;
22:        else if $p \notin PC(t)$ then
23:            Look for a new $PC'(t) \supseteq SP(t) \cup \{p\}$;
24:            if $PC'(t)$ is found then
25:                $PC(t) \leftarrow PC'(t)$, $SP(t) \leftarrow SP(t) \cup \{p\}$;
26:            end if
27:        end if
28:    end for
29: end for
30: for each task $t \in E_R$ do
31:    for each period $p \in H$ do
32:        if $p \in PC(t)$ and $t$ is not served in period $p$ of $C_1$ then
33:            Insert a service of $t$ into period $p$ of $C_1$ with best insertion;
34:        else if $p \notin PC(t)$ and $t$ is served in period $p$ of $C_1$ then
35:            Remove the service of $t$ from period $p$ of $C_1$;
36:        end if
37:    end for
38: end for
39: return $C_1$;
```

4.5. Local Search

The process of local search is shown in Fig. 2. It consists of two stages. The first stage is a traditional small-step local search with four traditional move operators, i.e. Single Insertion (SI), Double Insertion (DI), Swap and 2-*opt*, along with a novel Replace Period Combination (RPC) operator which is given in Algorithm 6. In Algorithm 6, given a solution S and a task t, each feasible period combination $PC_{new}(t) \in \mathcal{APC}(t) \setminus \{PC_{curr}(t)\}$ is examined, and a corresponding solution S_{new} is generated by removing redundant services and inserting missing services of t (lines 4–10). Then, the best solution S_{best} is obtained to replace the original solution S. In line 11, the solutions are compared in terms of $fitness$ value and constraint violation. That is, a solution S_1 is better than another solution S_2 if 1) $tol(S_1) < tol(S_2)$ or 2) $tol(S_1) = tol(S_2)$ and $f(S_1) < f(S_2)$.

For the RPC operator, all the possible t's are examined, and the best neighbor is selected to be the next solution. Similarly, four other solutions are produced by the SI, DI, Swap and 2-*opt* operators. Then, the first stage returns the best solution S' among the five solutions produced by the five operators.

Algorithm 6 The RPC operator

```
Input: A solution $S$, a task $t$;
Output: A new solution $S'$;
 1: Obtain the current period combination $PC_{curr}(t)$ of $t$ from $S$;
 2: $S_{best} \leftarrow S$;
 3: for $PC_{new}(t) \in \mathcal{APC}(t) \setminus \{PC_{curr}(t)\}$ do
 4:     $S_{new} \leftarrow S$;
 5:     for each period $p \in PC_{curr}(t) \setminus PC_{new}(t)$ do
 6:         Remove the service of $t$ from $p$ in $S_{new}$;
 7:     end for
 8:     for each period $p \in PC_{new}(t) \setminus PC_{curr}(t)$ do
 9:         Conduct the best insertion for $t$ into $p$ in $S_{new}$;
10:     end for
11:     if $S_{new}$ is better than $S_{best}$ then
12:         $S_{best} \leftarrow S_{new}$;
13:     end if
14: end for
15: return $S' \leftarrow S_{best}$;
```

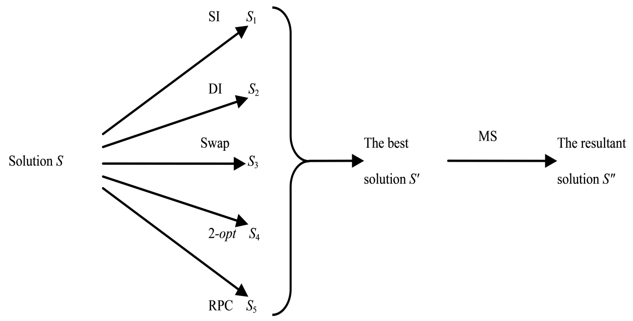

Fig.2. The local search process.

The second stage is an extended local search employing the Merge-Split (MS) operator proposed by Tang et al. [18], in an attempt to further exploit in a larger promising region. Although the MS operator has shown promise in generating high-quality solutions, it suffers from a very high computational complexity. In this paper, we improve the efficiency of the MS operator as follows: First, randomly shuffle the routes. Then, for each period, apply the MS operator to the ith and $(i+1)$th route $(i = 1, \ldots, \lfloor nr_p/2 \rfloor)$. The resultant two routes replace the original ones if they are better. Finally, the resultant solution S'' is returned.

4.6. Summary of MARD

Finally, we summarize the characteristics of MARD by comparing with the state-of-the-art algorithms including MARM [34], LMA [31] and SS [32]. The differences between MARD and other algorithms are as follows:

1) A new Route Decomposition (RD) operator is developed and embedded in MARD to improve both *nv* and *tc* simultaneously. The RD operator selects a route from the period with the maximum number of routes, removes it from the period, and re-inserts the tasks to other periods so that the additional cost is minimized.
2) To benefit the convergence of the search, a hybridized initialization is employed to include high-quality solutions. The PS and BIH constructive heuristics for CARP are extended to PCARP for generating the initial solutions.
3) A new crossover operator called the PPBX operator is designed to combine both the period combination and routing information of the two parents to produce the offsprings.
4) A new local search operator is designed for modifying the period combination of tasks.

5. Experimental Studies

In this section, we evaluate MARD by comparing it with MARM[34], LMA [31] and SS [32] on the *pgdb*, *pval* and *pG* test sets. In addition, the effect of the RD operator is verified by comparing MARD and its counterpart without the RD operator.

5.1. Dataset and Parameter Settings

The *pgdb*, *pval* and *pG* test sets are used in our experiments. The *pgdb* and *pval* instances were generated by Chu *et al.* [32] by extending from the *gdb* and *val* single-period CARP test sets. The *pG* instances were generated by Mei *et al.* [34] from the *G* large-scale CARP test set in the same way.

The *gdb* set contains 23 small size instances with 11 to 55 edges. The *val* set consists of 34 instances based on 10 different graphs with 34 to 97 edges. In each graph, different instances were created by changing the capacity of vehicles. The real-world data set G was built by Brandão and Eglese in [11], which contains 10 large-scale CARP instances with 375 edges based on a road network of the country in Lancashire, U.K. The G set can be divided into two groups based on the required edges, i.e., the G_1 and G_2. Each group has 5 instances (denoted as 1-A $\sim$ 1-E and 2-A $\sim$ 2-E), and different instances in each group were generated by varying vehicle capacities.

For extending a CARP instance to a PCARP instance, we adopt the definition used in previous works [32, 34], which is a weekly horizon H with two idle days (weekend) is defined. For each task t, the demand vector is defined as $\vec{d}(t) = (d_1(t), \dots, d_7(t))$, where all the $d_p(t)$'s equal the demand of t in the CARP instance. The service frequency of t is defined as $\eta(t) = 1 + (v_1(t) + v_2(t)) \mod 5$, where $v_1(t)$ and $v_2(t)$ are the IDs of the two end-vertices of t. If $\eta(t)$ is 2 or 3, then the period combinations with services in consecutive days are forbidden. Otherwise, all the period combinations are allowed. Note that after the conversion, the accumulated demand of a task can exceed the original capacity. To address this issue, in the PCARP instances, the capacity is multiplied by 2, 3, 4 or 5, depending on the instance. Details of the instance generation can be found in [32] and [34]. For the *pG* instances, the 1-D, 1-E, 2-D and 2-E instances are essentially the same as 1-A, 1-B, 2-A and 2-B after the capacity multiplication, and are omitted in the experiments.

In MARD, the population size and local search probability are set to 30 and 0.1, which are the same as MARM. To make a fair comparison, the maximal number of iteration is set to 5000 for the *pgdb* and *pval* sets, and 500 for the *pG* instances. All the compared algorithms are run for 30 times independently. The results of LMA and SS were obtained from the reimplemented version in [34].

5.2. Experimental Results and Analysis

5.2.1. Small and Medium Sized Instances

First, we compare the algorithm on the small and median sized instances. To this end, the *pgdb* and *pval* test sets are used. The results of the average performance are summarized in Table 2, and the detailed average results are shown in Tables 12 and 13 in the Appendix. The results of LMA, SS and MARM are obtained directly from the corresponding literatures. For LMA, three versions (DMA, PMA and IPMA) were presented in [31] and IPMA showed the best performance. Thus, in our comparison, the column LMA stands for the results of IPMA. Note that LMA and SS were run only once in the original literature. On the other hand, MARD and MARM were run for 30 times. With no guarantee on the accuracy of our reimplementation, the best we can do is to consider the results obtained from the single run of LMA and SS as their average performances. Additionally, there is no result available for LMA on the *pval* set from the original literature. Thus the comparison with LMA on the *pval* set is ignored. For each algorithm and each objective (*nv* and *tc*) in Table 2, there are three columns that stand for the mean value over all the instances (denoted as “Mean”), the number of the best mean values among all the compared results (denoted as “NB”) and the average percentage deviation from the corresponding lower bound respectively (denoted as “APD”). APD is calculated as follows:

$$APD = \frac{1}{N}\sum_{i=1}^{N}\frac{f_i - lb_i}{lb_i} \times 100 \tag{16}$$

where N indicates the number of instances, and f_i and lb_i stand for the objective value and lower bound of the ith instance respectively. For nv, the lower bound is obtained from Eq. (12). For tc, the lower bound is obtained from [32]. All of the results can be downloaded from `http://homepages.ecs.vuw.ac.nz/~yimei`.

Table 2: Summary of the average results obtained by the compared algorithms on the *pgdb* and *pval* sets. For each instance, MARM and MARD were run 30 times independently. The results of LMA and SS were obtained from literature, which were obtained by running once using sophisticated parameter settings.

Test set	LMA(IPMA)						SS						MARM						MARD					
	nv			*tc*			*nv*			*tc*			*nv*			*tc*			*nv*			*tc*		
	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD
pgdb	3.13	19	8.7	752.22	0	12.48	3.35	14	13.84	743.35	0	10.73	2.98	23	1.16	711.06	0	6.77	3.2	16	9.94	698.95	23	4.86
pval							4.38	10	25.14	1208.85	0	26.12	3.65	34	1.5	1129.97	0	17.72	3.93	20	11.79	1097.22	34	14.51

From Table 2, it can be seen that MARD performed much better than other compared algorithms overall. It achieved the best mean nv values for 16/23 *pgdb* and 20/34 *pval* instances, which is much better than SS. MARM performed the best, achieving the best mean nv value for all the 23 *pgdb* and 34 *pval* instances. However, in terms of tc, MARD significantly outperformed the other algorithms. It achieved the best mean tc value for all the 23 *pgdb* and 34 *pval* instances, and its APD value is 4.86 and 14.51 for the *pgdb* and *pval* sets respectively, which is much lower than that of the other algorithms. For the *pgdb* test set, MARD outperformed LMA (12.48) by 61.06%, SS (10.73) by 54.71% and MARM (6.77) by 28.21%. For the *pval* test set, MARD outperformed SS (26.12) by 44.45% and MARM (17.72) by 18.12%.

Table 3 shows the t-test results between MARD and all the other algorithms on the *pgdb* and *pval* datasets. For each instance, the 30 outputs of MARD and MARM are assumed to approach the normal distribution, and the single outputs of LMA and SS are considered to be the mean value. Therefore, t-test is considered to be applicable. In the table, W, D and L stand for Win (significantly better), Draw (no significant difference) and Lose (significantly worse), respectively. It is obvious that MARD is significantly better than both LMA and SS on most instances. Compared to both algorithms, MARD obtained the same nv on most instances, and significantly better tc, and thus significantly better $fitness$ value on most instances. Compared to MARM, MARD still obtained the same nv and significantly better tc on a fair number of instances. As a result, MARD performed significantly better than MARM on most instances (34 out of the total 57 instances) in terms of $fitness$. In addition, we conduct the t-tests with Bonferroni correction between MARD and other three compared algorithms simultaneously to see if MARD performed significantly the best, and Bonferroni correction is necessary here to do multiple comparisons. The test results on $fitness$ are given in Table 4, in which “W” indicates the number of instances for which MARD performed significantly better than all the other algorithms. “D” stands for the number of instances for which MARD performed either significantly the same as at least one algorithm, and significantly better than the remaining algorithms. “L” is the number of instances for which MARD performed significantly worse than at least one of the compared algorithms. From the table, it can be seen that MARD achieved significantly better $fitness$ values than all the other compared algorithms on more than half (14 out of 23 *pgdb* and 19 out of 34 *pval*) instances.

Table 3: The results of the t-tests with signficance probability of 95% between MARD and the other compares algorithms on the *pgdb* and *pval* instances.

Test set	MARD and LMA									MARD and SS									MARD and MARM								
	nv			tc			$fitness$			nv			tc			$fitness$			nv			tc			$fitness$		
	W	D	L	W	D	L	W	D	L	W	D	L	W	D	L	W	D	L	W	D	L	W	D	L	W	D	L
pgdb	1	18	4	23	0	0	18	1	4	6	17	0	23	0	0	23	0	0	0	15	8	23	0	0	14	1	8
pval										18	16	0	34	0	0	31	3	0	0	20	14	33	1	0	20	0	14
Overall	1	18	4	23	0	0	18	1	4	24	33	0	57	0	0	54	3	0	0	35	22	56	1	0	34	1	22

Table 4: The results of the t-tests with Bonferroni correction (significance probability of 95%) between MARD and the other compared algorithms on the *pgdb* and *pval* instances.

Test set	$fitness$			Test set	$fitness$		
	W	D	L		W	D	L
pgdb	14	2	7	*pval*	19	1	14

Table 5 shows the summary of the best results obtained by the compared algorithms on the *pgdb* and *pval* test sets, and Tables 14 and 15 show the detailed results. Note that the results of LMA and SS were obtained by a single run with a sophisticated setting, thus we still keep them in the tables for reference.

Table 5: Overall comparison of the best results obtained by the compared algorithms on the test sets *pgdb* and *pval*. MARM and MARD were run for 30 times independently, and LMA and SS were run once using sophisticated parameter settings.

Test set	LMA(IPMA)						SS						MARM						MARD					
	nv			tc			nv			tc			nv			tc			nv			tc		
	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD	Mean	NB	APD
pgdb	3.13	19	8.7	752.22	0	12.48	3.35	14	13.84	743.35	0	10.73	2.96	23	0	696.91	2	4.75	3.13	19	8.7	688.13	23	3.43
pval							4.38	10	25.14	1208.85	0	26.12	3.62	34	0	1106.5	3	15.14	3.74	30	5.88	1083.18	34	12.8

From Table 5, it is clear that MARD performed competitively in terms of both nv and tc in the best case. It achieved the best nv value on 19 *pgdb* and 30 *pval* instances, which is slightly worse than MARM, the same as LMA and much better than SS. In terms of tc, MARD achieved the best results on all the 57 instances, and further updated the best-known results for 19 *pgdb* and 31 *pval* instances (see Tables 14 and 15). On the other hand, the APD values of nv obtained by MARD on test sets *pgdb* and *pval* are 8.7 and 5.88, they are still comparable with those achieved by the other algorithms like the compared algorithms' average performance. However, in terms of tc, the APD value of the best tc obtained by MARD on test set *pgdb* is 3.43, which is better than that of LMA (12.48) by 73.24%, SS (10.73) by 68.03% and MARM (4.75) by 17.03%. For the *pval* test set, the APD value of the best tc obtained by MARD is 12.8, which is better than that of SS (26.12) by 51.00% and MARM (15.14) by 15.46%. Overall, in the best case, MARD outperformed all the other algorithms and the best-known results.

Table 6 shows the average running time (in seconds) of the compared algorithms. Note that the running time depends on various factors such as CPU frequency, RAM, programming language, compiler, etc. It is hardly possible to make a fair comparison between the running times on different platforms. However, we can still get a rough idea by normalizing the running times with respect to the CPU frequency. MARD was run on Intel(R) Core(TM) i3-3240 M 3.40 GHz. MARM was run on Intel(R) Xeon(R) E5335 2.00 GHz. LMA and SS were run on Pentium 4 at 1.40 GHz and 2.40 GHz, respectively. Therefore, the running times of MARM, LMA and SS are multiplied by

2/3.4, 1.4/3.4 and 2.4/3.4, respectively. The table shows that MARD is slower than MARM and SS, while faster than LMA. The high complexity of MARD is mainly due to the RD operator, which enumerates all the possible positions when inserting each task service. In addition, during the local search MARD employs more operators than the other algorithms (e.g. MS and RPC). How to improve the efficiency of MARD by identifying and removing possible redundant computations will be our future direction. Some studies in memetic computing take advantage of the domain knowledge (represented as the so-called memes) learned from solving related problems in the past and/or online learning. For instance, Feng et al. [41] proposed to use the promising solution structure learned from solving CVRP to initialise better solutions for CARP, and vice versa. Chen et al. [64] proposed to use memes storing different local search operators, and developed an algorithm to learn when to use which local search operator during the search process. To improve efficiency, we can employ similar ideas to learn the domain knowledge to focus on the promising regions in the search space.

Table 6: Average running time (in seconds) of the compared algorithms, normalized by CPU frequency. The running times of LMA, SS and MARM are multiplied by 2/3.4, 1.4/3.4 and 2.4/3.4, respectively.

Test set	LMA(IPMA)	SS	MARM	MARD
pgdb	192.2	78.8	7.2	82
pval		312.6	30.9	358.6

Since the RD operator is the main contribution of this work, it is important to verify its efficacy in MARD. To this end, we remove the RD operator (lines 7 and 8 in Algorithm 4) from MARD, and compare this variant, called MA*, against MARD on the *pgdb* and *pval* instances over 30 independent runs. The summarized average results are given in Table 7. From the table, it can be seen that MARD achieved much smaller nv value than MA*, while the tc value is nearly the same. This is consistent with our expectation, which suggests that the RD operator can reduce nv while not much deteriorating tc.

Table 7: Summary of the average results obtained by 30 independent runs of MARD and MA* on the test sets *pgdb* and *pval*.

Test set	MA*(without RD)		MARD	
	nv	tc	nv	tc
pgdb	3.35±0.11	699.79±5.82	3.20±0.10	698.95±5.23
pval	4.42±0.20	1092.92±10.02	3.93±0.12	1097.22±11.42
Overall	3.89±0.16	896.36±7.92	3.56±0.11	898.09±8.33

More specifically, Table 8 gives the numbers of Win, Draw and Lose instances of MARD against MA* when conducting the t-test under the confidence probability of 95%. It can be seen that MARD improved the solutions on the test sets greatly in contrast with MA* in both objectives. In term of nv, MARD is significantly better on 10 *pgdb* and 24 *pval* instances, while never performed significantly worse. In term of tc, MARD and MA* performed statistically the same on 17 *pgdb* and 23 *pval* instances. Note that MARD was significantly outperformed by MA* on 10 *pval* instances. However, MARD achieved significantly better nv values on most of these instances. As a result, MARD still showed significantly better performance than MA* in terms of $fitness$.

5.2.2. Large-Scale Instances

In addition to the small and medium sized instances, we also compared MARD with MARM, LMA and SS on the large *pG* instances to evaluate its scalability. The results were obtained directly from [34]. Note that LMA

Table 8: The statistical test results obtained by 30 independent runs of MARD and MA* on the test sets *pgdb* and *pval*.

Test Set	nv			tc			$fitness$		
	W	D	L	W	D	L	W	D	L
pgdb	10	13	0	4	17	2	11	12	0
pval	24	10	0	1	23	10	24	9	1
Overall	34	23	0	5	40	12	35	21	1

and SS were not tested on the pG set in their original works [31], [32], but reimplemented in [34]. Thanks to the clear description of the algorithms in the original literatures, we believe that the reimplemented version will have statistically comparable behaviour with the original ones. The best and average results (with the t-test under confidence probability of 95%) of the compared algorithms are presented in Tables 9 and 10. The summarized statistical test results are shown in Table 11. Being a new PCARP test set, the lower bounds of tc are unavailable for the pG test set.

Table 9: Best results of the composed algorithms on pG. The results in bold face are best and the new best results achieved by our MARD are showed with "*".

Name	$\lvert V\rvert$	$\lvert E\rvert$	γ	LBV	LMA		SS		MARM		MARD	
					nv	tc	nv	tc	nv	tc	nv	tc
G1-A	255	347	1062	10	11	3623482	12	3513307	**10**	3678504	**10**	**3106420***
G1-B	255	347	1062	12	14	3874996	15	3808003	**13**	3947359	**13**	**3437158***
G1-C	255	347	1062	11	13	3690919	14	3710681	**11**	3799614	12	**3260348***
G2-A	255	375	1138	11	12	3820365	14	3814253	**11**	3923292	**11**	**3383565***
G2-B	255	375	1138	13	15	4118258	16	4108672	**14**	4235707	**14**	**3656182***
G2-C	255	375	1138	11	14	4019023	15	3949113	13	4103535	**12***	**3501008***
Mean				11.3	13.2	3857840.5	14.3	3817338.2	12.0	3948001.8	12.0	3390780.2

From Table 9, MARD is superior to LMA and SS on all the 6 pG instances and obtained the same number of best nv as MARM. In particular, MARD achieved solutions with significantly smaller tc than that of all other composed algorithms on all the 6 instances, Although on instance G1-C, nv obtained by MARD is worse than that of MARM, $fitness$ achieved by MARD is better than that of MARM ($\alpha = 100000$ in the experiment). Thus, MARD achieved the best solutions on all pG instances.

In terms of average performance, Tables 10 and 11 show that MARD showed significantly better performance than both LMA and SS on all the 6 pG instances in terms of nv, tc and $fitness$. When comparing to MARM, MARD performed significantly better on 3 instances in terms of both nv and tc. On the remaining 3 instances, MARD achieved the same nv and significantly better tc, and thus significantly better $fitness$. In addition, we conducted the t-tests with Bonferroni correction between MARD and other three compared algorithms, and the results show that MARD obtained significant better $fitness$ than all the other algorithms on all the pG instances.

In summary, MARD showed significant advantage over all the compared state-of-the-art algorithms on the large-scale pG instances. That is, MARD has a good scalability in terms of solution quality, i.e. its advantage becomes more obvious as the problem size increases. This may be because when the problem becomes larger and more complicated, the solution repairing (reinsertion of missing tasks) needs to be more sophisticated to improve the convergence of the population. In this sense, the RD operator has a strong advantage over the previous operators.

Table 10: Average results obtained by 30 independent runs of the composed algorithms on pG. The results with "†" obtained by one of the compared algorithm are significantly better than that of all the others (with confidence probability of 95%)

Name	LMA		SS		MARM		MARD	
	nv	tc	nv	tc	nv	tc	nv	tc
G1-A	12.1±0.6	3846199.2±204563.0	13.1±0.4	4191076.8±589288.5	11.0±0.4	3758286.1±48305.9	10.9±0.4	3157654.0±37669.3†
G1-B	15.1±0.8	4170835.4±223624.0	16.3±0.6	4319454.0±457226.7	14.0±0.7	4093805.0±67843.8	13.6±0.6	3488414.7±34891.7†
G1-C	13.6±0.6	4034616.1±185334.1	14.8±0.4	4230602.5±593157.0	12.3±0.7	3935364.7±66648.1	12.4±0.5	3315679.9±36691.5†
G2-A	13.5±0.8	4189723.8±233001.0	14.6±0.5	4306027.1±504355.2	12.3±0.5	4051017.5±56278.0	12.0±0.4†	3414931.2±29000.3†
G2-B	16.3±0.8	4515713.9±273187.1	17.6±0.6	4584540.6±511354.6	15.0±0.8	4411075.3±81881.3	14.5±0.5†	3720668.8±40338.2†
G2-C	14.6±0.6	4362438.9±299696.1	15.8±0.5	4493359.9±586055.1	13.4±0.5	4210976.6±55288.2	13.2±0.5	3531334.8±34262.3†
Mean	14.2	4186587.9	15.4	4354176.8	13	4076754.2	12.8	3438113.9

Table 11: The statistical test results obtained by 30 independent runs of MARD and other compared algorithms on the pG set.

Item	MARD and LMA			MARD and SS			MARD and MARM		
	W	D	L	W	D	L	W	D	L
nv	6	0	0	6	0	0	3	3	0
tc	6	0	0	6	0	0	6	0	0
$fitness$	6	0	0	6	0	0	6	0	0

6. Conclusions and Future Work

In this paper, we investigate an extended version of CARP, i.e., PCARP, which is considered over the horizon with multiply periods. PCARP is closer to practical applications (e.g. waste collection) than CARP. In addition, there are two objectives for PCARP, i.e, nv and tc. PCARP is more complex than CARP due to the interaction between the service allocation and routing sub-problems in minimizing both nv and tc.

The difficulty in coping with PCARP is to provide high-quality solutions with both small nv and tc values simultaneously. The existing approaches for PCARP either focus on one of the two objectives or tackle them separately. As a result, their performance are not satisfactory especially when facing the large-scale problems. To overcome these drawbacks, we propose a novel Route Decomposition (RD) operator which can improve nv and tc at the same time. The main idea of the RD operator is to decompose routes of the periods with the maximum number of routes, and reinsert the task services to other periods with possibly fewer routes. Additional cost is considered during the insertion to improve tc as well. We further develop a MARD by embedding the RD operator into a Memetic Algorithm (MA) framework. The experimental studies show that MARD outperformed significantly better than the state-of-the-art algorithms including LMA, SS and MARM, especially on the large-scale pG instances. We also verified the importance of the RD operator in MARD.

In the future, we will take more real-world factors into account, e.g. the time window constraints, time-dependent service costs and multi-depots. In addition, we will improve the efficiency of MARD by identifying and removing redundant computations.

Acknowledgements

This work was partially supported by the Natural Science Key Research Project for Higher Education Institutions of Anhui Province (No. KJ2016A438, No. KJ2014A140, No. KJ2013A177) and the Anhui Provincial Natural Science Foundation (No. 1408085MF131). Yi Mei is supported by the Marsden Funds of New Zealand (VUW1209), and the University Research Funds of Victoria University of Wellington (No. 203936/3337).

References

[1] B. Golden, R. Wong, Capacitated arc routing problems, Networks 11 (3) (1981) 305–315.

[2] M. Yaghini, M. Momeni, M. Sarmadi, A simplex-based simulated annealing algorithm for node-arc capacitated multicommodity network design, Appl. Soft Comput. 12 (2012) 2997–3003.

[3] J. Kratica, M. Milanović, Z. Stanimirović, D. Tošić, An evolutionary-based approach for solving a capacitated hub location problem, Appl. Soft Comput. 11 (2011) 1858–1866.

[4] R. Hirabayashi, Y. Saruwatari, Tour construction algorithm for the capacitated arc routing-problems, Asia-Pacific J. Oper. Res. 9 (2) (1992) 155–175.

[5] P. Lacomme, C. Prins, W. Ramdane-Chérif, Competitive memetic algorithms for arc routing problems, Annals of Operations Research 131 (1–4) (2004) 159–185.

[6] B. Golden, J. DeArmon, E. Baker, Computational experiments with algorithms for a class of routing problems, Comput. Oper. Res. 10 (1) (1983) 47–59.

[7] G. Ulusoy, The fleet size and mix problem for capacitated arc routing, Eur. J. Oper. Res. 22 (3) (1985) 329–337.

[8] W. Pearn, Approximate solutions for the capacitated arc routing problem, Comput. Oper. Res. 16 (6) (1989) 589–600.

[9] W. Pearn, Augment-insert algorithms for the capacitated arc routing problem, Comput. Oper. Res. 18 (2) (1991) 189–198.

[10] A. Hertz, G. Laporte, M. Mittaz, A tabu search heuristic for the capacitated arc routing problem, Oper. Res. 48 (1) (2000) 129–135.

[11] J. Brandão, R. Eglese, A deterministic tabu search algorithm for the capacitated arc routing problem, Comput. Oper. Res. 35 (4) (2008) 1112–1126.

[12] Y. Mei, K. Tang, X. Yao, A global repair operator for capacitated arc routing problem, IEEE Trans. Syst., Man, Cybern. B, Cybern. 39 (3) (2009) 723–734.

[13] A. Hertz, M. Mittaz, A variable neighborhood descent algorithm for the undirected capacitated arc routing problem, Transp. Sci. 35 (4) (2001) 425–434.

[14] M. Polacek, K. Doerner, R. Hartl, et al., A variable neighborhood search for the capacitated arc routing problem with intermediate facilities, J. Heuristics 14 (5) (2008) 405–423.

[15] P. Beullens, L. Muyldermans, D. Cattrysse, et al., A guided local search heuristic for the capacitated arc routing problem, Eur. J. Oper. Res. 147 (3) (2003) 629–643.

[16] P. Repoussis, C. Tarantilis, G. Ioannou, Arc-guided evolutionary algorithm for the vehicle routing problem with time windows, IEEE Trans. Evol. Comput. 13 (3) (2009) 624–647.

[17] Y. Mei, K. Tang, X. Yao, Improved memetic algorithm for capacitated arc routing problem, in: Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC2009), IEEE Press, 18–21 May, 2009, pp. 1699–1706.

[18] K. Tang, Y. Mei, X. Yao, Memetic algorithm with extended neighborhood search for capacitated arc routing problems, IEEE Trans. Evol. Comput. 13 (5) (2009) 1151–1166.

[19] Y. Mei, K. Tang, X. Yao, Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem, IEEE Trans. Evol. Comput. 15 (2) (2011) 151–165.

[20] R. Shang, J. Wang, L. Jiao, et al., An improved decomposition-based memetic algorithm for multi-objective capacitated arc routing problem, Appl. Soft Comput. 19 (2014) 343–361.

[21] Z. Wang, H. Jin, M. Tian, Rank-based memetic algorithm for capacitated arc routing problems, Appl. Soft Comput. 37 (2015) 572–584.

[22] P. Lacomme, C. Prins, M. Sevaux, A genetic algorithm for a bi-objective capacitated arc routing problem, Comput. Oper. Res. 33 (12) (2006) 3473–3493.

[23] Y. Mei, X. Li, X. Yao, Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems, IEEE Trans. Evol. Comput. 18 (3) (2014) 435–449.

[24] R. Eglese, Routing winter gritting vehicles, Discrete Appl. Math. 48 (3) (1994) 231–244.

[25] K. Doerner, R. Hartl, V. Maniezzo, et al., Applying ant colony optimization to the capacitated arc routing problem, in: Proc. Ant Colony Optimization Swarm Intell., Lecture Notes in Computer Science 3172, Springer, 2004, pp. 420–421.

[26] A. Corberán, C. Prins, Recent results on arc routing problems: An annotated bibliography, Networks 56 (1) (2010) 50–69.

[27] P. Lacomme, C. Prins, W. Ramdane-Chérif, Evolutionary algorithms for multiperiod arc routing problems, in: Proc. 9th Int. Conf. Inf. Process. Manage. Uncertainty Knowl.-Based Syst., 2002, pp. 845–852.

[28] F. Chu, N. Labadi, C. Prins, Lower bounds for the periodic capacitated arc routing problem, in: Oral Communication, 2^{nd} International Workshop on Freight Transportation and Logistics (Odysseus 2003), Palermo, Italy, 2003.

[29] F. Chu, N. Labadi, C. Prins, The periodic capacitated arc routing problem: Linear programming model, metaheuristic and lower bounds, J. Syst. Sci. Syst. Eng. 13 (4) (2004) 423–435.

[30] F. Chu, N. Labadi, C. Prins, Heuristics for the periodic capacitated arc routing problem, J. Intell. Manuf. 16 (2) (2005) 243–251.

[31] P. Lacomme, C. Prins, W. Ramdane-Chérif, Evolutionary algorithms for periodic arc routing problems, Eur. J. Oper. Res. 165 (2) (2005) 535–553.

[32] F. Chu, N. Labadi, C. Prins, A scatter search for the periodic capacitated arc routing problem, Eur. J. Oper. Res. 169 (2) (2006) 586–605.

[33] A. Kansou, A. Yassine, Ant colony system for the periodic capacitated arc routing problem, in: Proc. 2009 International Network Optimization Conference, 2009, pp. 1–7.

[34] Y. Mei, K. Tang, X. Yao, A memetic algorithm for periodic capacitated arc routing problem, IEEE Trans. Syst. Man Cybern. B 41 (6) (2011) 1654–1667.

[35] Á. Corberán, G. Laporte, Arc routing: problems, methods, and applications, Vol. 20, SIAM - Society For Industrial And Applied Mathematics, 2015.

[36] I. Monroy, C. Amaya, A. Langevin, The periodic capacitated arc routing problem with irregular services, Discrete Appl. Math. 161 (4) (2013) 691–701.

[37] S. Huang, T. Lin, Using ant colony optimization to solve periodic arc routing problem with refill points, Journal of Industrial and Production Engineering 31 (7) (2014) 441–451.

[38] J.-P. Riquelme-Rodríguez, M. Gamache, A. Langevin, Periodic capacitated arc-routing problem with inventory constraints, J. Oper. Res. Soc. 65 (12) (2014) 1840–1852.

[39] F. Hanshar, B. Ombuki-Berman, Dynamic vehicle routing using genetic algorithms, Appl. Intell. 27 (1) (2007) 89–99.

[40] I. Osman, N. Christofides, Capacitated clustering problems by hybrid simulated annealing and tabu search, Int. T. Oper. Res. 1 (3) (1994) 317–336.

[41] L. Feng, Y. Ong, M. Lim, et al., Memetic search with inter-domain learning: A realization between cvrp and carp, IEEE Trans. Evol. Comput. 19 (5) (2014) 1–15.

[42] J. Luo, X. Li, M. Chen, et al., A novel hybrid shuffled frog leaping algorithm for vehicle routing problem with time windows, Inf. Sci. 316 (2015) 266–292.

[43] L. Tan, F. Lin, H. Wang, Adaptive comprehensive learning bacterial foraging optimization and its application on vehicle routing problem with time windows, Neurocomputing 151 (2015) 1208–1215.

[44] M. Gaudioso, G. Paletta, A heuristic for the periodic vehicle routing problem, Transp. Sci. 26 (2) (1992) 86–92.

[45] E. Angelelli, G. Speranza, The periodic vehicle routing problem with intermediate facilities, Eur. J. Oper. Res. 137 (2) (2002) 233–247.

[46] P. Francis, K. Smilowitz, Modeling techniques for periodic vehicle routing problems, Transp. Res. Part B: Methodol. 40 (10) (2006) 872–884.

[47] F. Alonso, M. Alvarez, J. Beasley, A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions, J. Oper. Res. Soc. 59 (7) (2008) 963–976.

[48] H. Longo, M. de Aragão, E. Uchoa, Solving capacitated arc routing problems using a transformation to the CVRP, Comput. Oper. Res. 33 (6) (2006) 1823–1837.

[49] E. Dijkstra, A note on two problems in connection with graphs, Numer. Math., Vol. 1, 1959, pp. 269–271.

[50] Z. Xiao, Z. Ming, A method of workflow scheduling based on colored petri nets, Data Knowl. Eng. 70 (2) (2011) 230–247.

[51] Z. Chen, M. Qiu, Z. Ming, et al., Clustering scheduling for hardware tasks in reconfigurable computing systems, J. Syst. Architect. 59 (10) (2013) 1424–1432.

[52] J. Li, M. Qiu, J. Niu, et al., Thermal-aware task scheduling in 3d chip multiprocessor with real-time constrained workloads, ACM Trans. Embedded Comput.Syst. 12 (2) (2013) 24:1–24:22.

[53] H. Luo, J. Fang, G. Huang, Real-time scheduling for hybrid flowshop in ubiquitous manufacturing environment, Computers & Industrial Engineering 84 (2015) 12–23.

[54] J. Li, M. Qiu, Z. Ming, et al., Online optimization for scheduling preemptable tasks on iaas cloud systems, Journal of Parallel and Distributed Computing 72 (5) (2012) 666–677.

[55] P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, CalTech, Pasadena, CA, CalTech Concurrent Computation Program Rep. 826, 1989.

[56] Z. Zhu, J. Xiao, J.-Q. Li, et al., Global path planning of wheeled robots using multi-objective memetic algorithms, Integr. Comput. Aided Eng. 22 (4) (2015) 387–404.

[57] R. Dawkins, The Selfish Gene, Oxford, U.K.: Oxford Univ., 1989.

[58] Y. Ong, Artificial intelligence technologies in complex engineering design, Ph.D. dissertation, Sch. Eng. Sci., Univ. Southampton, Southampton, U.K., 2002.

[59] J. Smith, Co-evolving memetic algorithms: A learning approach to robust scalable optimization, in: IEEE Congress on Evolutionary Computation, Vol. 1, IEEE Press, 2003, pp. 498–505.

[60] H. Ishibuchi, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, IEEE Trans. Evol. Comput. 7 (2003) 204–223.

[61] Y. Mei, X. Li, X. Yao, On investigation of interdependence between sub-problems of the travelling thief problem, Soft Comput. 20 (1) (2016) 157–172.

[62] Y. Richard, R. Liu, Z. Jiang, A memetic algorithm for the open capacitated arc routing problem, Transp. Res. Part E 50 (2013) 53–67.

[63] T. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Trans. Evol. Comput. 4 (3) (2000) 284–294.

[64] X. Chen, L. Feng, Y. Ong, A self-adaptive memeplexes robust search scheme for solving stochastic demands vehicle routing problem, Int. J. Syst. Sci. 43 (7) (2012) 1347–1366.

Appendix

In this section, the detailed results obtained by MARD and the other compared algorithms on the test sets are given in Tables 12 – 15. Tables 12 and 13 show the detailed average results, and Tables 14 and 15 show the detailed best results.

In Tables 12 and 13, the columns $|V|$, $|E|$ and γ stand for the number of vertices, edges and services, and LBV and LB indicates the lower bounds of nv and tc, respectively.

In Tables 14 and 15, the achieved best results are compared against the best-known results (obtained from [31], [32], [33], [34]), which is shown in the column BK. If the best-known result is updated, the new best result is marked with "*".

Table 12: The average results over 30 independent runs of MARM and MARD and the results of LMA and SS on the *pgdb* instances. For each instance, the t-tests have been conducted between the compared algorithms under the confidence probability of 95%. If an algorithm is significantly better than all the others, then its corresponding entry is marked with "†".

Name	$\|V\|$	$\|E\|$	γ	LBV	LB	LMA(IPMA)		SS		MARM		MARD	
										Average		Average	
						nv	tc	nv	tc	nv	tc	nv	tc
1	12	44	65	3	771	3	890	3	869	3.0±0.0	827.4±8.6	3.0±0.0	818.3±7.3†
2	12	52	76	3	887	3	1030	3	997	3.0±0.0	934.6±12.7	3.0±0.0	915.2±9.4†
3	12	44	61	3	673	3	794	3	788	3.0±0.0	710.4±9.9	3.0±0.0	698.6±7.3†
4	11	38	52	2	673	2	858	2	795	2.0±0.0	762.1±12.7	2.0±0.0	744.0±5.9†
5	13	52	75	3	965	3	1120	3	1098	3.0±0.0	1032.4±14.2	3.0±0.0	1021.6±8.6†
6	12	44	67	3	861	3	1008	3	951	3.0±0.0	912.2±7.9	3.0±0.0	895.6±6.9†
7	12	44	65	3	801	3	935	3	882	3.0±0.0	836.0±8.1	3.0±0.0	818.6±5.2†
8	27	92	143	5	814	5	1000	6	1014	5.0±0.0	986.0±14.7	5.0±0.2	963.4±13.9†
9	27	102	155	5	772	5	934	6	966	5.0±0.0	917.3±11.5	5.2±0.4	892.0±10.0†
10	12	50	65	2	669	2	744	2	761	2.0±0.0	696.3±9.1	2.0±0.0	684.4±4.5†
11	22	90	133	3	1043	3	1172	3	1193	3.0±0.0	1113.4±11.0	3.0±0.0	1089.7±7.3†
12	13	46	67	3	1021	3	1218	3	1245	3.0±0.0	1149.8±14.5	3.0±0.0	1132.8±12.5†
13	10	56	81	3	1541	3	1597	3	1593	3.0±0.0	1564.9±5.3	3.0±0.0	1556.5±3.5†
14	7	42	64	2	278	3	300	3	285	2.5±0.5†	291.0±8.7	3.0±0.0	282.5±1.1†
15	7	42	64	2	174	2	182	2	180	2.0±0.0	176.2±1.5	2.0±0.0	174.0±0.0†
16	8	56	85	3	352	3	374	3	372	3.0±0.0	364.2±2.1	3.0±0.0	357.8±1.3†
17	8	56	85	2	255	3	267	3	265	2.0±0.0†	266.1±2.0	3.0±0.0	256.8±0.9†
18	9	72	106	2	484	3	506	3	506	2.0±0.0†	494.4±3.3	3.0±0.0	485.2±1.0†
19	8	22	30	2	155	2	181	2	173	2.0±0.0	172.7±0.8	2.0±0.0	171.5±0.8†
20	11	44	63	2	339	3	375	3	356	2.0±0.0†	357.8±4.1	2.8±0.4	348.6±3.3†
21	11	66	101	3	488	3	525	4	509	3.0±0.0	504.9±3.8	3.2±0.4	499.7±4.4†
22	11	88	129	4	578	4	600	5	597	4.0±0.0	593.0±2.7	4.4±0.5	587.6±3.0†
23	11	110	165	5	671	5	691	6	702	5.0±0.0	691.4±3.1	5.9±0.3	681.4±2.2†

Table 13: The average results over 30 independent runs of MARM and MARD and the results of SS on the *pval* instances. For each instance, t-tests have been conducted between the compared algorithms under the confidence probability of 95%. If an algorithm is significantly better than all the others, then its corresponding entry is marked with "†".

Name	$\lvert V\rvert$	$\lvert E\rvert$	γ	LBV	LB	SS		MARM		MARD	
								Average		Average	
						nv	tc	nv	tc	nv	tc
1a	24	78	105	2	434	2	544	2.0±0.0	488.8±6.6	2.0±0.0	480.4±5.1†
1b	24	78	105	3	458	3	585	3.0±0.0	540.0±4.7	3.0±0.0	528.9±4.2†
1c	24	78	105	4	532	5	701	4.0±0.0†	678.5±11.6	4.4±0.5	658.3±11.0†
2a	24	68	94	2	606	2	782	2.0±0.0	706.0±5.0	2.0±0.0	700.4±2.0†
2b	24	68	94	2	695	3	868	2.0±0.0	788.8±6.7	2.0±0.0	781.0±4.1†
2c	24	68	94	4	1012	5	1259	4.0±0.0	1183.3±16.1	4.0±0.0	1174.1±13.3†
3a	24	70	96	2	192	2	249	2.0±0.0	225.9±2.3	2.0±0.0	223.7±2.3
3b	24	70	96	2	212	3	278	2.0±0.0†	263.8±5.0	2.4±0.5	261.3±6.9
3c	24	70	96	4	289	4	358	4.0±0.0	347.1±4.5	4.0±0.0	343.4±4.7†
4a	41	138	205	2	1089	3	1330	2.0±0.0†	1262.4±18.6	2.8±0.4	1204.9±15.0†
4b	41	138	205	3	1113	4	1471	3.0±0.0	1314.7±14.0	3.0±0.0	1289.2±15.1†
4c	41	138	205	4	1205	4	1583	4.0±0.0	1438.7±13.6	4.0±0.0	1410.2±13.9†
4d	41	138	205	6	1502	8	1988	6.1±0.3†	1905.6±31.0	7.0±0.2	1831.2±17.3†
5a	34	130	194	2	1184	3	1454	2.0±0.0†	1353.3±18.8	2.9±0.2	1296.2±15.3†
5b	34	130	194	3	1241	4	1528	3.0±0.0	1417.6±15.6	3.0±0.0	1401.4±13.4†
5c	34	130	194	4	1348	4	1655	4.0±0.0	1541.9±9.6	4.0±0.0	1516.5±11.9†
5d	34	130	194	6	1638	7	2097	6.0±0.0†	2033.3±21.9	6.6±0.5	1980.5±37.5†
6a	31	100	150	2	658	3	754	2.0±0.0†	741.4±8.5	2.4±0.5	731.6±10.6†
6b	31	100	150	3	684	4	863	3.0±0.0	786.4±7.5	3.0±0.0	769.4±8.4†
6c	31	100	150	7	921	8	1183	7.0±0.0	1139.0±11.3	7.0±0.0	1119.7±10.7†
7a	40	132	201	2	833	3	1040	2.0±0.2†	997.7±24.4	3.0±0.0	924.1±7.5†
7b	40	132	201	3	835	4	1046	3.0±0.0†	978.5±11.2	3.4±0.5	955.4±18.3†
7c	40	132	201	7	949	8	1283	7.0±0.0	1190.8±14.5	7.0±0.0	1153.2±10.6†
8a	30	126	194	2	1150	3	1343	2.7±0.5†	1282.3±47.1	3.0±0.0	1226.6±8.8†
8b	30	126	194	3	1197	4	1453	3.0±0.0	1330.4±14.8	3.0±0.0	1295.6±12.8†
8c	30	126	194	7	1567	7	1970	7.0±0.0	1886.0±14.6	7.0±0.0	1852.2±16.8†
9a	50	184	274	2	867	3	1083	2.0±0.2†	990.0±13.4	3.0±0.2	931.4±8.8†
9b	50	184	274	3	891	4	1087	3.0±0.0	1014.0±9.6	3.0±0.0	970.5±10.4†
9c	50	184	274	4	913	4	1195	4.0±0.0	1050.9±12.3	4.0±0.0	1007.0±12.3†
9d	50	184	274	7	1064	8	1418	7.0±0.0†	1367.2±23.4	7.9±0.3	1294.6±13.6†
10a	50	194	300	2	1247	3	1478	2.3±0.4†	1401.8±31.6	3.0±0.0	1347.8±9.8†
10b	50	194	300	3	1259	4	1580	3.0±0.0	1415.9±11.2	3.0±0.0	1387.5±10.4†
10c	50	194	300	4	1301	4	1613	4.0±0.0	1477.1±11.0	4.0±0.0	1442.8±11.2†
10d	50	194	300	7	1551	9	1964	7.0±0.0†	1879.8±21.0	7.9±0.3	1814.4±14.2†

Table 14: The best results of the compared algorithms on 23 *pgdb* instances. For MARM and MARD, "Best" stands for the best results obtained by 30 runs, while for LMA and SS, the results were obtained by one run using sophisticated parameter settings. For each instance, the best results among all the compared results are marked in bold, and the new best results obtained by MARD are marked with "*".

Name	LBV	LB	LMA(IPMA)		SS		MARM		MARD		BK	
							Best		Best			
			nv	*tc*	*nv*	*tc*	*nv*	*tc*	*nv*	*tc*	*nv*	*tc*
1	3	771	**3**	890	**3**	869	**3**	810	**3**	**799***	**3**	810
2	3	887	**3**	1030	**3**	997	**3**	917	**3**	**899***	**3**	917
3	3	673	**3**	794	**3**	788	**3**	691	**3**	**687***	**3**	691
4	2	673	**2**	858	**2**	795	**2**	740	**2**	**729***	**2**	740
5	3	965	**3**	1120	**3**	1098	**3**	1004	**3**	**1003***	**3**	1004
6	3	861	**3**	1008	**3**	951	**3**	900	**3**	**881***	**3**	900
7	3	801	**3**	935	**3**	882	**3**	819	**3**	**813***	**3**	819
8	5	814	**5**	1000	6	1014	**5**	953	**5**	**928***	**5**	953
9	5	772	**5**	934	6	966	**5**	892	**5**	**864***	**5**	892
10	2	669	**2**	744	**2**	761	**2**	677	**2**	**675***	**2**	677
11	3	1043	**3**	1172	**3**	1193	**3**	1089	**3**	**1075***	**3**	1089
12	3	1021	**3**	1218	**3**	1245	**3**	1118	**3**	**1104***	**3**	1118
13	3	1541	**3**	1597	**3**	1593	**3**	1555	**3**	**1551***	**3**	1555
14	2	278	3	300	3	285	**2**	290	3	**280***	**2**	290
15	2	174	**2**	182	**2**	180	**2**	**174**	**2**	**174**	**2**	**174**
16	3	352	**3**	374	**3**	372	**3**	360	**3**	**356***	**3**	360
17	2	255	3	267	3	265	**2**	261	3	**255**	**2**	261
18	2	484	3	506	3	506	**2**	487	3	**484**	**2**	487
19	2	155	**2**	181	**2**	173	**2**	**171**	**2**	**171**	**2**	**171**
20	2	339	3	375	3	356	**2**	348	3	**345***	**2**	348
21	3	488	**3**	525	4	509	**3**	498	**3**	**496***	**3**	498
22	4	578	**4**	600	5	597	**4**	589	**4**	**583***	**4**	589
23	5	671	**5**	691	6	702	**5**	686	**5**	**675***	**5**	686

Table 15: The best results of the compared algorithms on 34 *pval* instances. For MARM and MARD, "Best" stands for the best results obtained by 30 runs, while for LMA and SS, the results were obtained by one run using sophisticated parameter settings. For each instance, the best results among all the compared results are marked in bold, and the new best results obtained by MARD are marked with "*".

Name	LBV	LB	SS		MARM		MARD		BK	
					Best		Best			
			nv	*tc*	*nv*	*tc*	*nv*	*tc*	*nv*	*tc*
1a	2	434	**2**	544	**2**	470	**2**	**466***	**2**	470
1b	3	458	**3**	585	**3**	530	**3**	**518***	**3**	530
1c	4	532	5	701	**4**	653	**4**	**646***	**4**	653
2a	2	606	**2**	782	**2**	**697**	**2**	**697**	**2**	**697**
2b	2	695	3	868	**2**	775	**2**	**773***	**2**	775
2c	4	1012	5	1259	**4**	1149	**4**	**1140***	**4**	1149
3a	2	192	**2**	249	**2**	222	**2**	**219***	**2**	222
3b	2	212	3	278	**2**	255	**2**	**249***	**2**	255
3c	4	289	**4**	358	**4**	336	**4**	**333***	**4**	336
4a	2	1089	3	1330	**2**	1228	**2**	**1201***	**2**	1228
4b	3	1113	4	1471	**3**	1288	**3**	**1265***	**3**	1288
4c	4	1205	**4**	1583	**4**	1409	**4**	**1382***	**4**	1409
4d	6	1502	8	1988	**6**	**1858**	**6**	**1858**	**6**	**1858**
5a	2	1184	3	1454	**2**	1315	**2**	**1300***	**2**	1315
5b	3	1241	4	1528	**3**	1384	**3**	**1378***	**3**	1384
5c	4	1348	**4**	1655	**4**	1522	**4**	**1498***	**4**	1522
5d	6	1638	7	2097	**6**	1991	**6**	**1950***	**6**	1991
6a	2	658	3	754	**2**	722	3	**719***	**2**	722
6b	3	684	4	863	**3**	774	**3**	**754***	**3**	774
6c	7	921	8	1183	**7**	1117	**7**	**1098***	**7**	1117
7a	2	833	3	1040	**2**	966	3	**912***	**2**	966
7b	3	835	4	1046	**3**	960	**3**	**947***	**3**	960
7c	7	949	8	1283	**7**	1165	**7**	**1137***	**7**	1165
8a	2	1150	3	1343	**2**	1292	3	**1203***	**2**	1292
8b	3	1197	4	1453	**3**	1301	**3**	**1267***	**3**	1301
8c	7	1567	**7**	1970	**7**	1853	**7**	**1816***	**7**	1853
9a	2	867	3	1083	**2**	**966**	**2**	**966**	**2**	**966**
9b	3	891	4	1087	**3**	990	**3**	**951***	**3**	990
9c	4	913	**4**	1195	**4**	1031	**4**	**982***	**4**	1031
9d	7	1064	8	1418	**7**	1324	**7**	**1278***	**7**	1324
10a	2	1247	3	1478	**2**	1385	3	**1330***	**2**	1385
10b	3	1259	4	1580	**3**	1395	**3**	**1367***	**3**	1395
10c	4	1301	**4**	1613	**4**	1461	**4**	**1411***	**4**	1461
10d	7	1551	9	1964	**7**	1837	**7**	**1817***	**7**	1837