

# Cooperative Coevolution With Route Distance Grouping for Large-Scale Capacitated Arc Routing Problems

Yi Mei, *Member, IEEE*, Xiaodong Li, *Senior Member, IEEE*, and Xin Yao, *Fellow, IEEE*

**Abstract**—In this paper, a divide-and-conquer approach is proposed to solve the large-scale capacitated arc routing problem (LSCARP) more effectively. Instead of considering the problem as a whole, the proposed approach adopts the cooperative coevolution (CC) framework to decompose it into smaller ones and solve them separately. An effective decomposition scheme called the route distance grouping (RDG) is developed to decompose the problem. Its merit is twofold. First, it employs the route information of the best-so-far solution, so that the quality of the decomposition is upper bounded by that of the best-so-far solution. Thus, it can keep improving the decomposition by updating the best-so-far solution during the search. Second, it defines a distance between routes, based on which the potentially better decompositions can be identified. Therefore, RDG is able to obtain promising decompositions and focus the search on the promising regions of the vast solution space. Experimental studies verified the efficacy of RDG on the instances with a large number of tasks and tight capacity constraints, where it managed to obtain significantly better results than its counterpart without decomposition in a much shorter time. Furthermore, the best-known solutions of the EGL-G LSCARP instances are much improved.

**Index Terms**—Capacitated arc routing problem, cooperative coevolution, memetic algorithm, route distance grouping, scalability.

## I. INTRODUCTION

THE CAPACITATED arc routing problem (CARP) [1] is a well known combinatorial optimization problem, which has a lot of applications in the logistics area, such as winter gritting [2]–[5], waste collection [6]–[8], and snow removal [9], [10]. The problem requires designing an optimal schedule to finish the service of a set of edges and arcs (i.e., directed edges) in a given network map subject to predefined constraints so that the total cost is minimized.

Manuscript received January 3, 2013; revised July 21, 2013; accepted July 30, 2013. Date of publication September 11, 2013; date of current version May 27, 2014. This work was supported in part by the ARC Discovery under Grant DP120102205 and in part by the EPSRC under Grant EP/I010297/1. The work of X. Yao was supported by the Royal Society Wolfson Research Merit Award.

Y. Mei and X. Li are with the Evolutionary Computation and Machine Learning Research Group, School of Computer Science and Information Technology, RMIT University, Melbourne VIC 3000, Australia (e-mail: yi.mei@rmit.edu.au; xiaodong.li@rmit.edu.au).

X. Yao is with the Centre of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, Birmingham B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

Digital Object Identifier 10.1109/TEVC.2013.2281503

In real-world situations, the problem size of CARP is usually very large. For example, for the urban waste collection problem, there may be hundreds or even thousands of streets in the city for which waste is to be collected. Therefore, it is important to study how to solve large-scale CARP (LSCARP). Here, LSCARP is a CARP with more than 300 edges (i.e., the required edges) to be served. The size of 300 is chosen because previous studies have shown that it is large enough to pose a scalability challenge [11]–[14], where the tested algorithms either failed to obtain competitive results [11], [12], [14] or required too much computational time [13].

LSCARP was first considered in [11], where an LSCARP test set named the EGL-G set was generated to evaluate the performance of algorithms. In contrast with the commonly-used benchmark sets (the gdb, [15], val, [16], egl, [17]–[19] and Beullens' test sets [20]), in which the number of required edges ranges from 11 to 190, all the instances in the EGL-G set have more than 300 required edges. Nevertheless, most of the algorithms for CARP that showed competitive performance on the small and medium-sized test sets [21]–[26] were not examined on the EGL-G test set. Clearly, LSCARP has been overlooked so far, with only a few exceptions [11]–[14].

Solving LSCARP is much more challenging than solving ones of small and medium size. This is because the solution space increases exponentially as the problem size increases. Given  $n$  required edges and  $m$  vehicles, the size of the solution space is  $O(2^n(n+m)!/m!)$ . It becomes much more difficult for the algorithm to locate the promising regions in such a vast solution space. Hence, all the previously tested algorithms that neglected the issue of scalability showed poor performance on the EGL-G set.

In order to search more efficiently in the large solution space, one can use the divide-and-conquer strategy to decompose the original large problem into a number of smaller subcomponents, and then solve each subcomponent separately. In this way, the solution space can be much reduced, and the search can focus only on the regions defined by the subcomponents. When decomposing the problem, the key issue is to ensure that the solution spaces of the subcomponents are promising regions, and thus, solving the subcomponents separately can lead to a good result of the overall problem. Nevertheless, in practice, the information known *a priori* is often not enough to obtain the ideal decomposition. For LSCARP, it is difficult to develop a good decomposition

beforehand, since the problem is very complex and the quality of the decomposition depends not only on the mutual distance between the vertices, but also on the satisfaction of the capacity constraint.

In this case, one alternative is to dynamically change the ingredients of the subcomponents during the search process so that the quality of the decomposition can be improved continuously as more information about the solution space is gathered. Here, we use the cooperative coevolution (CC) framework to achieve this, as it has been successfully applied to solving large-scale function optimization problems [27]–[30]. The CC framework divides the whole evolutionary optimization process into several cycles. In each cycle, the elements of the subcomponents are reassigned by some decomposition scheme (e.g., random grouping [27] and delta grouping [28]).

The performance of the CC framework largely depends on the decomposition scheme. Thus, it is important to identify the promising decompositions. However, this is not a trivial task for LSCARP due to two difficulties. First, the large problem size leads to an enormous number of possible decompositions, and the proportion of promising decompositions within all the possible decompositions is very low. Second, after the decomposition, each subcomponent is still NP-hard, and cannot be solved exactly. Therefore, the decompositions cannot be accurately evaluated. To the best of our knowledge, the above issues for decomposing LSCARP have not been considered before, and this paper is the first attempt to address them.

In contrast to LSCARP, there has been quite a lot of work focused on solving large-scale vehicle routing problem (VRP) with decomposition strategies. Taillard *et al.* [31] introduced a route-clustering scheme based on the gravity of the routes and combined it with tabu search and variable neighborhood search. Bent and Van Hentenryck [32] developed a decoupling strategy of the routes for large-scale VRP with time windows (VRPTW) by considering the customer region as a circle and dividing it into wedges based on the coordinates of the customers. Mester *et al.* [33] proposed a route-clustering method that divides the entire area into smaller rectangles. Qi *et al.* [34] designed a clustering strategy of the customers for large-scale VRPTW that uses the  $k$ -medoid approach to combine the customers that are close to each other in terms of both Euclidean distance and time window. Note that one can transform LSCARP to the corresponding VRP by problem reformulation and adopt the existing decomposition strategies. However, this will lead to a much larger problem size. The best reformulations [35], [36] double the problem size. However, further changes are needed on the algorithms for VRP when applying them to solve these formulations. The best-so-far reformulation without modifying the algorithms [37] still triples the problem size. Additionally, in LSCARP, the distance between two vertices is not necessarily proportional to the Euclidean distance between their coordinates, but the shortest traversing distance obtained by Dijkstra's algorithm [38]. For example, two geographically close vertices may be disconnected in the graph, and have a much larger traversing distance than the Euclidean distance between them. For this reason, the geographic-based decomposition strategies for VRPs may not perform well for LSCARP. Therefore, in this paper, we will

focus on developing decomposition strategies for LSCARP directly based on the distances between the vertices rather than the geographic information.

In this paper, an effective decomposition scheme named the route distance grouping (RDG) is proposed for LSCARP. RDG has two main merits. First, it uses the route information of the best-so-far solution. That is, in each cycle, it decomposes the problem by dividing the routes of the best-so-far solution into multiple groups. In this way, the quality of the obtained decomposition is upper bounded by that of the best-so-far solution, and can be continuously improved as the best-so-far solution is updated during the search. Second, it defines a distance between two routes, and divides the routes based on such distance so that the routes that are closer to each other are more likely to be placed in the same subcomponent. Therefore, RDG can identify the promising decompositions without using the geographic information. Then, to solve the subcomponents effectively, the proposed CC framework is combined with the memetic algorithm with extended neighborhood search (MAENS) [23], which has been demonstrated to be competitive in solving small and medium-sized CARPs. The resultant algorithm, which is named RDG-MAENS, is evaluated on a number of test sets with various parameter settings. The experimental studies verified the efficacy of RDG for solving the large and difficult EGL-G instances, on which RDG-MAENS managed to obtain significantly better solutions in a much shorter time than its counterpart without decomposition (MAENS).

The rest of the paper is organized as follows. First, CARP is introduced in Section II. Note that LSCARP is essentially CARP with large problem size (i.e., more than 300 required edges). After that, the CC framework for LSCARP is described in Section III. Then, in Section IV, the difficulties in developing a decomposition scheme for LSCARP are addressed and RDG is developed. Afterward, the experimental studies are carried out in Section V. Finally, the conclusion and future work are provided in Section VI.

## II. CAPACITATED ARC ROUTING PROBLEM

In CARP, a graph  $G(V, E, A)$  is given, where  $V$ ,  $E$ , and  $A$  are the set of vertices, edges, and arcs of the graph. For both  $E$  and  $A$ , there are subsets  $Z_E \subseteq E$  and  $Z_A \subseteq A$ , which are also called tasks, that need to be served. For the tasks in  $Z_E$ , service from either direction is acceptable, while for the tasks in  $Z_A$ , only the same direction as the arc is allowed. The services are done by a number of vehicles located at a depot vertex  $v_0 \in V$ . Let the set of all the tasks be denoted as  $Z = Z_E \cup Z_A$ . Each task  $z \in Z$  has a positive demand  $d(z) > 0$  and a positive serving cost  $sc(z) > 0$ . Besides, traversing from any vertex  $v_i$  to a different vertex  $v_j$  induces a positive deadheading cost  $dc(v_i, v_j) > 0$ . If  $v_i$  and  $v_j$  are disconnected, then  $dc(v_i, v_j) = \infty$ . Each vehicle has a limited capacity  $Q$  of the demand, which is smaller than the total demand of all the tasks. Hence, multiple vehicles are needed. CARP aims at designing a routing plan to finish the service of the task set  $Z$  with the minimal total cost (deadheading plus serving costs) subject to the following constraints.

- 1) Each vehicle must start and end at the depot.
- 2) Each task is served exactly once.
- 3) The total demand of the tasks served by each vehicle cannot exceed its capacity  $Q$ .

The mathematical formulation of CARP has been intensively investigated. Baldacci and Maniezzo [35] formulated the undirected CARP with  $n$  tasks as a VRP with  $2n + 1$  customers. Belenguer and Benavent [39] proposed a mathematical formulation of CARP based on the cut constraints, and designed a cutting plane method to solve it. Bartolini *et al.* [40] and Martinelli *et al.* [14] developed relaxed mathematical CARP models and obtained the best-so-far lower bounds for the benchmark instances with exact methods. For the sake of brevity, the details of the mathematical CARP models are not described here, since they can be found in [39], [40] and [14].

### III. COOPERATIVE COEVOLUTION FOR LSCARP

The concept of CC was first proposed by Potter and De Jong [41] to solve a problem by dividing the decision variables into smaller subcomponents and evolving them separately. In their framework (CCGA-1), the original  $n$ -dimensional decision vector is predecomposed into  $n$  one-dimensional subcomponents before the optimization process. Then, in each generation, the subcomponents are evolved in turn. Subsequent to CCGA-1, there are a number of works on the static decomposition (e.g., the 1-D [42], split-in-half [43] strategies and the more general one dividing into  $k$   $s$ -dimensional subcomponents where  $k \times s = n$  [44]). However, they did not consider the interdependency between the variables and fixed the subcomponents throughout the search process. Yang *et al.* [27] considered the interaction between variables and proposed a new CC framework, which divides the whole optimization process into a number of cycles. In each cycle, the variables are randomly reassigned into the subcomponents to increase the probability of placing the interacting variables in the same subcomponent. It is obvious that the variables of LSCARP interact with each other, and thus, Yang *et al.*'s CC framework is quite suitable for solving LSCARP. It can be described as follows.

- Step 1** Set  $i = 1$  to start a new cycle.
- Step 2** Split the original  $n$ -dimensional decision vector  $\mathbf{x} = (x_1, \dots, x_n)$  into  $g$  non-overlapping  $l$ -dimensional subcomponents  $\mathbf{x}_1, \dots, \mathbf{x}_g$  ( $g \cdot l = n$ ) randomly. Here, randomly means that each variable  $x_i$  ( $i = 1, \dots, n$ ) has the same chance to be assigned into any of the subcomponents.
- Step 3** Evolve the subcomponent  $\mathbf{x}_i$  by an EA for a predefined number of iterations.
- Step 4** If  $i < g$ , then set  $i = i + 1$  and go to Step 3.
- Step 5** If the stopping criteria are met, then stop. Otherwise go to Step 1 for the next cycle.

The random assignment of the variables in Step 2 is called the random grouping scheme.

When adopting the above CC framework in LSCARP, the decision variables are the tasks  $z \in Z$ . However, unlike function optimization, which is to determine the optimal value

---

#### Algorithm 1 CC framework for LSCARP

---

```

1: procedure CC(LSCARP,  $g$ )
2:   Initialize population  $\mathbf{p}(Z)$ ;
3:    $\bar{\mathbf{s}}(Z) = \arg \min_{\mathbf{s}(Z) \in \mathbf{p}(Z)} (tc(\mathbf{s}(Z)))$ ;
4:    $t \leftarrow 1$ 
5:   repeat
6:      $(Z_1, \dots, Z_g) = \text{Decompose}(Z)$ ;
7:     for  $i = 1 \rightarrow g$  do
8:        $\mathbf{p}(Z_i) = \text{Pop2Subpop}(\mathbf{p}(Z), Z_i)$ ;
9:        $\bar{\mathbf{s}}(Z_i) = \arg \min_{\mathbf{s}(Z_i) \in \mathbf{p}(Z_i)} (tc(\mathbf{s}(Z_i)))$ ;
10:       $(\bar{\mathbf{s}}(Z_i), \mathbf{p}(Z_i)) = \text{Evolve}(\bar{\mathbf{s}}(Z_i), \mathbf{p}(Z_i))$ ;
11:    end for
12:     $\mathbf{p}(Z) = \text{Subpop2Pop}(\mathbf{p}(Z_1), \dots, \mathbf{p}(Z_g))$ ;
13:     $\bar{\mathbf{s}}^{(t)}(Z) = \{\bar{\mathbf{s}}(Z_1), \dots, \bar{\mathbf{s}}(Z_g)\}$ ;
14:    if  $tc(\bar{\mathbf{s}}^{(t)}(Z)) < tc(\bar{\mathbf{s}}(Z))$  then
15:       $\bar{\mathbf{s}}(Z) = \bar{\mathbf{s}}^{(t)}(Z)$ ;
16:    end if
17:     $t \leftarrow t + 1$ ;
18:  until  $t$  reaches to a predefined upper bound
19:  return  $\bar{\mathbf{s}}(Z)$ ;
20: end procedure

```

---

for each decision variable of the vector  $\mathbf{x}$ , LSCARP is to cluster all the tasks into different routes, and sort the tasks within each route, both in an optimal way, so that the total cost (denoted as  $tc$ ) of the routes is minimized.

To decompose a LSCARP, the task set  $Z$  is essentially decomposed into nonoverlapping subsets  $Z_1, \dots, Z_g$ . Then, at Step 3 of the above CC framework, evolving the subcomponent  $Z_i$  is defined as finding a solution  $\mathbf{s}(Z_i)$  that serves all the tasks in  $Z_i$  with least total cost  $tc(\mathbf{s}(Z_i))$  so that

- 1) each route of  $\mathbf{s}(Z_i)$  starts and ends at the depot;
- 2) each task in  $Z_i$  is served exactly once, while all the tasks in  $Z \setminus Z_i$  are not served;
- 3) the total demand of each route of  $\mathbf{s}(Z_i)$  cannot exceed  $Q$ .

It is obvious that given the feasible solutions  $\mathbf{s}(Z_1), \dots, \mathbf{s}(Z_g)$  for the subcomponents, concatenating their routes  $\cup_{i=1}^g \mathbf{s}(Z_i)$  will lead to a feasible solution for the overall problem.

The pseudo code of the CC for LSCARP is given in Algorithm 1. It maintains the population  $\mathbf{p}(Z)$  and the best feasible solution  $\bar{\mathbf{s}}(Z)$  of the overall problem throughout the search process, and finally outputs  $\bar{\mathbf{s}}(Z)$ . In each cycle,  $Z$  is first decomposed into  $(Z_1, \dots, Z_g)$  by *Decompose()*. Afterward, for each  $Z_i$ , a subpopulation  $\mathbf{p}(Z_i)$  is generated by *Pop2Subpop()*. Then,  $\mathbf{p}(Z_i)$  is evolved by *Evolve()* and the corresponding best feasible solution  $\bar{\mathbf{s}}(Z_i)$  is updated. Finally,  $\mathbf{p}(Z)$  is updated from  $\mathbf{p}(Z_1), \dots, \mathbf{p}(Z_g)$  by *Subpop2Pop()*. Besides, the best feasible solution  $\bar{\mathbf{s}}^{(t)}(Z)$  found in this cycle is obtained by concatenating  $\bar{\mathbf{s}}(Z_1), \dots, \bar{\mathbf{s}}(Z_g)$  (the same as the context vector [30], [44]), and replaces  $\bar{\mathbf{s}}(Z)$  if it is better.

*Decompose()* is the most important and difficult part of the CC, and thus, will be described separately in Section IV. In contrast, *Pop2Subpop()* and *Subpop2Pop()* are rather straightforward. Given  $\mathbf{p}(Z)$  and  $Z_i$ , to obtain the  $j$ th individual  $\mathbf{p}_j(Z_i)$  from  $\mathbf{p}_j(Z)$ , *Pop2Subpop()* scans each route

of  $\mathbf{p}_j(Z)$ , and removes all the tasks in  $Z \setminus Z_i$ . Conversely, given  $\mathbf{p}(Z_1), \dots, \mathbf{p}(Z_g)$ , *Subpop2Pop()* simply update each individual  $\mathbf{p}_j(Z)$  by concatenating all the  $\mathbf{p}_j(Z_i)$ 's routes.

#### IV. ROUTE DISTANCE GROUPING

The objective of *Decompose()* is to obtain the optimal decomposition  $(Z_1^*, \dots, Z_g^*)$  of  $Z$ , so that the union of the optimal solutions  $\mathbf{s}^*(Z_1^*), \dots, \mathbf{s}^*(Z_g^*)$  of the subcomponents is equivalent to the optimal solution  $\mathbf{s}^*(Z)$  of the overall problem, i.e.,  $tc(\cup_{i=1}^g \mathbf{s}^*(Z_i^*)) = tc(\mathbf{s}^*(Z))$ . In other words, the objective is to minimize  $tc(\cup_{i=1}^g \mathbf{s}^*(Z_i))$ .

It is obvious that an optimal decomposition can be obtained by dividing the routes of the optimal solution  $\mathbf{s}^*(Z)$  into subsets of routes. However, in practice, the route information of  $\mathbf{s}^*(Z)$  cannot be known in advance. Therefore, we approximate the route information of  $\mathbf{s}^*(Z)$  with that of the best-so-far solution  $\bar{\mathbf{s}}(Z)$ . That is, the tasks in the same route of  $\bar{\mathbf{s}}(Z)$  are considered more likely to be in the same route of  $\mathbf{s}^*(Z)$ . This idea is similar to grouping solutions with respect to route instead of customer; the latter as in VRP [31]–[33]. Here, we give the theoretical analysis to show that as long as the subcomponents  $(Z_1, \dots, Z_g)$  are obtained by dividing the routes of the best-so-far solution  $\bar{\mathbf{s}}(Z)$ , one can guarantee that the quality of the decomposition must improve along with the improvement of  $\bar{\mathbf{s}}(Z)$  in terms of upper bound of  $tc(\cup_{i=1}^g \mathbf{s}^*(Z_i))$ . The analysis is as follows.

First, we define  $\bar{\mathbf{s}}(Z_i)$  as the subset of routes of  $\bar{\mathbf{s}}$  placed in subcomponent  $Z_i$ . Then, it is obvious that

$$tc(\cup_{i=1}^g \bar{\mathbf{s}}(Z_i)) = tc(\bar{\mathbf{s}}(Z)). \quad (1)$$

On the other hand, since the total cost of a set of routes is the sum of the total costs of each route in the set, we have

$$tc(\cup_{i=1}^g \mathbf{s}^*(Z_i)) = \sum_{i=1}^g tc(\mathbf{s}^*(Z_i)) \quad (2)$$

$$tc(\cup_{i=1}^g \bar{\mathbf{s}}(Z_i)) = \sum_{i=1}^g tc(\bar{\mathbf{s}}(Z_i)). \quad (3)$$

Therefore, we have

$$\begin{aligned} tc(\cup_{i=1}^g \mathbf{s}^*(Z_i)) &= \sum_{i=1}^g tc(\mathbf{s}^*(Z_i)) \\ &\leq \sum_{i=1}^g tc(\bar{\mathbf{s}}(Z_i)) = tc(\cup_{i=1}^g \bar{\mathbf{s}}(Z_i)) = tc(\bar{\mathbf{s}}(Z)). \end{aligned} \quad (4)$$

That is,  $tc(\cup_{i=1}^g \mathbf{s}^*(Z_i))$  is upper bounded by  $tc(\bar{\mathbf{s}}(Z))$ . Thus,  $tc(\cup_{i=1}^g \mathbf{s}^*(Z_i))$  improves along with the improvement of  $tc(\bar{\mathbf{s}}(Z))$  during the search process.

When grouping the routes of  $\bar{\mathbf{s}}(Z)$ , we can leverage on the domain knowledge and develop heuristics to favor potentially better groupings over other groupings. Since the objective is to minimize the total cost, the optimal solution tends to link the tasks that are close to each other, and place them in the same route. Therefore, the tasks that are closer to each other should be more likely to be placed in the same subcomponent. Based on such domain knowledge, it is better to combine the routes

whose tasks are closer to each other. For VRP, the closeness between routes was defined based on geographic information, i.e., coordinates [31]–[33]. However, such a definition may not be proper for LSCARP, because the distance between the nodes is not the Euclidean distance between their coordinates, but the shortest traversing distance obtained by Dijkstra's algorithm. Then, the nodes that are geographically closer do not necessarily have smaller traversing distance. In this case, it is more proper to define the closeness directly based on the distance matrix rather than the coordinates.

To define the closeness between routes for LSCARP, we first define the distance  $\Delta_{\text{task}}(z_1, z_2)$  between two tasks  $z_1$  and  $z_2$  as follows:

$$\Delta_{\text{task}}(z_1, z_2) = \frac{\sum_{i=1}^2 \sum_{j=1}^2 \Delta(v_i(z_1), v_j(z_2))}{4} \quad (5)$$

where  $\Delta(v_i(z_1), v_j(z_2))$  is the traversing distance between the  $i$ th end-node  $v_i(z_1)$  of the task  $z_1$  and the  $j$ th end-node  $v_j(z_2)$  of the task  $z_2$ . Hence,  $\Delta_{\text{task}}(z_1, z_2)$  is defined as the average distance of the four possible links between  $z_1$  and  $z_2$ .

Based on the task distance, the distance  $\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2)$  between two routes  $\mathbf{s}_1$  and  $\mathbf{s}_2$  can be defined as follows:

$$\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{\sum_{z_1 \in \mathbf{s}_1} \sum_{z_2 \in \mathbf{s}_2} \Delta_{\text{task}}(z_1, z_2)}{|\mathbf{s}_1| \cdot |\mathbf{s}_2|}. \quad (6)$$

Therefore,  $\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2)$  is the average distance of all the pairs of the tasks so that one is from  $\mathbf{s}_1$  and the other from  $\mathbf{s}_2$ .

Finally,  $\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2)$  needs to be normalized with respect to  $\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_1)$  and  $\Delta_{\text{route}}(\mathbf{s}_2, \mathbf{s}_2)$ . To this end, we define the following normalized distance to represent the closeness between  $\mathbf{s}_1$  and  $\mathbf{s}_2$ :

$$\hat{\Delta}_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2)}{\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_1)} \cdot \frac{\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2)}{\Delta_{\text{route}}(\mathbf{s}_2, \mathbf{s}_2)}. \quad (7)$$

An example of computing the  $\hat{\Delta}_{\text{route}}$  matrix is given in Figs. 1 and 2. First, the graph is shown in Fig. 1, where the tasks  $z_1, \dots, z_7$  are represented by the solid lines.  $v_0$  is the depot. The deadheading costs of the edges of the graph are all one. Thus, the distance matrix  $\Delta$  between the vertices is

$$\Delta = \begin{matrix} & \begin{matrix} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{matrix} \\ \begin{matrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} & \begin{pmatrix} 0 & 1 & 2 & 2 & 2 & 1 & 1 \\ 1 & 0 & 1 & 2 & 2 & 2 & 1 \\ 2 & 1 & 0 & 1 & 2 & 2 & 1 \\ 2 & 2 & 1 & 0 & 1 & 2 & 1 \\ 2 & 2 & 2 & 1 & 0 & 1 & 1 \\ 1 & 2 & 2 & 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{matrix}.$$

Then, Fig. 2 shows three routes  $\mathbf{s}_1 = (0, x_1, x_3, x_5, 0)$  (the inner circle),  $\mathbf{s}_2 = (0, x_2, x_4, x_6, 0)$  (the outer circle), and  $\mathbf{s}_3 = (0, x_7, 0)$  to serve all the tasks, where  $x_i$  is the ID assigned to the current direction of the task  $z_i$ ,  $\forall i = 1, \dots, 7$ . Each route starts and ends at the depot loop indexed by 0. The services are represented by solid arrows, while the deadheading paths are represented by dashed arrows. It can be seen that  $\mathbf{s}_1$  and  $\mathbf{s}_2$  are more tangled with each other than with  $\mathbf{s}_3$ , and thus, should be more likely to be placed in the same subcomponent. In



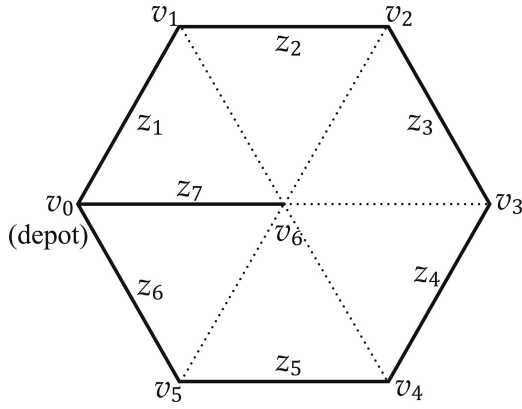
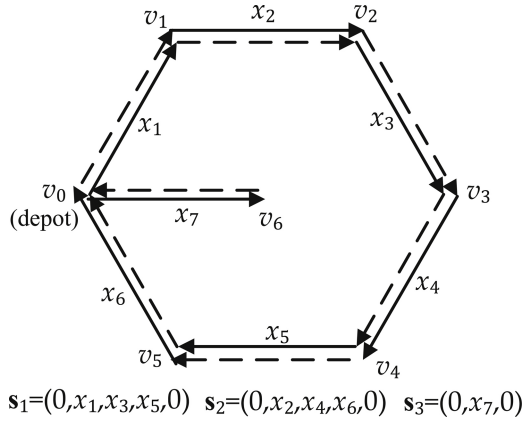
Fig. 1. Example graph of computing the  $\hat{\Delta}_{\text{route}}$  matrix.

Fig. 2. Given three routes for the example graph.

fact, if we have a capacity constraint that prohibits each route from serving more than three tasks, then, combining  $s_1$  and  $s_2$  together can lead to two improved routes  $s'_1 = (0, x_1, x_2, x_3, 0)$  and  $s'_2 = (0, x_4, x_5, x_6, 0)$ . When calculating the total cost, we have

$$tc(s_1) = tc(s_2) = \sum_{i=0}^4 \Delta(v_i, v_{i+1}) + \Delta(v_5, v_0) = 6 \quad (8)$$

$$tc(s'_1) = \sum_{i=0}^2 \Delta(v_i, v_{i+1}) + \Delta(v_3, v_0) = 5 \quad (9)$$

$$tc(s'_2) = \Delta(v_0, v_3) + \sum_{i=3}^4 \Delta(v_i, v_{i+1}) + \Delta(v_5, v_0) = 5. \quad (10)$$

Then,  $tc(s'_1) + tc(s'_2) < tc(s_1) + tc(s_2)$ . However, combining  $s_1$  (or  $s_2$ ) and  $s_3$  cannot lead to improvement. According to (5), the distance matrix between the tasks is

$$\Delta_{\text{task}} = \begin{matrix} & \begin{matrix} z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 \end{matrix} \\ \begin{matrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{matrix} & \begin{pmatrix} 1/2 & 1 & 7/4 & 2 & 7/4 & 1 & 3/4 \\ 1 & 1/2 & 1 & 7/4 & 2 & 7/4 & 5/4 \\ 7/4 & 1 & 1/2 & 1 & 7/4 & 2 & 3/2 \\ 2 & 7/4 & 1 & 1/2 & 1 & 7/4 & 3/2 \\ 7/4 & 2 & 7/4 & 1 & 1/2 & 1 & 5/4 \\ 1 & 7/4 & 2 & 7/4 & 1 & 1/2 & 3/4 \\ 3/4 & 5/4 & 3/2 & 3/2 & 5/4 & 3/4 & 1/2 \end{pmatrix} \end{matrix}.$$

Then, the distance matrix between the routes in Fig. 2 is computed by (6) and the results are

$$\Delta_{\text{route}} = \begin{matrix} & \begin{matrix} s_1 & s_2 & s_3 \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} & \begin{pmatrix} 4/3 & 4/3 & 7/6 \\ 4/3 & 4/3 & 7/6 \\ 7/6 & 7/6 & 1/2 \end{pmatrix} \end{matrix}.$$

Then, based on (7), we have

$$\hat{\Delta}_{\text{route}} = \begin{matrix} & \begin{matrix} s_1 & s_2 & s_3 \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ s_3 \end{matrix} & \begin{pmatrix} 1 & 1 & 49/24 \\ 1 & 1 & 49/24 \\ 49/24 & 49/24 & 1 \end{pmatrix} \end{matrix}.$$

It can be seen that  $\hat{\Delta}_{\text{route}}(s_1, s_2)$  is much smaller than  $\hat{\Delta}_{\text{route}}(s_1, s_3)$  and  $\hat{\Delta}_{\text{route}}(s_2, s_3)$ , which is consistent with the intuition that  $s_1$  and  $s_2$  should be more likely to be combined together. On the other hand, if there is no normalization, then  $\Delta_{\text{route}}(s_1, s_2)$  is larger than  $\Delta_{\text{route}}(s_1, s_3)$  and  $\Delta_{\text{route}}(s_2, s_3)$ . This indicates the importance of normalization for identifying the more tangled routes.

Based on the above definition, if two routes have smaller  $\hat{\Delta}_{\text{route}}(s_1, s_2)$ , they are considered to be closer to each other, and it should be more promising to place them in the same subcomponent. Following this understanding, we can define the decomposition as a clustering problem of the routes based on the normalized distance matrix  $(\hat{\Delta}_{\text{route}}(s_{k_1}, s_{k_2}))_{m \times m}$ , where  $m$  is the number of routes in the solution  $s$ . If two routes have a smaller normalized distance, they are more likely to be in the same cluster.

Here, we define the clustering problem as a fuzzy  $k$ -medoids [45], which can be stated as follows:

$$\min_{\mathbf{c} \subseteq \mathbf{s}} \mathcal{J}_\alpha(\mathbf{c}; \mathbf{s}) = \sum_{s_i \in \mathbf{s} \setminus \mathbf{c}} \sum_{c_j \in \mathbf{c}} \mathcal{M}_\alpha(s_i, c_j) \cdot \hat{\Delta}_{\text{route}}(s_i, c_j). \quad (11)$$

Given  $(\hat{\Delta}_{\text{route}}(s_{k_1}, s_{k_2}))_{m \times m}$  and the predefined number of groups  $g$ , the aim of the fuzzy  $k$ -medoids problem is to select a subset of the routes  $\mathbf{c} = \{c_1, \dots, c_g\}$ , which are called the medoids, out of the entire set of routes  $\mathbf{s} = \{s_1, \dots, s_m\}$ , so that the sum of the fuzzy distances between all the pairs of the medoids and nonmedoids is minimized. The function  $\mathcal{M}_\alpha(s_i, c_j)$ , which is called the membership of  $s_i$  to  $c_j$ , is used to obtain the fuzzy distances. Generally,  $\mathcal{M}_\alpha(s_i, c_j)$  is larger if  $\hat{\Delta}_{\text{route}}(s_i, c_j)$  is smaller. Here, it is defined as follows:

$$\mathcal{M}_\alpha(s_i, c_j) = \frac{\left( \frac{1}{\hat{\Delta}_{\text{route}}(s_i, c_j)} \right)^\alpha}{\sum_{k=1}^g \left( \frac{1}{\hat{\Delta}_{\text{route}}(s_i, c_k)} \right)^\alpha}. \quad (12)$$

The parameter  $\alpha \in [0, \infty)$  controls the degree of fuzziness. When  $\alpha = 0$ , the fuzziness is maximized and the membership of a nonmedoid to each medoid is the same as  $1/g$  regardless of the distance between them. When  $\alpha = \infty$ , there is no fuzziness. The membership is 1 to the closest medoid, and 0 to the other medoids. In this case, the fuzzy  $k$ -medoids problem is reduced to the  $k$ -medoids problem [46].

To solve the fuzzy  $k$ -medoids problem, we employ the partitioning around medoids (PAM) algorithm [46], which is a simple and commonly used method. It is described in

**Algorithm 2** PAM algorithm for fuzzy  $k$ -medoids problem

---

```

1: procedure PAM( $(\hat{\Delta}_{\text{route}}(\mathbf{s}_{k_1}, \mathbf{s}_{k_2}))_{m \times m}, g, \alpha$ )
2:   Randomly choose  $\mathbf{c} \subseteq \mathbf{s}$ ;
3:   Compute  $\mathcal{J}_\alpha(\mathbf{c}; \mathbf{s})$  based on Eqs. (11) and (12);
4:   repeat
5:      $\mathbf{c}^{\text{old}} \leftarrow \mathbf{c}, \mathbf{c}^* \leftarrow \mathbf{c}$ ;
6:     Set  $\mathcal{J}_\alpha(\mathbf{c}^*; \mathbf{s}) = \mathcal{J}_\alpha(\mathbf{c}; \mathbf{s})$ ;
7:     for  $j = 1 \rightarrow g$  do
8:       for  $\mathbf{s}_i \in \mathbf{s} \setminus \mathbf{c}$  do
9:          $\mathbf{c}' \leftarrow \mathbf{c}$ ;
10:        Swap  $\mathbf{c}'_j$  and  $\mathbf{s}_i$ ;
11:        Compute  $\mathcal{J}_\alpha(\mathbf{c}'; \mathbf{s})$ ;
12:        if  $\mathcal{J}_\alpha(\mathbf{c}'; \mathbf{s}) < \mathcal{J}_\alpha(\mathbf{c}^*; \mathbf{s})$  then
13:           $\mathbf{c}^* \leftarrow \mathbf{c}'$ ;
14:        end if
15:      end for
16:    end for
17:     $\mathbf{c} \leftarrow \mathbf{c}^*$ ;
18:  until  $\mathbf{c}^{\text{old}} = \mathbf{c}$ 
19:  return  $\mathbf{c}$ ;
20: end procedure

```

---

Algorithm 2. At the beginning, the medoids  $\mathbf{c} = \{\mathbf{c}_1, \dots, \mathbf{c}_g\}$  are randomly chosen. Then, at each iteration, all the swaps between the medoids  $\mathbf{c}_j \in \mathbf{c}$  and the nonmedoids  $\mathbf{s}_i \in \mathbf{s} \setminus \mathbf{c}$  are evaluated in terms of the new objective value  $\mathcal{J}_\alpha(\mathbf{c}'; \mathbf{s})$  after the swap, and the one with the minimal  $\mathcal{J}_\alpha(\mathbf{c}'; \mathbf{s})$  is selected to be conducted on  $\mathbf{c}$ . The swap is repeated until no improvement is obtained on  $\mathcal{J}_\alpha(\mathbf{c}; \mathbf{s})$ .

After applying the PAM algorithm to the routes  $\mathbf{s}$ , the output is a number of the medoids  $\mathbf{c} = \{\mathbf{c}_1, \dots, \mathbf{c}_g\}$ , each representing a group. Then, it is necessary to assign all the nonmedoid routes to the groups. Here, we adopt the simple roulette wheel method based on the memberships of the nonmedoids to the medoids. The probability of assigning a nonmedoid to the group of each medoid is proportional to its membership to the medoid. Such an assignment process is described in Algorithm 3.

Finally, the decomposition  $(Z_1, \dots, Z_g)$  of the task set  $Z$  can be obtained directly from the grouping of the routes. Specifically, if a task is in a route of  $\mathcal{G}_i$ , then it is put into  $Z_i$ . This can be depicted in Algorithm 4.

In summary, at the beginning of each cycle of the CC framework, the normalized distance matrix of the routes of the best-so-far solution is computed, and the corresponding decomposition is then obtained by the aforementioned route grouping method. This decomposition is called the RDG, as it uses a grouping scheme of the routes based on the distance between them. It is described in Algorithm 5, and replaces *Decompose*( $Z$ ) in line 6 of Algorithm 1.

Compared with existing clustering methods [31]–[34] for VRP, RDG has the following advantages. First, it is based on a rigorous theoretical analysis so that the decomposition can be guaranteed to be improved along with the improvement of the best-so-far solution. Second, a normalized distance matrix of the routes for LSCARP is defined to reflect the closeness

**Algorithm 3** Assignment of the non-medoids

---

```

1: procedure NONMEDASSIGN( $\mathcal{M}_\alpha(\mathbf{s}_i, \mathbf{c}_j)$ )
2:   for  $i = 1 \rightarrow g$  do
3:      $\mathcal{G}_i = \{\mathbf{c}_i\}$ ;
4:   end for
5:   for  $\mathbf{s}_i \in \mathbf{s} \setminus \mathbf{c}$  do
6:     Sample the random variable  $r \in [0, 1]$  uniformly;
7:      $\varphi = 0$ ;
8:     for  $k = 1 \rightarrow g$  do
9:        $\varphi \leftarrow \varphi + \mathcal{M}_\alpha(\mathbf{c}_k; \mathbf{s}_i)$ ;
10:      if  $\varphi \geq r$  then
11:        break;
12:      end if
13:     $\mathcal{G}_k \leftarrow \mathcal{G}_k \cup \mathbf{s}_i$ ;
14:  end for
15: end for
16: return  $(\mathcal{G}_1, \dots, \mathcal{G}_g)$ ;
17: end procedure

```

---

**Algorithm 4** Task decomposition from route grouping

---

```

1: procedure TASKDECOMP( $(\mathcal{G}_1, \dots, \mathcal{G}_g)$ )
2:   for  $i = 1 \rightarrow g$  do
3:      $Z_i = \{\}$ ;
4:     for  $\mathbf{s}_k \in \mathcal{G}_i$  do
5:       for  $z \in \mathbf{s}_k$  do
6:          $Z_i \leftarrow Z_i \cup z$ ;
7:       end for
8:     end for
9:   end for
10:  return  $(Z_1, \dots, Z_g)$ ;
11: end procedure

```

---

**Algorithm 5** Route distance grouping decomposition

---

```

1: procedure RDGDECOMPOSE( $\bar{\mathbf{s}}, g, \alpha$ )
2:   Compute  $(\hat{\Delta}_{\text{route}}(\bar{\mathbf{s}}_{k_1}, \bar{\mathbf{s}}_{k_2}))_{m \times m}$  by Eqs. (5)–(7);
3:    $\bar{\mathbf{c}} = \text{PAM}((\hat{\Delta}_{\text{route}}(\bar{\mathbf{s}}_{k_1}, \bar{\mathbf{s}}_{k_2}))_{m \times m}, g, \alpha)$ ;
4:    $(\bar{\mathcal{G}}_1, \dots, \bar{\mathcal{G}}_g) = \text{NonMedAssign}(\mathcal{M}_\alpha(\bar{\mathbf{s}}_i, \bar{\mathbf{c}}_j))$ ;
5:    $(Z_1, \dots, Z_g) = \text{TaskDecomp}((\bar{\mathcal{G}}_1, \dots, \bar{\mathcal{G}}_g))$ ;
6:   return  $(Z_1, \dots, Z_g)$ ;
7: end procedure

```

---

between the routes, and a fuzzy route clustering problem that is directly based on the normalized distance matrix is defined. In this way, one does not need to assume that geographically closer nodes must have smaller traversing distance and the coordinates of the nodes are known. Finally, the introduced fuzziness parameter  $\alpha$  can control the degree of freedom during the route clustering, and help the decomposition to jump out of the local optima.

## V. EXPERIMENTAL STUDIES

For the proposed CC algorithm with RDG for LSCARP, the performance mainly depends on two parameters:  $g$  and  $\alpha$ . First, it is obvious that  $g$  influences the performance of the algorithm in the same way as the number of clusters

influences the quality of solutions in the clustering problem.  $g$  also determines the number of possible decompositions and thus the probability of obtaining ideal decompositions. Second,  $\alpha$  controls the degree of freedom of the route grouping. If  $\alpha$  is too small, then the route grouping is too random and there is not enough bias to the promising decompositions. If  $\alpha$  is too large, the grouping may be too deterministic to jump out of the local optima. To investigate the effect of  $g$  and  $\alpha$ , we evaluated the performance of the proposed algorithm under different values of  $g$  and  $\alpha$ .

Finally, the algorithm to solve each subcomponent is to be decided. Here, we adopt MAENS [23] due to its competitiveness for small and medium-sized CARP in terms of the quality of the final solution. Note that Chen extended MAENS to MAENS+ [13] to address the scalability issue. However, the experimental studies only showed marginal improvement on the quality of solution. For the sake of simplicity, the standard version of MAENS is adopted. Together with RDG, the proposed algorithm is called RDG-MAENS.

#### A. Parameter Settings

The parameter settings are given in Table I, where  $g$  and  $\alpha$  are the parameters of RDG, and the remaining are the parameters of MAENS and the CC framework. Here,  $g = 2$  and  $3$  and  $\alpha = 1, 5$  and  $10$  are tested. Therefore, there are, in total,  $2 \cdot 3 = 6$  versions of RDG-MAENS to be compared. For the sake of simplicity, we refer to the different versions of RDG-MAENS ( $g, \alpha$ ) hereafter. For example, (2, 5) refers to the RDG-MAENS with  $g = 2$  and  $\alpha = 5$ . The number of cycles is arbitrarily set to 50 given 500 total generations to achieve a relatively good tradeoff between the number of cycles and the exploitation of the subcomponents in each cycle. To verify the efficacy of RDG, RDG-MAENS is compared with MAENS, which is its counterpart without RDG decomposition, under the same parameter settings and computational platform. To this end, we obtained the source code of MAENS<sup>1</sup> and ran it again under the same computational platform as RDG-MAENS on all the test instances.

The Beullens' C, D, E, F sets [20], egl [17]–[19], and EGL-G [11] test sets are selected to evaluate RDG-MAENS. Beullens' sets are based on the intercity road network in Flanders, Belgium, each containing 25 different instances with 28–121 tasks. The D and F instances share the same networks with the C and E instances, respectively, but with a larger capacity. The egl set is the largest commonly used CARP test set in the literature. It was derived from a winter gritting application in Lancashire, U.K., which has 24 instances with the number of tasks ranging from 51 to 190. The EGL-G set was also based on the road network of Lancashire, U.K., consisting of TEN LSCARP instances with 347 to 375 tasks. In summary, the test sets consist of small, medium, and large scale CARP instances. Although RDG-MAENS is proposed specifically for solving LSCARP, one may still be interested in its performance on the small and medium sized instances. The gdb [15] and val [16] test sets are not selected here, since all the gdb instances have

TABLE I  
PARAMETER SETTINGS OF RDG-MAENS

Parameter	Description	Value
$g$	Number of subcomponents	2, 3
$\alpha$	Fuzziness control parameter	1, 5, 10
$p_{size}$	Population size	30
$offsize$	Offspring population size	$6 \cdot p_{size}$
$P_{ls}$	Probability of local search	0.2
$gen$	Maximal generations	500
$cycles$	Number of cycles	50

been solved optimally, and the problem size of the val set is covered by that of Beullens' sets.

For the test sets, the state-of-the-art algorithms are selected for comparison. To be specific, for Beullens' sets, GLS [20] and Ant-CARP (the version with 12 move types) [25] are taken into account. For the egl set, Ant-CARP and VNS (the 3.6 GHz version) [10] are selected, and for the EGL-G set, ILS-RVND [14] is chosen.

For each parameter setting, 30 independent runs of RDG-MAENS and MAENS were conducted on all the test instances. The code of the MAENS part was directly obtained from the one provided in the original reference [23], and the other parts were implemented in C++. RDG-MAENS and MAENS were compiled by GNU Compiler Collection (GCC) for Windows and run on the CPU Intel Core i7-2600 @3.4 GHz, using only one core.

#### B. Results and Discussions

First, the average performance and runtime of RDG-MAENS are evaluated. Tables II–VII show the mean of the 30 total costs and computational time of RDG-MAENS and MAENS, along with the average performance of each compared algorithm and the features of the test instances.  $|V|$ ,  $|E|$ , and  $|Z|$  refer to the number of vertices, edges, and tasks, respectively.  $\tau$  is the minimal number of vehicles required to serve all the tasks, which is obtained as follows:

$$\tau = \left\lceil \frac{\sum_{z \in Z} d(z)}{Q} \right\rceil. \quad (13)$$

In general, a larger  $\tau$  indicates a tighter capacity constraint and, thus, a higher level of difficulty of the problem.

In the tables, the average performance and computational time of the compared algorithms were obtained directly from the original references except MAENS. For GLS, the column labelled "Cost" indicates the total cost of the final solution, since it was run only once. For Ant-CARP, the median total cost and average computational time of five independent runs are provided. For VNS and ILS-RVND, the mean total cost and computational time of ten independent runs are given. For Beullens' sets, the past results were presented in the form of either total cost or total deadheading cost, i.e., total cost minus total serving cost. For the sake of consistency, all the results have been transformed to total deadheading cost.

It is difficult to conduct a fair comparison on computational time as there is no common computational platform, including CPU frequency, RAM, operating system, implementing language, compiler, etc. Here, the computation time of each

<sup>1</sup>The code was obtained from the website .

TABLE II

AVERAGE (MEAN OR MEDIAN) TOTAL COST AND COMPUTATIONAL TIME OF THE COMPARED ALGORITHMS ON BEULLENS' C TEST SET. FOR EACH INSTANCE, THE MINIMAL TOTAL COST OF MAENS AND RDG-MAENS IS MARKED WITH  $\dagger$ . FOR EACH VERSION OF RDG-MAENS, IF ITS MEAN TOTAL COST IS STATISTICALLY SIGNIFICANTLY SMALLER THAN THAT OF MAENS, THEN, IT IS MARKED IN BOLD. OTHERWISE, THE MEAN TOTAL COST OF MAENS IS MARKED IN BOLD IF IT IS STATISTICALLY SIGNIFICANTLY SMALLER THAN THAT OF ALL THE VERSIONS OF RDG-MAENS

Name ( $ V , E , Z ,\tau$ )		GLS		Ant-CARP		MAENS		RDG-MAENS											
								$g = 2$						$g = 3$					
								$\alpha = 1$		$\alpha = 5$		$\alpha = 10$		$\alpha = 1$		$\alpha = 5$		$\alpha = 10$	
								[20]		[25]		[23]		$\alpha = 1$		$\alpha = 5$		$\alpha = 10$	
		Cost	Time	Median	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time
C01	(69,98,79,9)	1660	48	1705	42	1691	94	1687	102	1674 <sup>†</sup>	103	1696	75	1689	107	1697	110	1754	91
C02	(48,66,53,7)	1095	1	1095	16	1095 <sup>†</sup>	49	1103	68	1108	66	1132	59	1132	81	1153	83	1206	70
C03	(46,64,51,6)	925	25	925	18	927 <sup>†</sup>	52	936	69	944	68	959	52	957	80	996	79	1046	65
C04	(60,84,72,8)	1340	42	1340	36	1341 <sup>†</sup>	77	1346	95	1361	94	1370	74	1350	95	1398	99	1416	83
C05	(56,79,65,10)	2475	34	2540	25	2495 <sup>†</sup>	65	2507	84	2510	85	2511	62	2538	100	2542	101	2541	80
C06	(38,55,51,6)	895	24	895	17	902	47	899 <sup>†</sup>	70	943	71	958	50	955	79	985	79	1023	69
C07	(54,70,52,8)	1795	24	1795	16	1795 <sup>†</sup>	51	1818	71	1834	70	1841	54	1837	84	1852	84	1885	70
C08	(66,88,63,8)	1730	34	1730	18	1732 <sup>†</sup>	62	1736	79	1785	76	1781	56	1756	89	1792	89	1829	74
C09	(76,117,97,12)	1825	65	1860	67	1847	137	1841 <sup>†</sup>	136	1856	136	1884	96	1869	128	1933	127	1969	97
C10	(60,82,55,9)	2290	27	2305	18	2284 <sup>†</sup>	52	2327	73	2320	73	2343	52	2341	84	2358	85	2380	73
C11	(83,118,94,10)	1815	62	1820	63	1846 <sup>†</sup>	130	1869	138	1873	141	1867	98	1886	126	1905	128	1924	97
C12	(62,88,72,9)	1610	42	1610	30	1610 <sup>†</sup>	78	1619	93	1630	90	1641	60	1638	98	1662	100	1679	77
C13	(40,60,52,7)	1110	26	1110	17	1110 <sup>†</sup>	47	1119	70	1132	73	1139	48	1133	82	1169	84	1190	72
C14	(58,79,57,8)	1680	29	1680	17	1682 <sup>†</sup>	56	1683	78	1707	78	1722	51	1694	91	1727	91	1723	75
C15	(97,140,107,11)	1860	81	1880	93	1889	170	1882	164	1879 <sup>†</sup>	166	1903	110	1909	149	1953	171	2007	128
C16	(32,42,32,3)	585	18	585	8	585 <sup>†</sup>	30	592	48	629	49	626	34	679	55	676	54	681	54
C17	(43,56,42,7)	1610	20	1610	11	1610 <sup>†</sup>	37	1633	64	1661	63	1677	42	1683	73	1734	75	1750	67
C18	(93,133,121,11)	2410	83	2390	117	2406 <sup>†</sup>	222	2414	208	2422	207	2431	130	2413	170	2446	180	2458	131
C19	(62,84,61,6)	1395	31	1400	23	1414 <sup>†</sup>	62	1424	77	1442	79	1456	54	1434	85	1474	85	1534	66
C20	(45,64,53,5)	665	0	665	15	666 <sup>†</sup>	49	677	68	677	68	690	48	698	76	714	74	736	60
C21	(60,84,76,8)	1725	48	1725	41	1725 <sup>†</sup>	86	1747	99	1752	103	1794	68	1757	97	1847	99	1886	68
C22	(56,76,43,4)	1070	0	1070	11	1070 <sup>†</sup>	38	1070 <sup>†</sup>	65	1081	66	1086	45	1086	72	1124	70	1156	51
C23	(78,109,92,8)	1690	56	1710	56	1699 <sup>†</sup>	126	1705	134	1739	147	1740	83	1730	116	1751	120	1797	81
C24	(77,115,84,7)	1360	46	1360	49	1363 <sup>†</sup>	103	1371	108	1377	79	1390	69	1379	105	1404	105	1464	86
C25	(37,50,38,5)	905	0	905	9	905 <sup>†</sup>	33	920	56	931	45	943	41	961	69	962	68	1066	57
Avg.		1501	35	1508	33	1507	78	1517	93	1530	92	1543	64	1540	96	1570	98	1604	78

compared algorithm is simply normalized with respect to the CPU frequency, as has been done in the previous studies [12], [14], [23], [25]. The normalized computational time of each compared algorithm is obtained as follows:

$$\bar{\rho}(\text{algorithm}) = \rho(\text{algorithm}) \cdot \frac{\psi(\text{algorithm})}{\psi(\text{RDG-MAENS})} \quad (14)$$

where  $\bar{\rho}(\text{algorithm})$  and  $\rho(\text{algorithm})$  are the normalized computational time and original computational time obtained from publications, and  $\psi(\text{RDG-MAENS})$  and  $\psi(\text{algorithm})$  are the CPU frequency of RDG-MAENS and the algorithm, respectively. In the experiments, RDG-MAENS and MAENS were run on an Intel Core i7-2600 @3.4 GHz, using only one core. GLS was run on Intel Pentium II @500 MHz. Ant-CARP was run on Intel Pentium III @1 GHz. VNS was run on Intel Pentium IV @3.6 GHz. ILS-RVND was run on Intel Core i5 @3.2 GHz. Thus, the computational time of GLS, Ant-CARP, VNS, and ILS-RVND are multiplied by 0.5/3.4, 1/3.4, 3.6/3.4, and 3.2/3.4, respectively. Normalization is not needed for MAENS as it was run on the same computational platform as RDG-MAENS. As shown in [47], even identical CPUs may perform radically differently under different cache or RAM capacities. Therefore, the above normalization is only indicative. However, since MAENS and RDG-MAENS

were run on the same computational platform, the comparison between them is guaranteed to be fair and thus provides a meaningful comparison on computational time.

In the tables, for each instance, the minimal mean total cost between MAENS and RDG-MAENS is marked with  $\dagger$ . Besides, for each instance and each version of RDG-MAENS, the 30 total costs are compared with that of MAENS using Wilcoxon's rank sum test [48] at the significance level of 0.05. If they are significantly smaller, then the corresponding mean total cost is marked in bold. Otherwise, the mean total cost of MAENS is marked in bold if it is significantly smaller than that of all the versions of RDG-MAENS. Note that for Beullens' D16, D22, and E25 instances,  $\tau = 2$  and there are not enough routes to be divided into three groups. Hence, RDG-MAENS with  $g = 3$  were not applicable to them, and the corresponding results were marked with  $-$ .

From Tables II–V, it can be seen that on almost all Beullens' instances, MAENS showed better average performance than RDG-MAENS. It obtained smaller mean total cost than RDG-MAENS on 92 out of the total 100 Beullens' instances (21 C, 24 D, 22 E, and 25 F instances), 63 of which were statistically significant (12 C, 19 D, 13 E, and 19 F instances). There are only two instances (C01 and C15) on which (2, 5) performed significantly better than MAENS. Overall, MAENS obtained

TABLE III

AVERAGE (MEAN OR MEDIAN) TOTAL COST AND COMPUTATIONAL TIME OF THE COMPARED ALGORITHMS ON BEULLENS' D TEST SET. FOR EACH INSTANCE, THE MINIMAL TOTAL COST OF MAENS AND RDG-MAENS IS MARKED WITH  $\dagger$ . THE MEAN TOTAL COST OF MAENS IS MARKED IN BOLD IF IT IS STATISTICALLY SIGNIFICANTLY SMALLER THAN THAT OF ALL THE VERSIONS OF RDG-MAENS

Name	( V , E , Z ,τ)	GLS	Ant-CARP	MAENS	RDG-MAENS																	
					g = 2												g = 3					
					[20]		[25]		[23]		α = 1		α = 5		α = 10		α = 1		α = 5		α = 10	
					Cost	Time	Median	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time
D01	(69,98,79,5)	725	2	745	59	744	142	744 <sup>†</sup>	114	802	136	834	93	758	103	879	114	1009	85			
D02	(48,66,53,4)	480	0	480	21	480 <sup>†</sup>	68	483	69	552	71	553	51	545	72	648	73	711	51			
D03	(46,64,51,3)	415	0	415	20	415 <sup>†</sup>	67	421	63	439	62	498	49	739	67	759	69	755	47			
D04	(60,84,72,4)	615	0	615	50	615 <sup>†</sup>	115	615	91	624	98	645	74	690	88	757	89	789	56			
D05	(56,79,65,5)	1040	0	1040	35	1040 <sup>†</sup>	93	1062	85	1080	86	1108	65	1070	89	1139	90	1209	65			
D06	(38,55,51,3)	485	0	485	19	485 <sup>†</sup>	66	535	61	587	60	584	46	716	67	702	69	727	50			
D07	(54,70,52,4)	835	20	855	17	839 <sup>†</sup>	71	884	67	900	67	905	49	983	74	1050	73	1141	63			
D08	(66,88,63,4)	685	29	685	23	691 <sup>†</sup>	85	718	74	752	77	735	57	751	78	785	76	809	65			
D09	(76,117,97,6)	680	0	680	77	680 <sup>†</sup>	212	688	138	710	139	731	104	695	117	832	126	930	88			
D10	(60,82,55,5)	910	22	910	23	910 <sup>†</sup>	69	933	63	1033	67	1067	54	951	78	1157	81	1172	66			
D11	(83,118,94,5)	930	54	935	81	937 <sup>†</sup>	210	948	136	950	130	1013	87	976	119	1009	118	1090	92			
D12	(62,88,72,5)	680	0	680	36	680 <sup>†</sup>	117	697	89	705	76	754	62	744	89	777	96	803	75			
D13	(40,60,52,4)	690	0	690	22	690 <sup>†</sup>	66	693	63	790	52	817	62	769	79	825	75	907	62			
D14	(58,79,57,4)	930	26	930	22	930 <sup>†</sup>	79	947	68	971	47	1019	54	984	74	1073	72	1080	59			
D15	(97,140,107,6)	910	54	920	120	919 <sup>†</sup>	266	921	166	968	119	965	122	943	137	1027	130	1054	117			
D16	(32,42,32,2)	170	0	170	8	170 <sup>†</sup>	35	391	41	387	33	383	38	-	-	-	-	-	-			
D17	(43,56,42,4)	675	0	675	14	675 <sup>†</sup>	51	685	56	690	40	704	47	736	76	767	76	829	68			
D18	(93,133,121,6)	930	0	930	153	939 <sup>†</sup>	342	951	203	989	120	1024	130	973	163	1093	174	1074	117			
D19	(62,84,61,3)	680	23	680	30	680 <sup>†</sup>	92	685	72	720	55	730	64	745	74	788	75	791	56			
D20	(45,64,53,3)	415	0	415	22	415 <sup>†</sup>	72	461	64	490	43	503	52	660	65	682	66	687	57			
D21	(60,84,76,4)	805	37	810	48	826 <sup>†</sup>	137	852	98	869	62	881	75	911	81	1018	80	1055	70			
D22	(56,76,43,2)	690	0	690	13	690 <sup>†</sup>	53	734	51	756	37	741	41	-	-	-	-	-	-			
D23	(78,109,92,4)	735	49	735	63	752 <sup>†</sup>	189	765	134	754	92	757	108	810	114	823	125	922	86			
D24	(77,115,84,4)	670	36	670	73	675 <sup>†</sup>	157	695	101	738	69	714	78	734	96	824	92	881	62			
D25	(37,50,38,3)	410	0	410	10	410 <sup>†</sup>	43	415	50	479	38	528	45	711	68	707	67	707	55			
Avg.		688	14	690	42	691	116	717	89	749	75	768	68	-	-	-	-	-	-			

similar average performance with Ant-CARP in terms of the average mean or median total cost over all Beullens' instances. This implies that MAENS is still one of the best performing algorithms on Beullens' sets. Additionally, it is observed that MAENS showed better performance than RDG-MAENS on more D and F instances. Note that the D and F instances have smaller  $\tau$ 's than the C and E instances. Therefore, MAENS performed better on the instances with smaller  $\tau$ 's. For RDG-MAENS, smaller  $g$  and  $\alpha$  generally have better performance.

On Beullens' instances, RDG-MAENS did not always have a smaller computational time than MAENS. In fact, for many instances (e.g., C06 and C17), MAENS had a smaller computational time than RDG-MAENS. The reason can be explained as follows: most Beullens' instances are small or medium scaled instances. When the problem size is not large, the computational effort for solving each subcomponent is nearly the same as solving the overall problem. In this case, the total computational time for solving all the  $g$  components can be larger than that of solving the problem itself. Therefore, the decomposition strategy is not effective when the problem size is not large. Finally, the computational time of RDG-MAENS was much larger than that of the compared state-of-the-art algorithms. The reason is likely to be that RDG-MAENS did not employ the lower bound, and thus, always stopped after the maximal number of generations. However, the other

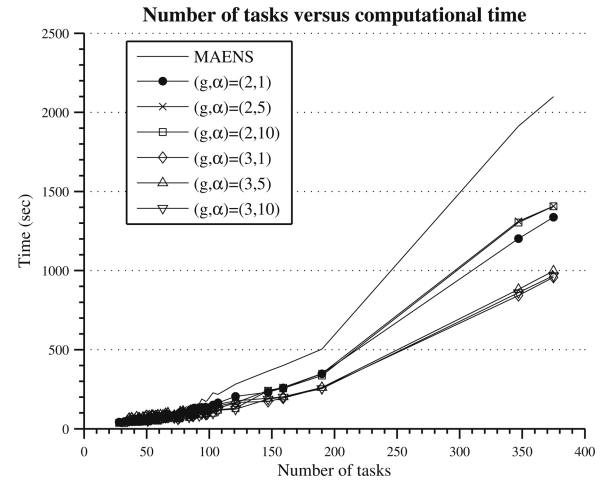


Fig. 3. Average computational time versus the number of tasks over all the test instances for each compared algorithm.

algorithms may stop much earlier than the maximal number of generations when reaching the lower bound, especially for the simple instances.

On the egl instances, which are shown in Table VI, the average performance of RDG-MAENS became much better, especially on the second half of the set. More specifically,

TABLE IV

AVERAGE (MEAN OR MEDIAN) TOTAL COST AND COMPUTATIONAL TIME OF THE COMPARED ALGORITHMS ON BEULLENS' E TEST SET. FOR EACH INSTANCE, THE MINIMAL TOTAL COST OF MAENS AND RDG-MAENS IS MARKED WITH  $\dagger$ . THE MEAN TOTAL COST OF MAENS IS MARKED IN BOLD IF IT IS STATISTICALLY SIGNIFICANTLY SMALLER THAN THAT OF ALL THE VERSIONS OF RDG-MAENS

Name ( $ V ,  E ,  Z , \tau$ )	GLS	Ant-CARP	MAENS	RDG-MAENS											
				$g = 2$						$g = 3$					
				[20]		[25]		[23]		$\alpha = 1$		$\alpha = 5$		$\alpha = 10$	
				Cost	Time	Median	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time
E01 (73,105,85,10)	1940	50	1945	41	<b>1941<math>\dagger</math></b>	114	1955	120	1951	82	1973	74	1971	117	1984
E02 (58,81,58,8)	1610	28	1610	20	<b>1615<math>\dagger</math></b>	78	1632	76	1640	51	1652	50	1660	84	1674
E03 (46,61,47,5)	750	0	750	14	<b>750<math>\dagger</math></b>	59	761	61	788	47	780	44	765	69	824
E04 (70,99,77,9)	1610	48	1675	36	1660 $\dagger$	135	1675	107	1663	74	1678	65	1678	104	1703
E05 (68,94,61,9)	2170	31	2220	23	2209 $\dagger$	90	2221	84	2221	62	2237	54	2275	92	2248
E06 (49,66,43,5)	670	0	670	11	670 $\dagger$	53	670 $\dagger$	59	696	45	730	41	671	72	767
E07 (73,94,50,8)	1900	24	1900	15	<b>1900<math>\dagger</math></b>	71	1910	72	1922	51	1925	44	1925	81	1931
E08 (74,98,59,9)	2150	33	2150	22	<b>2150<math>\dagger</math></b>	91	2158	82	2167	55	2179	49	2165	91	2182
E09 (91,141,103,12)	2250	65	2295	70	2291 $\dagger$	224	2307	156	2313	108	2327	97	2319	142	2346
E10 (56,76,49,7)	1690	0	1690	13	<b>1690<math>\dagger</math></b>	67	1706	67	1726	52	1756	42	1735	77	1793
E11 (80,113,94,10)	1850	62	1860	69	1885 $\dagger$	200	1904	135	1905	96	1910	91	1929	127	1911
E12 (74,103,67,9)	1710	39	1760	27	1753	103	1758	88	1752 $\dagger$	61	1761	63	1788	97	1770
E13 (49,73,52,7)	1325	26	1325	16	1325 $\dagger$	74	1327	69	1325	50	1338	55	1348	84	1380
E14 (53,72,55,8)	1810	28	1810	18	<b>1810<math>\dagger</math></b>	83	1835	75	1834	59	1843	60	1843	88	1851
E15 (85,126,107,9)	1610	74	1610	95	1613	248	1618	165	1611 $\dagger$	111	1618	116	1634	146	1632
E16 (60,80,54,7)	1825	29	1825	21	<b>1825<math>\dagger</math></b>	84	1850	75	1881	58	1913	62	1874	85	2034
E17 (38,50,36,5)	1290	1	1290	11	1291 $\dagger$	46	1299	54	1303	44	1315	47	1299	68	1351
E18 (78,110,88,8)	1610	53	1610	63	1612 $\dagger$	176	1637	132	1625	92	1647	100	1634	117	1661
E19 (77,103,66,6)	1435	31	1435	30	1437 $\dagger$	106	1453	86	1449	67	1452	73	1468	94	1510
E20 (56,80,63,7)	990	34	990	31	<b>990<math>\dagger</math></b>	100	998	83	1004	58	1039	60	1007	90	1072
E21 (57,82,72,7)	1705	43	1760	40	<b>1743<math>\dagger</math></b>	128	1771	99	1770	67	1793	70	1785	95	1849
E22 (54,73,44,5)	1185	19	1185	13	<b>1185<math>\dagger</math></b>	57	1200	63	1194	47	1199	49	1202	73	1238
E23 (93,130,89,8)	1430	58	1435	62	1462	177	1473	125	1460 $\dagger$	93	1469	92	1463	109	1475
E24 (97,142,86,8)	1785	53	1785	57	<b>1790<math>\dagger</math></b>	168	1825	124	1834	81	1846	82	1835	125	1877
E25 (26,35,28,4)	655	0	655	8	<b>655<math>\dagger</math></b>	36	717	46	752	38	734	38	769	62	852
Avg.	1558	33	1570	33	1570	111	1586	92	1591	66	1605	65	1602	96	1637

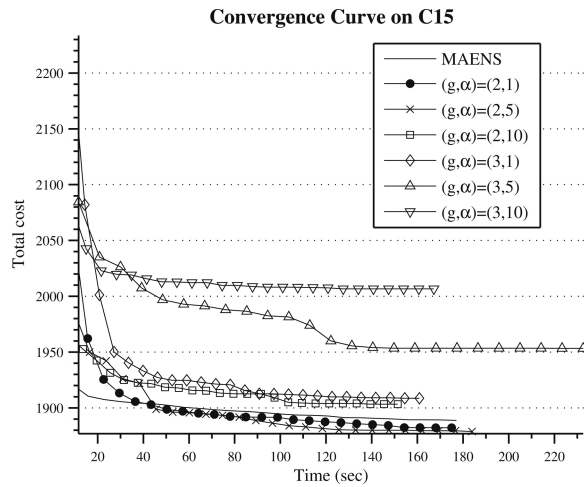


Fig. 4. Convergence curves of RDG-MAENS on Beullens' C15.

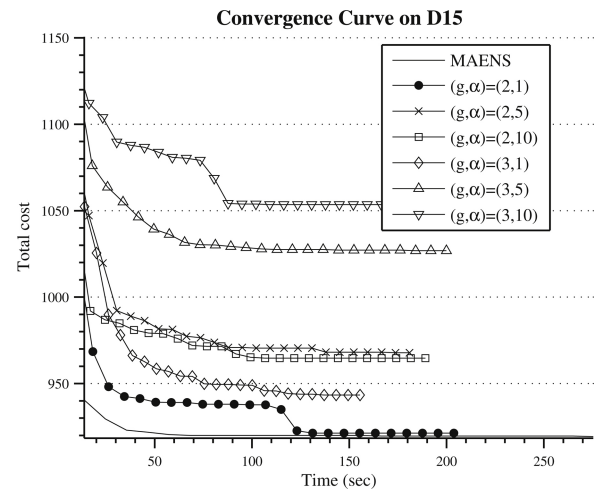


Fig. 5. Convergence curves of RDG-MAENS on Beullens' D15.

from s2-A to s4-C, at least one version of RDG-MAENS showed significantly better performance than MAENS. On these nine instances, all the versions of RDG-MAENS with  $g = 2$  performed no worse than MAENS and other state-of-the-art algorithms. In other words, RDG-MAENS outperformed MAENS and other state-of-the-art algorithms on the instances

with  $|Z| \geq 147$  and  $\tau \geq 14$ . In addition, there is an obvious trend that the computational time decreases when  $g$  increases.

Finally, as shown in Table VII, on the EGL-G instances, it is obvious that RDG-MAENS performed significantly better than MAENS and ILS-RVND. In terms of the average results and computational time over the ten EGL-G instances, (3, 5) performed the best, as it obtained nearly the best average



TABLE V

AVERAGE (MEAN OR MEDIAN) TOTAL COST AND COMPUTATIONAL TIME OF THE COMPARED ALGORITHMS ON BEULLENS' F TEST SET. FOR EACH INSTANCE, THE MINIMAL TOTAL COST OF MAENS AND RDG-MAENS IS MARKED WITH  $\dagger$ . THE MEAN TOTAL COST OF MAENS IS MARKED IN BOLD IF IT IS STATISTICALLY SIGNIFICANTLY SMALLER THAN THAT OF ALL THE VERSIONS OF RDG-MAENS

Name		( V , E , Z ,τ)		GLS		Ant-CARP		MAENS		RDG-MAENS																	
										g = 2						g = 3											
										[20]		[25]		[23]		α = 1		α = 5		α = 10		α = 1		α = 5		α = 10	
										Cost	Time	Median	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time
F01	(73,105,85,5)	1065	1	1065	46	1070 <sup>†</sup>	173	1086	122	1103	88	1124	95	1146	109	1182	113	1198	100								
F02	(58,81,58,4)	920	0	920	22	920 <sup>†</sup>	78	929	68	963	51	987	54	1002	75	1044	75	1093	71								
F03	(46,61,47,3)	400	0	400	19	400 <sup>†</sup>	53	419	55	524	48	541	53	724	66	741	71	743	53								
F04	(70,99,77,5)	940	36	955	54	955 <sup>†</sup>	137	966	102	969	76	974	82	976	104	1082	108	1091	77								
F05	(68,94,61,5)	1180	0	1180	32	1180 <sup>†</sup>	83	1180	74	1219	63	1205	70	1199	88	1331	96	1391	77								
F06	(49,66,43,3)	490	0	490	16	490 <sup>†</sup>	45	515	47	530	41	557	46	670	65	685	71	674	60								
F07	(73,94,50,4)	1080	0	1080	16	1080 <sup>†</sup>	63	1142	57	1164	46	1201	49	1259	82	1323	81	1377	69								
F08	(74,98,59,5)	1145	27	1145	27	1145 <sup>†</sup>	82	1149	68	1160	55	1170	54	1167	79	1225	82	1222	67								
F09	(91,141,103,6)	1145	1	1225	125	1172 <sup>†</sup>	231	1203	145	1213	109	1225	106	1232	133	1277	133	1349	100								
F10	(56,76,49,4)	1010	0	1010	20	1010 <sup>†</sup>	63	1015	55	1062	44	1078	50	1128	74	1258	72	1279	71								
F11	(80,113,94,5)	1015	1	1045	78	1025 <sup>†</sup>	212	1049	130	1067	89	1094	95	1089	114	1140	117	1179	95								
F12	(74,103,67,5)	910	34	975	40	938 <sup>†</sup>	101	966	81	1042	67	1064	74	996	95	1082	101	1172	76								
F13	(49,73,52,4)	835	0	835	22	835 <sup>†</sup>	70	839	58	866	52	896	55	882	77	983	81	1025	59								
F14	(53,72,55,4)	1025	3	1025	22	1047 <sup>†</sup>	77	1091	62	1111	53	1122	54	1172	81	1213	81	1224	67								
F15	(85,126,107,5)	945	0	945	120	945 <sup>†</sup>	186	954	161	962	126	1014	135	976	143	1011	162	1069	115								
F16	(60,80,54,4)	775	0	775	25	775 <sup>†</sup>	51	789	66	921	52	1015	52	959	79	1079	81	1121	63								
F17	(38,50,36,3)	605	0	605	9	605 <sup>†</sup>	27	605 <sup>†</sup>	43	610	38	640	40	931	64	1020	62	977	55								
F18	(78,110,88,4)	850	40	850	72	852 <sup>†</sup>	122	894	131	946	90	954	99	967	104	1040	105	1078	89								
F19	(77,103,66,3)	725	23	725	37	725 <sup>†</sup>	71	726	102	751	68	762	75	803	78	847	79	846	73								
F20	(56,80,63,4)	610	0	610	42	615 <sup>†</sup>	63	620	81	678	71	734	75	655	79	775	87	803	73								
F21	(57,82,72,4)	905	1	905	54	905 <sup>†</sup>	85	919	97	969	74	990	76	1039	84	1154	84	1218	73								
F22	(54,73,44,3)	790	0	790	19	790 <sup>†</sup>	38	796	61	808	49	797	52	873	66	935	59	944	61								
F23	(93,130,89,4)	725	47	730	69	729 <sup>†</sup>	117	741	119	741	97	758	99	772	98	825	111	902	103								
F24	(97,142,86,4)	975	4	975	72	1003 <sup>†</sup>	119	1007	118	1028	89	1051	88	1128	96	1235	70	1278	82								
F25	(26,35,28,2)	430	0	430	7	430 <sup>†</sup>	24	553	38	529	34	559	38	-	-	-	-	-	-								
Avg.		860	9	868	43	866	95	886	86	917	67	940	71	-	-	-	-	-	-								

results (only slightly worse than that of (2, 10)) with the smallest computational time. It is also obvious that a larger  $g$  leads to a much smaller computational time.

In summary, the average performance of RDG-MAENS improves as  $|Z|$  and  $\tau$  increases in terms of both solution quality and speed. When  $|Z|$  and  $\tau$  are large (e.g.,  $|Z| \geq 147$ ,  $\tau \geq 14$ ), RDG-MAENS can obtain significantly better solutions in a much shorter time than MAENS. On the other hand, the previous studies have shown that larger  $|Z|$  (problem size) and  $\tau$  (tightness of the capacity constraint) lead to a higher level of difficulty of the problem. Therefore, the efficacy of the RDG decomposition scheme in solving large and difficult CARP instances has been verified.

To better understand the scalability of RDG-MAENS, we plot the average computational time versus the number of tasks over all the test instances for MAENS and each version of RDG-MAENS. The results are shown in Fig. 3, where the  $x$ -axis represents the number of tasks, and the  $y$ -axis indicates the average computational time in seconds. Note that there are multiple instances with the same number of tasks ( $x$ -axis value). In this case, the average of the  $y$ -axis values of these instances is computed to represent the average computational time for the corresponding number of tasks. From the figure, it is obvious that as  $g$  increases, the scalability improves significantly. When the number of tasks is no larger than

50, the average computational time of RDG-MAENS is not different from that of MAENS. Then, as the number of tasks increases, the effect of  $g$  on the computational time increases. As a result, the curve of MAENS is the steepest, while the versions of RDG-MAENS with  $g = 3$  have the flattest curves. Given the same  $g$ , different values of  $\alpha$  lead to similar curves. This implies that the scalability of RDG-MAENS depends largely on  $g$ , but not much on  $\alpha$ .

Table VIII shows the mean of the best total costs of RDG-MAENS and the other state-of-the-art algorithms over the instances of each test set. One can see that on Beullens' sets, the best performance of MAENS is no worse than that of the other state-of-the-art algorithms. The best performance of (2, 1) is also as good as that of the state-of-the-art algorithms on Beullens' C, E, and F sets. On the egl set, all the versions of RDG-MAENS showed nearly the same best performance as the state-of-the-art results. However, on the EGL-G set, all the versions of RDG-MAENS performed much better than the state-of-the-art results in the best case.

Table IX shows the results on the instances where the best-known solutions were updated by RDG-MAENS. **BK** represents the previously best-known results of the instances, which were obtained from [10], [14], [25], [40]. For each instance, the new best-known result is marked in bold. One can see that the best-known results were updated for all the EGL-G

TABLE VI

AVERAGE (MEAN OR MEDIAN) TOTAL COST AND COMPUTATIONAL TIME OF THE COMPARED ALGORITHMS ON THE EGL TEST SET. FOR EACH INSTANCE, THE MINIMAL TOTAL COST OF MAENS AND RDG-MAENS IS MARKED WITH  $\dagger$ . FOR EACH VERSION OF RDG-MAENS, IF ITS MEAN TOTAL COST IS STATISTICALLY SIGNIFICANTLY SMALLER THAN THAT OF MAENS, THEN IT IS MARKED IN BOLD. OTHERWISE, THE MEAN TOTAL COST OF MAENS IS MARKED IN BOLD IF IT IS STATISTICALLY SIGNIFICANTLY SMALLER THAN THAT OF ALL THE VERSIONS OF RDG-MAENS

Name ( $ V ,  E ,  Z , \tau$ )	VNS		Ant-CARP		MAENS		RDG-MAENS											
							$g = 2$						$g = 3$					
							$\alpha = 1$		$\alpha = 5$		$\alpha = 10$		$\alpha = 1$		$\alpha = 5$		$\alpha = 10$	
							Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time
e1-A (77,98,51,5)	3548	5	3548	0	3548 $\dagger$	56	3555	42	3551	40	3568	40	3565	36	3623	37	3633	37
e1-B (77,98,51,7)	4522	59	4539	19	4501 $\dagger$	57	4526	44	4530	43	4538	44	4537	42	4554	41	4560	40
e1-C (77,98,51,10)	5608	687	5595	21	5598 $\dagger$	62	5615	47	5679	52	5706	51	5647	47	5846	45	5927	43
e2-A (77,98,72,7)	5024	381	5018	45	5018 $\dagger$	108	5029	68	5046	70	5042	68	5032	57	5077	56	5123	55
e2-B (77,98,72,10)	6335	732	6344	38	6331 $\dagger$	106	6342	71	6346	73	6373	73	6370	64	6389	65	6386	63
e2-C (77,98,72,14)	8356	457	8335	41	8339 $\dagger$	116	8360	75	8364	80	8365	77	8395	70	8414	76	8448	74
e3-A (77,98,87,8)	5898	208	5898	10	5898 $\dagger$	153	5912	89	5928	89	5923	95	5948	74	5955	76	6002	79
e3-B (77,98,87,12)	7806	502	7787	62	7790 $\dagger$	154	7815	93	7830	99	7835	99	7835	79	7843	86	7846	86
e3-C (77,98,87,17)	10322	584	10292	57	10316 $\dagger$	147	10324	98	10327	110	10342	106	10344	84	10346	89	10382	86
e4-A (77,98,98,9)	6459	518	6464	92	6472 $\dagger$	187	6475	105	6489	105	6517	104	6503	86	6508	86	6523	84
e4-B (77,98,98,14)	9016	523	9047	81	9009 $\dagger$	189	9033	111	9047	118	9040	115	9050	92	9057	99	9073	96
e4-C (77,98,98,19)	11750	533	11645	84	11626 $\dagger$	155	11651	116	11655	123	11694	120	11670	100	11663	107	11675	105
s1-A (140,190,75,7)	5018	193	5018	46	5021 $\dagger$	96	5063	65	5054	71	5050	71	5071	59	5223	60	5358	58
s1-B (140,190,75,10)	6388	385	6388	46	6412 $\dagger$	112	6424	68	6435	68	6432	70	6471	65	6481	65	6612	60
s1-C (140,190,75,14)	8518	383	8518	44	8518 $\dagger$	113	8540	73	8535	75	8555	76	8567	69	8622	69	8631	68
s2-A (140,190,147,14)	9998	848	9974	209	9979	433	9977	225	9970	259	9960 $\dagger$	261	9960	174	9977	211	9971	209
s2-B (140,190,147,20)	13176	857	13283	227	13198	337	13183 $\dagger$	223	13212	241	13229	244	13205	169	13214	183	13235	182
s2-C (140,190,147,27)	16552	713	16558	201	16510	303	16491 $\dagger$	229	16499	233	16493	235	16505	182	16512	186	16529	187
s3-A (140,190,159,15)	10291	794	10306	248	10295	483	10292	256	10272 $\dagger$	272	10278	262	10290	188	10312	215	10430	204
s3-B (140,190,159,22)	13829	606	13890	262	13848	397	13791	257	13775 $\dagger$	271	13804	274	13825	191	13847	201	13906	196
s3-C (140,190,159,29)	17328	763	17304	226	17291	349	17283 $\dagger$	256	17293	255	17319	254	17301	206	17334	202	17361	202
s4-A (140,190,190,19)	12440	721	12439	455	12404	588	12392 $\dagger$	355	12397	387	12394	369	12406	252	12415	264	12440	261
s4-B (140,190,190,27)	16410	450	16502	608	16437	485	16376	345	16378	350	16386	348	16375 $\dagger$	262	16434	261	16439	258
s4-C (140,190,190,35)	20732	885	20731	427	20703	438	20675	314	20696	317	20671	312	20682	266	20692	263	20667 $\dagger$	257
Avg.	9805	533	9809	148	9794	234	9797	151	9805	158	9813	157	9815	121	9847	127	9882	125

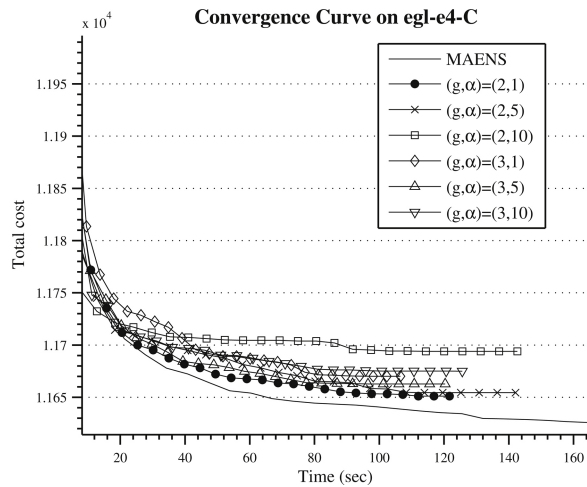


Fig. 6. Convergence curves of the compared algorithms on egl-e4-C.

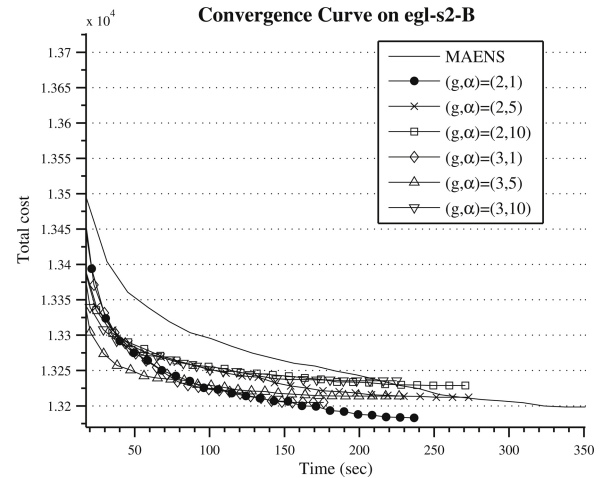


Fig. 7. Convergence curves of the compared algorithms on egl-s2-B.

instances. The other instances (C11, E09, E11, s2-B and s4-B) also have larger  $|Z|$  and  $\tau$  than most instances in their own test sets. Therefore, the best performance of RDG-MAENS showed a similar pattern as that of its average performance. That is, RDG-MAENS showed its competitiveness on the instances with large  $|Z|$ 's and  $\tau$ 's.

A more illustrative comparison is also made on the convergence curves of MAENS and RDG-MAENS for some representative instances, which are shown in Figs. 4–9. In the figures, the x-axis represents the computational time in seconds, and the y-axis is the average total cost of the best-so-far solutions over the 30 independent runs. Figs. 4 and 5 show the results on Beullens' C15 and D15 instances, which



TABLE VII

MEAN TOTAL COST AND COMPUTATIONAL TIME OF THE COMPARED ALGORITHMS ON THE EGL-G TEST SET. FOR EACH INSTANCE, THE MINIMAL TOTAL COST OF MAENS AND RDG-MAENS IS MARKED WITH  $\dagger$ . FOR EACH VERSION OF RDG-MAENS, IF ITS MEAN TOTAL COST IS STATISTICALLY SIGNIFICANTLY SMALLER THAN THAT OF MAENS, THEN, IT IS MARKED IN BOLD

Name ( $ V ,  E ,  Z , \tau$ )	ILS-RVND		MAENS		RDG-MAENS											
					$g = 2$						$g = 3$					
					[14]		[23]		$\alpha = 1$		$\alpha = 5$		$\alpha = 10$		$\alpha = 1$	
					Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time	Mean	Time
<b>G1-A</b> (255,375,347,20)	1010937	1023	1009302	2453	1008295	1426	1007956	1643	1007619	1628	1009720	933	1007223 $\dagger$	1049	1014068	985
<b>G1-B</b> (255,375,347,25)	1137142	916	1128114	2058	1127223	1288	<b>1124191</b>	1407	<b>1122863<math>\dagger</math></b>	1390	1127022	860	<b>1124751</b>	901	1127199	888
<b>G1-C</b> (255,375,347,30)	1266577	860	1255709	1841	1254508	1203	<b>1251535</b>	1266	<b>1250174<math>\dagger</math></b>	1264	1255083	829	<b>1251718</b>	843	1255139	846
<b>G1-D</b> (255,375,347,35)	1406929	834	1390034	1686	<b>1386729</b>	1066	<b>1384225</b>	1141	1386120	1145	1387528	818	<b>1382695<math>\dagger</math></b>	834	1387294	822
<b>G1-E</b> (255,375,347,40)	1554220	833	1535511	1566	<b>1531185</b>	995	<b>1526998</b>	1060	<b>1525629</b>	1068	<b>1529293</b>	776	<b>1524890<math>\dagger</math></b>	781	<b>1530237</b>	775
<b>G2-A</b> (255,375,375,22)	1118363	1507	1109376	2593	1109513	1571	<b>1106823</b>	1674	<b>1104944<math>\dagger</math></b>	1675	1110053	1045	1108916	1140	1110665	1073
<b>G2-B</b> (255,375,375,27)	1233721	1376	1225361	2268	<b>1222816</b>	1448	<b>1221214<math>\dagger</math></b>	1515	<b>1221429</b>	1511	1225438	990	<b>1222183</b>	1055	1224629	1002
<b>G2-C</b> (255,375,375,32)	1374480	1019	1358398	2016	<b>1356468</b>	1313	<b>1353777</b>	1382	1355548	1373	1356212	947	<b>1353118<math>\dagger</math></b>	962	<b>1355141</b>	947
<b>G2-D</b> (255,375,375,37)	1515119	940	1500415	1881	<b>1496448</b>	1203	<b>1492116</b>	1272	<b>1492063</b>	1268	1496652	910	<b>1491013</b>	923	<b>1489114<math>\dagger</math></b>	920
<b>G2-E</b> (255,375,375,42)	1658378	900	1641260	1781	<b>1636683</b>	1119	<b>1632008</b>	1187	<b>1629002<math>\dagger</math></b>	1181	<b>1632348</b>	886	<b>1629310</b>	907	<b>1629837</b>	893
<b>Avg.</b>	1327587	1021	1315348	2014	1312987	1263	1310084	1354	1309539	1350	1312935	899	1309582	940	1312332	915

TABLE VIII

MEAN OF BEST TOTAL COST OF THE COMPARED ALGORITHMS OVER THE INSTANCES OF EACH TEST SET

Name	GLS	Ant-CARP	VNS	ILS-RVND	MAENS	RDG-MAENS					
						$g = 2$			$g = 3$		
						$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$
	[20]	[25]	[10]	[14]	[23]						
Beullens' C	1501	1499	-	-	1498	1499	1498	1499	1502	1503	1505
Beullens' D	688	688	-	-	688	696	697	697	-	-	-
Beullens' E	1558	1556	-	-	1557	1556	1555	1556	1556	1560	1562
Beullens' F	860	860	-	-	860	860	862	861	-	-	-
<i>egl</i>	-	9747	9752	-	9752	9746	9748	9749	9748	9752	9756
EGL-G	-	-	-	1318665	1305158	1303829	1300193	1300518	1302895	1299428	1300673

have the same graph. However, C15 has a larger  $\tau$ . One can see that the curves of RDG-MAENS tend to be closer to that of MAENS on C15 than D15, which implies that RDG-MAENS performed better on C15 than D15. Figs. 6 and 7 are the results on the *egl* e4-C and s2-B, which have the similar  $\tau$ 's. However, the number of tasks is much larger in s2-B than in e4-C. It is obvious that RDG-MAENS performed much better on s2-B. Figs. 8 and 9 give the results on EGL-G2-A and EGL-G2-D, both are LSCARP with larger  $\tau$ 's. On these instances with large  $\tau$ 's, there is a clear advantage of RDG-MAENS over MAENS as well. However, the best versions of RDG-MAENS are different. On EGL-G2-A, (2, 10) performed the best, while on EGL-G2-D, (3, 10) showed the best performance. In short, the representative illustrations show that RDG performed better on instances with larger number of tasks and  $\tau$ .

In summary, the competitiveness of the proposed RDG decomposition scheme was demonstrated by the substantially improved performance of RDG-MAENS on the instances with large  $|Z|$ 's and  $\tau$ 's in terms of both solution quality and computational time. In addition, it is shown that both parameters  $g$  and  $\alpha$  affect the performance of RDG-MAENS. For example, on the EGL-G set, if  $g = 2$ , then  $\alpha = 10$  is the best value among all the compared  $\alpha$ 's. When  $g = 3$ ,  $\alpha = 5$  is the best option. This implies the necessity of including both  $g$  and  $\alpha$  to achieve better decomposition performance.

TABLE IX

BEST TOTAL COST OBTAINED BY THE RDG-MAENS ON THE INSTANCES WHERE THE BEST-KNOWN RESULTS WERE UPDATED. FOR EACH INSTANCE, THE NEW BEST-KNOWN RESULT IS MARKED IN BOLD

Name ( $ Z , \tau$ )	BK	RDG-MAENS					
		$g = 2$			$g = 3$		
		$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$
<b>C11</b> (94,10)	1810	1815	<b>1805</b>	1815	1815	1820	1815
<b>E09</b> (103,12)	2235	<b>2225</b>	2235	<b>2225</b>	<b>2225</b>	2235	<b>2225</b>
<b>E11</b> (94,10)	1835	1840	1835	<b>1830</b>	1845	1840	1840
<b>s2-B</b> (147,20)	13100	13127	13120	13112	<b>13099</b>	13119	13127
<b>s4-B</b> (190,27)	16321	16282	16295	<b>16260</b>	16289	16296	16287
<b>G1-A</b> (347,20)	1002264	1002530	1000068	<b>998777</b>	1002460	1000575	1002154
<b>G1-B</b> (347,25)	1126509	1118739	1118030	1118030	1118248	<b>1111971</b>	1118030
<b>G1-C</b> (347,30)	1260193	1246738	1243629	1243403	1245297	1243779	<b>1241762</b>
<b>G1-D</b> (347,35)	1397656	1376534	1372125	1373389	1378682	<b>1371443</b>	1376280
<b>G1-E</b> (347,40)	1541853	1522253	1514171	1517424	1518205	<b>1512584</b>	1513648
<b>G2-A</b> (375,22)	1111127	1102159	<b>1094912</b>	1097578	1100664	1096027	1100422
<b>G2-B</b> (375,27)	1223737	1214113	<b>1208326</b>	1209694	1214184	1213617	1210276
<b>G2-C</b> (375,32)	1366629	1350282	1346184	1342637	1347530	1344148	<b>1341519</b>
<b>G2-D</b> (375,37)	1506024	1486042	1482669	1483558	1484667	<b>1481181</b>	1481649
<b>G2-E</b> (375,42)	1650657	<b>1618899</b>	1621812	1620692	1619012	1618955	1620992

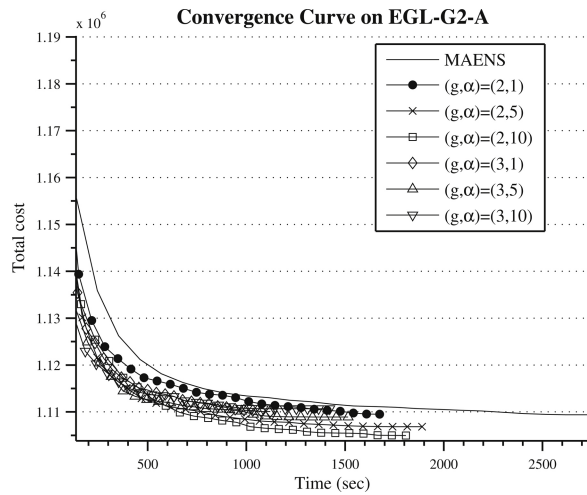


Fig. 8. Convergence curves of the compared algorithms on EGL-G2-A.

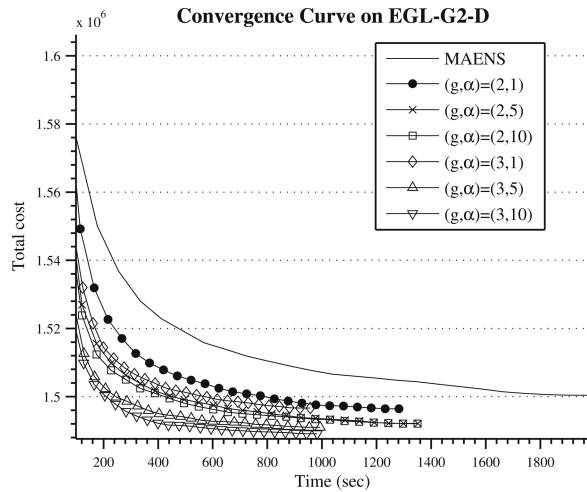


Fig. 9. Convergence curves of the compared algorithms on EGL-G2-D.

## VI. CONCLUSION

The practically important LSCARP is investigated in this paper, and a competitive decomposition-based approach is proposed for solving it. The proposed approach is a combination of the CC framework and an effective RDG decomposition scheme. The major contribution of this paper is the development of the RDG decomposition scheme. It takes the best-so-far solution as a guide, and defines a distance matrix between the routes to make the routes closer to each other more likely to be placed in the same subcomponent. In this way, more promising decompositions can be identified as the search continues. The final algorithm is named RDG-MAENS. The experimental studies demonstrated the efficacy of RDG-MAENS on the large and difficult CARP instances, such as all the EGL-G LSCARP instances and the egl instances with large  $|Z|$ 's and  $\tau$ 's. Furthermore, the best-known results of the EGL-G instances have been much improved by RDG-MAENS.

The performance of RDG-MAENS largely depends on  $g$  and  $\alpha$ . In this paper, the effects of  $g$  and  $\alpha$  were briefly investigated by comparing RDG-MAENS with different  $g$ 's and  $\alpha$ 's, and the experimental results showed that the best  $g$

and  $\alpha$  values are different for different instances. For example, Figs. 8 and 9 show that the best version of RDG-MAENS is (2, 10) on EGL-G2-A, and (3, 10) on EGL-G2-D. In addition, the tradeoff between the number of cycles and the generations in each cycle may also influence the performance of the algorithm. In the future, the relationship between the best parameter values and the problem characteristics, such as  $|Z|$ ,  $\tau$ , and the graph topology will be investigated, in an attempt to develop an adaptive parameter setting scheme to further enhance the performance of the algorithm.

## REFERENCES

- [1] M. Dror, *Arc Routing: Theory, Solutions and Applications*. Boston, MA, USA: Kluwer Academic, 2000.
- [2] H. Handa, L. Chapman, and X. Yao, "Robust salting route optimization using evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, Jul. 2006, pp. 10455–10462.
- [3] H. Handa, L. Chapman, and X. Yao, "Robust route optimization for gritting/salting trucks: A CERCIA experience," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 6–9, Feb. 2006.
- [4] G. Ghiani, G. Improta, and G. Laporte, "The capacitated arc routing problem with intermediate facilities," *Networks*, vol. 37, no. 3, pp. 134–143, 2001.
- [5] A. Amberg, W. Domschke, and S. Voß, "Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees," *Eur. J. Oper. Res.*, vol. 124, no. 2, pp. 360–376, 2000.
- [6] F. Chu, N. Labadi, and C. Prins, "A scatter search for the periodic capacitated arc routing problem," *Eur. J. Oper. Res.*, vol. 169, no. 2, pp. 586–605, 2006.
- [7] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Evolutionary algorithms for periodic arc routing problems," *Eur. J. Oper. Res.*, vol. 165, no. 2, pp. 535–553, 2005.
- [8] Y. Mei, K. Tang, and X. Yao, "A memetic algorithm for periodic capacitated arc routing problem," *IEEE Trans. Syst., Man, Cybern. B: Cybern.*, vol. 41, no. 6, pp. 1654–1667, Dec. 2011.
- [9] J. Campbell and A. Langevin, "Roadway snow and ice control," *Arc Routing: Theory, Solutions and Applications*. Boston, MA, USA: Kluwer, 2000, pp. 389–418.
- [10] M. Polacek, K. Doerner, R. Hartl, and V. Maniezzo, "A variable neighborhood search for the capacitated arc routing problem with intermediate facilities," *J. Heuristics*, vol. 14, no. 5, pp. 405–423, 2008.
- [11] J. Brandão and R. Eglese, "A deterministic tabu search algorithm for the capacitated arc routing problem," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1112–1126, 2008.
- [12] Y. Mei, K. Tang, and X. Yao, "A global repair operator for capacitated arc routing problem," *IEEE Trans. Syst., Man, Cybern. B: Cybern.*, vol. 39, no. 3, pp. 723–734, Jun. 2009.
- [13] X. Chen, "MAENS+: A divide-and-conquer based memetic algorithm for capacitated arc routing problem," in *Proc. 4th IEEE Int. Symp. Comput. Intell. Design*, vol. 1, Oct. 2011, pp. 83–88.
- [14] R. Martinelli, M. Poggi, and A. Subramanian, "Improved bounds for large scale capacitated arc routing problem," *Comput. Oper. Res.*, vol. 40, no. 8, pp. 2145–2160, 2013.
- [15] J. De Armon, "A comparison of heuristics for the capacitated Chinese postman problem," Master's thesis, University of Maryland, College Park, MD, USA, 1981.
- [16] E. Benavent, V. Campos, A. Corberán, and E. Mota, "The capacitated arc routing problem: Lower bounds," *Networks*, vol. 22, no. 7, pp. 669–690, 1992.
- [17] R. Eglese, "Routeing winter gritting vehicles," *Discrete Appl. Math.*, vol. 48, no. 3, pp. 231–244, 1994.
- [18] R. Eglese and L. Li, "A tabu search based heuristic for arc routing with a capacity constraint and time deadline," *Meta-Heuristics: Theory Applications*. Boston, MA, USA: Kluwer Academic, 1996, pp. 633–650.
- [19] L. Li and R. Eglese, "An interactive algorithm for vehicle routeing for winter-gritting," *J. Oper. Res. Soc.*, vol. 47, no. 2, pp. 217–228, 1996.
- [20] P. Beullens, L. Muylersmans, D. Catrysse, and D. Van Oudheusden, "A guided local search heuristic for the capacitated arc routing problem," *Eur. J. Oper. Res.*, vol. 147, no. 3, pp. 629–643, 2003.
- [21] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Competitive memetic algorithms for arc routing problems," *Ann. Oper. Res.*, vol. 131, no. 1, pp. 159–185, 2004.

- [22] Y. Mei, K. Tang, and X. Yao, "Improved memetic algorithm for capacitated arc routing problem," in *Proc. IEEE Congr. Evol. Comput.*, May 2009, pp. 1699–1706.
- [23] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1151–1166, Oct. 2009.
- [24] L. Feng, Y. Ong, Q. Nguyen, and A. Tan, "Towards probabilistic memetic algorithm: An initial study on capacitated arc routing problem," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 18–23.
- [25] L. Santos, J. Coutinho-Rodrigues, and J. R. Current, "An improved ant colony optimization based algorithm for the capacitated arc routing problem," *Transp. Res. B, Methodological*, vol. 44, no. 2, pp. 246–266, 2010.
- [26] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Trans. Evol. Comput.*, vol. 15, no. 2, pp. 151–165, Apr. 2011.
- [27] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.
- [28] M. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 1762–1769.
- [29] M. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 1–8.
- [30] X. Li and Y. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 1–15, Apr. 2011.
- [31] É. D. Taillard and S. Voss, "Popmusic: Partial optimization metaheuristic under special intensification conditions," in *Proc. Essays Surveys Metaheuristics*, 2002, pp. 613–629.
- [32] R. Bent and P. Van Hentenryck, "Randomized adaptive spatial decoupling for large-scale vehicle routing with time windows," in *Proc. Nat. Conf. Artif. Intell. Amer. Assoc. Artif. Intell.*, 2007, pp. 173–178.
- [33] D. Mester, O. Bräysy, and W. Dullaert, "A multi-parametric evolution strategies algorithm for vehicle routing problems," *Expert Syst. Applicat.*, vol. 32, no. 2, pp. 508–517, 2007.
- [34] M. Qi, W.-H. Lin, N. Li, and L. Miao, "A spatiotemporal partitioning approach for large-scale vehicle routing problems with time windows," *Transp. Res. E, Logistics Transp. Rev.*, vol. 48, no. 1, pp. 248–257, 2012.
- [35] R. Baldacci and V. Maniezzo, "Exact methods based on node routing formulations for arc routing problems," *Networks*, vol. 47, pp. 52–60, 2006.
- [36] H. Longo, M. de Aragão, and E. Uchoa, "Solving capacitated arc routing problems using a transformation to the CVRP," *Comput. Oper. Res.*, vol. 33, no. 6, pp. 1823–1837, 2006.
- [37] W.-L. Pearn, A. Assad, and B. L. Golden, "Transforming arc routing into node routing problems," *Comput. Oper. Res.*, vol. 14, no. 4, pp. 285–288, 1987.
- [38] E. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.
- [39] J. Belenguer and E. Benavent, "A cutting plane algorithm for the capacitated arc routing problem," *Comput. Oper. Res.*, vol. 30, no. 5, pp. 705–728, 2003.
- [40] E. Bartolini, J.-F. Cordeau, and G. Laporte, "Improved lower bounds and exact algorithm for the capacitated arc routing problem," *Math. Program.*, vol. 137, nos. 1–2, pp. 409–452, 2013.
- [41] M. Potter and K. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. PPSN*, 1994, pp. 249–257.
- [42] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, May 2001, pp. 1101–1108.
- [43] Y. Shi, H. Teng, and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Proc. 1st Int. Conf. Adv. Natural Comput.*, 2005, vol. II, pp. 1080–1088.
- [44] F. Van den Bergh and A. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.
- [45] R. Krishnapuram, A. Joshi, and L. Yi, "A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering," in *Proc. IEEE Int. Fuzzy Syst. Conf.*, vol. 3, Aug. 1999, pp. 1281–1286.
- [46] L. Kaufman and P. Rousseeuw, "Clustering by means of medoids," in *Statistical Data Analysis Based on the L1-Norm and Related Methods*, vol. 405, Y. Dodge, Ed. North-Holland, The Netherlands, Birkhäuser Basel, 1987.

- [47] J. L. Henning, "Spec CPU2000: Measuring CPU performance in the new millennium," *Computer*, vol. 33, no. 7, pp. 28–35, 2000.
- [48] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrical Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.



**Yi Mei** (S'09-M'13) received the bachelor's degree in mathematics from the University of Science and Technology of China (USTC), Hefei, China, in 2005, and the Ph.D. degree in computer science from the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, USTC, in 2010.

He is a Research Fellow with the School of Computer Science and Information Technology, RMIT University, Melbourne, Australia. His research interests include evolutionary algorithms, memetic algorithms, and other meta-heuristics with various real-world applications in the logistic area, such as arc routing problems, vehicle routing problems, traveling salesman problems, and supply chain. He is also interested in constrained optimization, multiobjective optimization, dynamic optimization, and robust design optimization.



**Xiaodong Li** (M'03-SM'07) received the B.Sc. degree in information science from Xidian University, Xi'an, China, in 1988, and the Dip.Com. and Ph.D. degrees in information science from the University of Otago, Dunedin, New Zealand, in 1992 and 1998, respectively.

He is an Associate Professor with the School of Computer Science and Information Technology, RMIT University, Melbourne, Australia. His research interests include evolutionary computation (in particular evolutionary multiobjective optimization, evolutionary optimization in dynamic environments, large scale optimization, and multimodal optimization), neural networks, complex systems, and swarm intelligence.

Dr. Li is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *International Journal of Swarm Intelligence Research (IJSIR)*. He is currently the Chair of the IEEE CIS Task Force on Large Scale Global Optimization and a Vice-Chair of the IEEE CIS Task Force on Swarm Intelligence. He is member of the editorial board of the *Journal of Swarm Intelligence* (Springer), and *Journal of Soft computing* (Springer), and a member of the Technical Committee on IEEE Soft Computing, Systems, Man and Cybernetics Society. He is an Advisor on the Scientific Advisory Board of SolveIT Software. He is a Vice-Chair of IEEE Victorian Section CIS Chapter, Melbourne, Australia. He is the recipient of the 2013 ACM SIGEVO Impact Award.



**Xin Yao** (F'03) is a Chair (Professor) of Computer Science and the Director of the Centre of Excellence for Research in Computational Intelligence and Applications, University of Birmingham, U.K. He is a Distinguished Lecturer of the IEEE Computational Intelligence Society (CIS). He has more than 400 refereed publications in international journals and conferences. His research interests include evolutionary computation and ensemble learning.

Dr. Yao won the 2001 IEEE Donald G. Fink Prize Paper Award, 2010 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist), 2011 IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award, and many other best paper awards at conferences. He won the prestigious Royal Society Wolfson Research Merit Award in 2012 and was selected to receive the 2013 IEEE CIS Evolutionary Computation Pioneer Award. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008.