Variable Neighborhood Decomposition for Large Scale Capacitated Arc Routing Problem

Yi Mei
School of Computer Science
and Information Technology
RMIT University
Melbourne, Victoria 3000, Australia
Email: yi.mei@rmit.edu.au

Xiaodong Li
School of Computer Science
and Information Technology
RMIT University
Melbourne, Victoria 3000, Australia
Email: xiaodong.li@rmit.edu.au

Xin Yao
CERCIA, School of Computer Science
University of Birmingham
B15 2TT Birmingham, UK
Email: x.yao@cs.bham.ac.uk

***Abstract*—In this paper, a Variable Neighborhood Decomposition (VND) is proposed for Large Scale Capacitated Arc Routing Problems (LSCARP). The VND employs the Route Distance Grouping (RDG) scheme, which is a competitive decomposition scheme for LSCARP, and generates different neighborhood structures with different tradeoffs between exploration and exploitation. The search first uses a neighborhood structure that is considered to be the most promising, and then broadens the neighborhood gradually as it is getting stuck in a local optimum. The experimental studies show that the VND performed better than the state-of-the-art RDG-MAENS counterpart, and the improvement is more significant when the subcomponent size is smaller. This implies a great potential of combining the VND with small subcomponents.**

I. Introduction

The Capacitated Arc Routing Problem (CARP) [1] is an important combinatorial optimization problem with many applications in the logistics area, such as winter gritting [2] [3] [4] [5], waste collection [6] [7] [8] and snow removal [9] [10]. It aims at designing a least cost plan for a given set of vehicles to serve a set of streets in the road network subject to some predefined constraints.

The Large Scale CARP (LSCARP) is practically significant, since the problem size (i.e., the number of streets) is very large in many real-world applications. For example, for the urban waste collection problem, there may be hundreds or even thousands of streets in a city for which waste is to be collected. Therefore, it is important to study how to solve LSCARP. The previous studies showed that applying the approaches for CARP directly to LSCARP cannot lead to promising performance [11] [12] [13] [14] [15] [16] due to the scalability issue. That is, the performance of the previous algorithms deteriorates rapidly either in quality [11] [12] or in speed [13] with the increasing problem size. In this case, the divide-and-conquer strategy becomes a promising alternative to address the scalability issue.

When decomposing LSCARP, an intuitive idea is to partition the set of tasks (the edges and arcs need to be served) into subsets, and then solve the sub-problems of serving each subset of tasks separately. Following this idea, Mei *et al.* proposed a Random Route Grouping (RRG) [15] and a Route Distance Grouping (RDG) [16] for partitioning the tasks. They both divide the routes of the best-so-far solution so that the upper bound of the optimal solution under the given decomposition is guaranteed to improve with the improvement of the best-so-far solution. RDG makes better use of domain knowledge (e.g., the distance between tasks) than RRG, and thus achieves much better results. Both decomposition schemes were combined with the Cooperative Co-evolution (CC), which is a generic framework in evolutionary computation that implements the divide-and-conquer strategy. Briefly speaking, CC divides the entire search process into a number of cycles. At the beginning of each cycle, the whole problem is decomposed by RRG or RDG, and then the resultant subcomponents are solved separately by the subcomponent optimizer (e.g., MAENS [17]).

When decomposing LSCARP with RRG or RDG, grouping the routes of the best-so-far solution $\bar{s}$ can also be seen as generating a neighborhood around $\bar{s}$. In other words, the neighborhood $\mathcal{N}(\bar{s})$ consists of all the solutions in which the tasks in different groups are in different routes. Therefore, the above CC can be considered as a Variable Neighborhood Search (VNS) process [18]. At cycle t of the CC, the neighborhood $\mathcal{N}^{(t)}(\bar{s})$ of the best-so-far solution $\bar{s}$ is defined by RRG or RDG, and then the local optimum within $\mathcal{N}^{(t)}(\bar{s})$ is to be found by the subcomponent optimizer MAENS.

Yao proposed simulated annealing algorithms with variable neighborhoods [19] [20] for combinatorial optimization problems (e.g., travelling salesman problems), and proved that the traditional fixed neighborhood structure, which is often the smallest and most greedy one, is not necessarily the best to obtain the global optimum. Dynamic neighborhoods can lead to better performance in combinatorial optimization [20]. The idea of using VNS for decomposing large scale optimization problems has also been proposed for continuous optimization and p-median problem [21], and the resultant approach is called the Variable Neighborhood Decomposition (VND). For continuous optimization, the neighborhood is naturally defined by fixing a subset of dimensions. For the p-median problem, the solution space is combinatorial, and thus a problem-specific neighborhood structure is defined to decompose the problem.

The advantage of VND is that it defines a set of differ-

ent neighborhood structures, which have different properties from each other, and examines them one after another. The variation of neighborhood structure makes the search explore more effectively within the solution space. For instance, for continuous optimization, the neighborhood $\mathcal{N}_k(\boldsymbol{x})$ can be defined by fixing all but k variables of the decision vector $\boldsymbol{x}$. This way, the search gradually increases the radius of the neighborhood around $\boldsymbol{x}$ as k increases. However, RRG and RDG do not have the advantage of VND since they only generate neighborhood structures with similar properties. RRG randomly groups the routes in each cycle, which is too blind. On the other hand, RDG groups the routes based on the fixed greediness parameter α, which leads to a fixed balance between exploration and exploitation throughout the search process.

In this paper, a set of variable neighborhood structures that lead to different extent of exploration are developed by varying the parameter α of RDG adaptively during the search process. The neighborhood that is expected to be most promising is examined first. Then, if the search is stuck in local optima, the broader neighborhoods are used to increase the exploration capability of the search to jump out of the local optima. The VND is combined with MAENS, and the resultant algorithm is tested on the EGL-G LSCARP benchmark instances.

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

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:

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

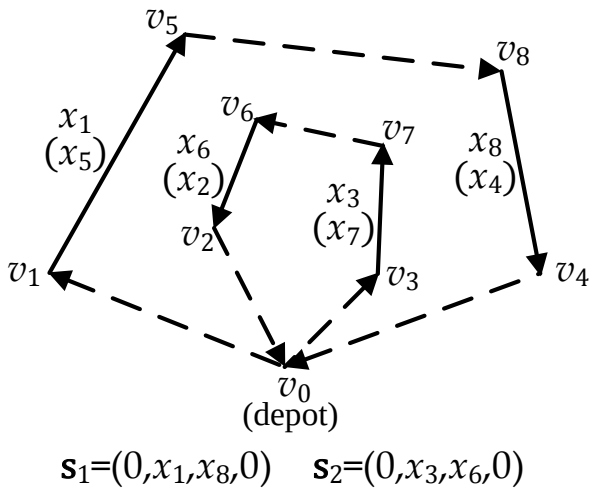


Fig. 1: An example of a CARP solution [15].

The mathematical formulation of CARP has been intensively investigated. Baldacci and Maniezzo [22] formulated the undirected CARP with n tasks as a VRP with $2n + 1$ customers. Belenguer and Benavent [23] proposed a mathematical formulation of CARP based on the cut constraints, and designed a cutting plane method to solve it. Bartolini *et al.* [24] 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 [14] [23] [24].

An example of a CARP solution is given in Fig. 1 [15]. In the figure, v_0 is the depot, and the tasks are represented by solid lines. There are four edge tasks, each of which are assigned two positive integer indices, representing both directions of it. For example, the task (v_2, v_6) is assigned the indices of x_2 and x_6. x_2 stands for the direction from v_2 to v_6, and x_6 the reverse direction. The dashed arrows between two nodes indicate the shortest path from the first node to the second. Then, the example solution $\boldsymbol{s}$ includes two routes $\boldsymbol{s}_1 = (0, x_1, x_8, 0)$ and $\boldsymbol{s}_2 = (0, x_3, x_6, 0)$, where 0 is the depot loop (v_0, v_0) defined to ensure that each route starts and ends at the depot.

III. Variable Neighborhood Decomposition for LSCARP

As shown in [21], the framework of VND can be briefly described as follows:

1) Define the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{\max}$. Generate an initial solution $\boldsymbol{x}$;
2) Set $k \leftarrow 1$;
3) Find the local optimum within the neighborhood $\mathcal{N}_k(\boldsymbol{x})$ of the current solution $\boldsymbol{x}$, i.e., $\boldsymbol{x}_k^* = \arg\min_{\boldsymbol{x} \in \mathcal{N}_k(\boldsymbol{x})}\{f(\boldsymbol{x})\}$;
4) If $f(\boldsymbol{x}_k^*)$ is better than $f(\boldsymbol{x})$ (e.g., $f(\boldsymbol{x}_k^*) < f(\boldsymbol{x})$ for a minimization problem), then move the current solution to the new local optimum ($\boldsymbol{x} \leftarrow \boldsymbol{x}_k^*$) and set $k \leftarrow$

1. Otherwise, examine the next neighborhood structure ($k \leftarrow k+1$);

5) If $k > k_{\max}$, then stop. Otherwise, go to Step 3).

When solving LSCARP with VND, the neighborhood structures are defined based on RDG. Given the best-so-far LSCARP solution $\bar{\boldsymbol{s}} = (\bar{\boldsymbol{s}}_1, \ldots, \bar{\boldsymbol{s}}_m)$, where the k^{th} route $\bar{\boldsymbol{s}}_k = (\bar{s}_{k1}, \ldots, \bar{s}_{kl_k})$ is represented by a sequence of tasks starting and ending at the depot ($\bar{s}_{k1} = \bar{s}_{kl_k} = 0$), RDG measures the distance between the routes and groups the routes by solving the corresponding fuzzy p-medoid problem [25] so that the routes that are closer to each other are more likely to be placed in the same group. Given the distance matrix between the routes $(\hat{\Delta}_{\text{route}})_{m\times m}$, the fuzzy p-medoid problem aims to obtain the optimal subset of routes $\boldsymbol{c} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_g)$ (g is the number of groups) out of the set of routes $\boldsymbol{s}$, which are called the *medoids*. It is stated as follows:

$$\min_{\boldsymbol{c}\subseteq\boldsymbol{s}} \mathcal{J}_\alpha(\boldsymbol{c};\boldsymbol{s}) = \sum_{\boldsymbol{s}_i\in\boldsymbol{s}\setminus\boldsymbol{c}} \sum_{\boldsymbol{c}_j\in\boldsymbol{c}} \mathcal{M}_\alpha(\boldsymbol{s}_i,\boldsymbol{c}_j)\cdot\hat{\Delta}_{\text{route}}(\boldsymbol{s}_i,\boldsymbol{c}_j) \quad (1)$$

where $\mathcal{M}_\alpha(\boldsymbol{s}_i,\boldsymbol{c}_j)$ is the membership function of $\boldsymbol{s}_i$ to $\boldsymbol{c}_j$, which is defined as follows:

$$\mathcal{M}_\alpha(\boldsymbol{s}_i,\boldsymbol{c}_j) = \frac{\left(\frac{1}{\hat{\Delta}_{\text{route}}(\boldsymbol{s}_i,\boldsymbol{c}_j)}\right)^\alpha}{\sum_{k=1}^{g}\left(\frac{1}{\hat{\Delta}_{\text{route}}(\boldsymbol{s}_i,\boldsymbol{c}_k)}\right)^\alpha} \quad (2)$$

After obtaining the set of medoids $\boldsymbol{c}$, each of the remaining routes is placed in the same group as one medoid using the roulette wheel method based on its membership function values to the medoids. That is, the route $\boldsymbol{s}_i \in \boldsymbol{s}\setminus\boldsymbol{c}$ has the probability of $\mathcal{M}_\alpha(\boldsymbol{s}_i,\boldsymbol{c}_j)$ to be in the group of $\boldsymbol{c}_j$. Finally, the tasks of the original LSCARP is decomposed into g subsets of tasks $(Z_1,\ldots,Z_g)$, each of which corresponds to all the tasks in the group of one medoid, i.e., $Z_j = \{z \in \boldsymbol{s}_i\cup\boldsymbol{c}_j | \boldsymbol{s}_i \textit{ is in the group of } \boldsymbol{c}_j\}$.

It can be seen that the greediness of the grouping depends on the parameter $\alpha \in [0,\infty)$. A larger α leads to a more greedy grouping. In the extreme case where $\alpha = \infty$, the greediness is maximized, and each $\boldsymbol{s}_i$ entirely belongs to the closest medoid to it. In the other extreme case where $\alpha = 0$, the greediness is minimized, and the grouping is totally blind, as the membership is the same for all the $\boldsymbol{s}_i$'s and $\boldsymbol{c}_j$'s regardless of the distance between them. In other words, different values of α lead to different groupings of the routes, i.e., neighborhood structures. Given the best-so-far solution $\bar{\boldsymbol{s}}$, one can select a set $\boldsymbol{\alpha} = \{\alpha_k | k = 1,\ldots,k_{\max}\}$ of different α values, and generate each neighborhood structure $\mathcal{N}_k(\bar{\boldsymbol{s}})$ as follows:

1) Obtain the grouping of the tasks $(Z_{k1},\ldots,Z_{kg})$ by applying RDG with the parameter α_k to the routes of $\bar{\boldsymbol{s}}$;
2) Let $\Omega(Z_{ki})$ be the solution space of the CARP with the task set of Z_{ki}, $i = 1,\ldots,g$, then $\mathcal{N}_k(\bar{\boldsymbol{s}}) = \{(\boldsymbol{s}_1,\ldots,\boldsymbol{s}_g) | \boldsymbol{s}_i \in \Omega(Z_{ki}), i = 1,\ldots,g\}$

Intuitively, it is more desirable to place close routes together in the same group, and thus a larger α tends to result in a more promising neighborhood and makes the search converge faster to a better local optimum. Hence, it is reasonable to start with a neighborhood generated by RDG with a larger α. Then, after the search gets stuck in the local optimum, it becomes more desirable to increase the exploring capability by switching to a broader neighborhood structure, that is, a neighborhood generated by RDG with a smaller α. For this reason, we set $\boldsymbol{\alpha}$ as a decreasing sequence of α's. From preliminary studies, it is observed that when $\alpha = 10$, the greediness is nearly maximized, and the membership function value of $\boldsymbol{s}_i$ is nearly 1 to the closest medoid to it. Therefore, we set $\alpha_1 = 10$. On the other hand, a condition of $\alpha_k \geq 1$ is imposed to guarantee the least bias to group the routes together with the medoids that are closer to them. Based on the above considerations, $\boldsymbol{\alpha}$ is defined as follows:

$$\alpha_k = 1 + 9\lambda^{k-1},\ 0 < \lambda < 1,\ k = 1,\ldots,k_{\max} \quad (3)$$

This way, it is obvious that $\alpha_1 = 10$ and $\alpha_k \geq 1$, $k = 1,\ldots,k_{\max}$. An illustration of the curves of α_k's with $\lambda = 0.2$, 0.4, 0.6 and 0.8 is given in Fig. 2.

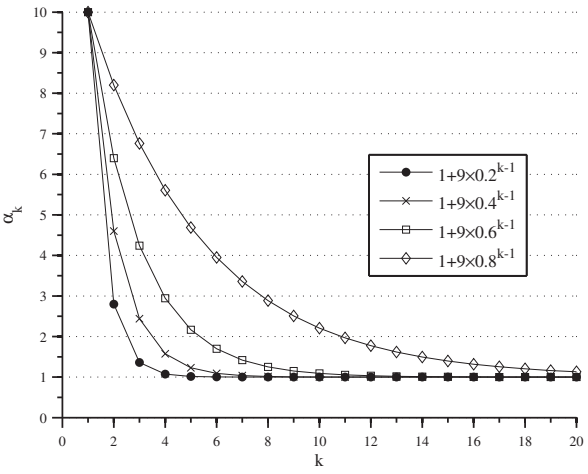


Fig. 2: The curves of α_k's with $\lambda = 0.2$, 0.4, 0.6 and 0.8, $k = 1,\ldots,20$.

Note that in Step 3) of VND, it is impossible to enumerate all the solutions within the neighborhood $\mathcal{N}_k(\bar{\boldsymbol{s}})$ of the best-so-far solution $\bar{\boldsymbol{s}}$, since the CARPs are NP-hard. Therefore, MAENS [17] is employed to search within $\mathcal{N}_k(\bar{\boldsymbol{s}})$ for a number of generations to approximate the local optimum. It is found that the CC framework is a natural implementation to this end, and the number of generations is naturally defined in a cycle. In summary, the VND for LSCARP can be described as follows:

1) *Initialization*: Define the set of neighborhood structures $\mathcal{N}_k$ based on RDG and $\boldsymbol{\alpha} = \{\alpha_k | k = 1,\ldots,k_{\max}\}$. Randomly generate the best-so-far solution $\bar{\boldsymbol{s}}$. Set $k \leftarrow 1$, $t = 1$;
2) *Repeat* the following steps until $t > t_{\max}$:
   a) Search within the neighborhood $\mathcal{N}_k(\bar{\boldsymbol{s}})$ by MAENS to obtain a new best-so-far solution $\bar{\boldsymbol{s}}'_k$. To be specific, apply MAENS to each CARP with the task set of Z_{ki},

$i = 1, \ldots, g$ for N_t generations to obtain the new best-so-far solution $\bar{\boldsymbol{s}}'_{ki}$ with respect to Z_{ki}. Then, the new best-so-far solution is $\bar{\boldsymbol{s}}'_k = (\bar{\boldsymbol{s}}'_{k1}, \ldots, \bar{\boldsymbol{s}}'_{kg})$;

b) If $tc(\bar{\boldsymbol{s}}'_k) < tc(\bar{\boldsymbol{s}})$, then $\bar{\boldsymbol{s}} \leftarrow \bar{\boldsymbol{s}}'_k$ and set $k \leftarrow 1$. Otherwise, $k \leftarrow k + 1$;

c) $t \leftarrow t + 1$;

3) *Return* the final best-so-far solution $\bar{\boldsymbol{s}}$.

The above VND framework is similar to the framework of RDG-MAENS [16], which divides the entire search process into $t_{\max}$ cycles. At the beginning of each cycle, the original LSCARP is decomposed into a number of smaller sized CARPs by RDG, which leads to the neighborhood $\mathcal{N}_k(\bar{\boldsymbol{s}})$. The key difference is that in the VND framework, the neighborhood structure changes as the number of cycles without improvement increases.

The relationship between the neighborhood structure and α value is not simply a one-to-one mapping. In contrast with the continuous value of α, the neighborhood $\mathcal{N}_k(\bar{\boldsymbol{s}})$ is a discrete grouping of the tasks obtained by solving the fuzzy p-medoid problem Eq. (1) and grouping the routes randomly according to their membership function value Eq. (2) to each medoid. It is known that when α is large, the grouping is greedy and all the routes are highly likely to be grouped together with the closest medoid to it. Therefore, it is possible that some of the $\mathcal{N}_k(\bar{\boldsymbol{s}})$'s are the same as each other, especially when k is small. On the other hand, the grouping becomes much more arbitrary when α is small, and even the same α can lead to different neighborhood structures. This property is helpful in cooperating with the search of MAENS within each cycle. Since CARP is NP-hard, the neighborhood $\mathcal{N}_k(\bar{\boldsymbol{s}})$ may not be fully exploited by MAENS in a single cycle, especially when the decomposed sub-problems still have large problem sizes. In this case, it might take multiple cycles to fully exploit the current neighborhood before changing to a broader one. As discussed before, the neighborhood structures are highly likely to be the same as each other when k is small (α is large). In other words, the more promising neighborhood structures are more likely to be exploited for multiple cycles and thus exploited more comprehensively. The parameter λ controls the changing speed of the neighborhood. If λ is larger, then it tends to take more cycles to switch to broader neighborhoods.

IV. Experimental Studies

The neighborhood structures generated by the VND for LSCARP are generated by RDG, which depends on the number of subcomponents g and parameter α. Eq. (3) shows that α_k depends on the parameter λ. Then, the VND for LSCARP has two parameters of g and λ. In the experimental studies, g is set to a constant throughout the entire search space, and thus the neighborhood structure solely depends on λ. The VND for LSCARP is tested on the EGL-G [11] LSCARP benchmark instances with different λ values and compared with the RDG-MAENS that uses a static neighborhood structure with $g = 2$, $\alpha = 10$ and $g = 3$, $\alpha = 10$, denoted as $(2, 10)$ and $(3, 10)$, respectively. From Eq. (3), it is obvious that the RDG-MAENS $(2, 10)$ and $(3, 10)$ are equivalent to the VNDs with $g = 2$, $\lambda = 1$ and $g = 3$, $\lambda = 1$, since $\alpha_k = 1 + 9 \times 1^k = 10$, $\forall k$ when $\lambda = 1$.

TABLE I: The parameter settings of the VND for LSCARP

Parameter	Description	Value
g	Number of subcomponents	2, 3
λ	Parameter of α_k series	1, 0.8, 0.6, 0.4
$psize$	Population size	30
$offsize$	Offspring population size	$6 \cdot psize$
P_{ls}	Probability of local search	0.2
N_t	Generations per cycle	10
$t_{\max}$	Number of cycles	50

A. Parameter Settings

The parameter settings are given in Table I, where g and λ are the parameters for generating the neighborhood structures $\mathcal{N}_k$, and the remaining are the parameters of MAENS and the VND. All the parameters are set the same as that of RDG-MAENS except the new parameter λ, as they essentially have the same framework.

The tested EGL-G benchmark set [11] was generated based on the road network of Lancashire, UK, consisting of 10 LSCARP instances with 347 to 375 tasks. Different instances were generated by selecting different task sets and capacities of vehicles. 30 independent runs were conducted for each of the compared algorithms on all the test instances. All the algorithms were implemented in C++, 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

Tables II and III show the mean and standard deviation of the 30 total costs obtained by the 30 independent runs of the compared VNDs with $g = 2$ and $g = 3$ on the EGL-G LSCARP instances, respectively. The features of the instances are given as well. "$|V|$", "$|E|$" and "$|Z|$" refer to the number of vertices, edges and tasks, respectively. "τ" is the minimal number of vehicles required to serve all the tasks, which is obtained as follows:

$$\tau = \left\lceil \frac{\sum_{z \in Z} d(z)}{Q} \right\rceil \tag{4}$$

In general, with the same number of tasks, a larger τ indicates more routes, and thus a tighter capacity constraint and higher level of difficulty of the problem.

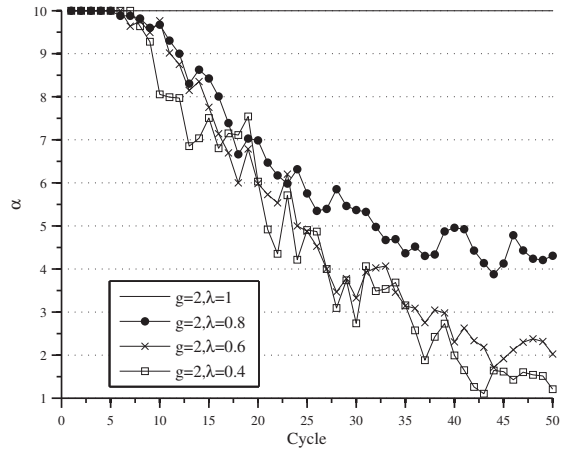
From Table II, one can see that when $g = 2$, all the compared VNDs show similar performance to each other. The VND with $\lambda = 0.4$ obtains the best mean of the total costs on 5 out of the 10 instances (G1-D, G1-E, G2-B, G2-C and G2-D), while the one with $\lambda = 1$ (i.e., the RDG-MAENS $(2, 10)$) performs the best in terms of the mean total cost on 3 instances (G1-B, G1-C and G2-E). However, when conducting Wilcoxon's rank sum test [26] with significance level of 0.05 to each pair of the results of the compared algorithms, it can be seen that none of the differences is statistically significant. There are two major reasons of this phenomenon. First, when

TABLE II: The mean and standard deviation (in the parenthesis) of the results obtained by the compared VNDs with $g = 2$ on the EGL-G test set. The best mean is marked with †.

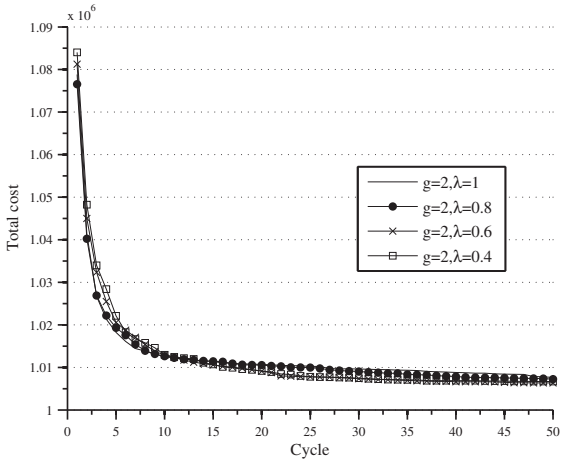
Name	$(\|V\|,\|E\|,\|Z\|,\tau)$	$g = 2$			
		$\lambda = 1$	$\lambda = 0.8$	$\lambda = 0.6$	$\lambda = 0.4$
G1-A	(255,375,347,20)	1007619 (4449)	1007232 (3949)	1006352† (4189)	1006694 (3637)
G1-B	(255,375,347,25)	1122863† (4587)	1123077 (4362)	1123231 (5477)	1125488 (5950)
G1-C	(255,375,347,30)	1250174† (5918)	1251803 (4671)	1252883 (4757)	1252728 (4144)
G1-D	(255,375,347,35)	1386120 (6590)	1384753 (5938)	1384673 (4955)	1384463† (5659)
G1-E	(255,375,347,40)	1525629 (5716)	1525482 (6472)	1528987 (6497)	1525420† (5302)
G2-A	(255,375,375,22)	1104944 (4781)	1106732 (5517)	1104909† (4922)	1106198 (5835)
G2-B	(255,375,375,27)	1221429 (6812)	1222592 (6051)	1222761 (5599)	1221117† (4753)
G2-C	(255,375,375,32)	1355548 (7329)	1353877 (3997)	1354401 (4685)	1353811† (4845)
G2-D	(255,375,375,37)	1492063 (5652)	1491564 (5051)	1493078 (6007)	1490923† (6629)
G2-E	(255,375,375,42)	1629002† (5056)	1632116 (6556)	1632677 (7016)	1631074 (4824)
Avg.		1309539	1309923	1310395	1309792

$g = 2$, the resultant CARP sub-problems are still large. Given around 350 tasks in the original problem, the average number of tasks in each subcomponent is $350/2 = 175$, which is still quite large, not to mention that the size of subcomponents are usually non-uniformly distributed. Thus, it is still hard to fully exploit the current neighborhood before moving to broader neighborhoods. Second, the actual performance of the VNDs largely depend on the initial neighborhood ($\mathcal{N}_1(\bar{\boldsymbol{s}})$ with $\alpha_1 = 10$). Fig. 3 shows the curves of the average α and total cost over the 30 independent runs of the compared VNDs with $g = 2$ on G1-A. The pattern is similar on all the other instances. From Fig. 3a, it can be seen that for all the compared VNDs with $g = 2$, the average α is almost equal to 10 in the first 10 cycles. On the other hand, Fig. 3b shows that the average total cost already converges to a good local optimum within the first 10 cycles. Starting from such a good local optimum, there is little space for improving the solution even by searching within a broader neighborhood. Meanwhile, as mentioned in the first reason, the original neighborhood with $\alpha_1 = 10$ is not fully exploited due to the large problem size, and thus continuing searching within the same neighborhood may still be able to locate more promising areas with more comprehensive exploitation. As a result, after being stuck in the local optimum with respect to the original neighborhood structure $\mathcal{N}_1(\bar{\boldsymbol{s}})$ (i.e., after the first 10 cycles), changing to broader neighborhood structures makes no significant difference to the performance of the algorithm.

From Table III, it can be seen that when $g = 3$, all the



(a) Curves of average α



(b) Curves of average total cost

Fig. 3: Curves of the average α and total cost of the VNDs with $g = 2$ on EGL-G1-A.

VNDs with $\lambda < 1$ performed much better than the VND with $\lambda = 1$, i.e., the RDG-MAENS $(3, 10)$. The VND with $\lambda = 0.6$ performed the best, obtaining significantly better results than the VND with $\lambda = 1$ (the RDG-MAENS $(3, 10)$) on 5 instances (G1-A, G1-B, G1-E, G2-A and G2-C) under Wilcoxon's rank sum test with significance level of 0.05. This is because when $g = 3$, the decomposed sub-problems are much smaller than that obtained by $g = 2$. Then, each neighborhood structure is exploited more comprehensively given the same number of cycles. In this situation, switching to broader neighborhood structures becomes more desirable than staying in the same narrow neighborhood. Fig. 4 shows the curves of the average α and total cost over the 30 independent runs of the compared VNDs with $g = 3$ on G1-A, which is selected as a representative of all the other instances. From the figures, one can see that after being stuck in the local optima of the original neighborhood with $\alpha_1 = 10$ (around 7 cycles), changing to broader neighborhood structures ($\lambda < 1$) can lead to clearly larger improvement than staying in the same neighborhood ($\lambda = 1$). Besides, the outperformance of the VND with $\lambda = 0.6$ over the ones with $\lambda = 0.8$ and 0.4

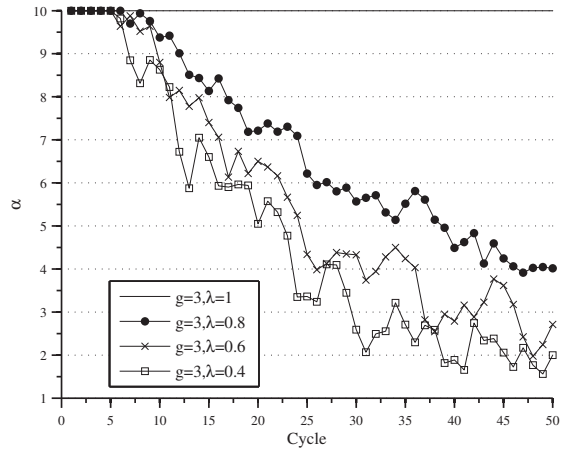
TABLE III: The mean and standard deviation (in the parenthesis) of the results obtained by the compared VNDs with $g = 3$ on the EGL-G test set. The best mean is marked with †. For each VND with $\lambda < 1$, if its results are significantly better than that of the VND with $\lambda = 1$, then the corresponding mean value is marked in bold.

Name	$(\|V\|,\|E\|,\|Z\|,\tau)$	$g = 3$			
		$\lambda = 1$	$\lambda = 0.8$	$\lambda = 0.6$	$\lambda = 0.4$
G1-A	(255,375,347,20)	1014068 (8986)	**1006813** (4637)	**1007393** (4289)	**1005873**† (4153)
G1-B	(255,375,347,25)	1127199 (7083)	1125378 (6318)	**1122077**† (4496)	1124759 (4895)
G1-C	(255,375,347,30)	1255139 (7580)	1252304 (5547)	1253789 (5706)	1251693† (4457)
G1-D	(255,375,347,35)	1387294 (7452)	1386849 (7801)	1383997† (7321)	1385181 (5854)
G1-E	(255,375,347,40)	1530237 (7280)	**1525712** (5440)	**1525994** (6491)	**1525671**† (5557)
G2-A	(255,375,375,22)	1110665 (6875)	1107977 (5274)	**1105870**† (5306)	**1106760** (3972)
G2-B	(255,375,375,27)	1224629 (10357)	1223538 (8920)	1220012† (5152)	1221305 (6440)
G2-C	(255,375,375,32)	1355141 (6345)	1353782 (5609)	**1351845**† (4073)	1352830 (4921)
G2-D	(255,375,375,37)	1489114† (6000)	1489398 (4816)	1489500 (5597)	1491292 (4988)
G2-E	(255,375,375,42)	1629837† (4557)	1630761 (6746)	1630048 (7242)	1630440 (7281)
Avg.		1312332	1310251	1309052	1309580

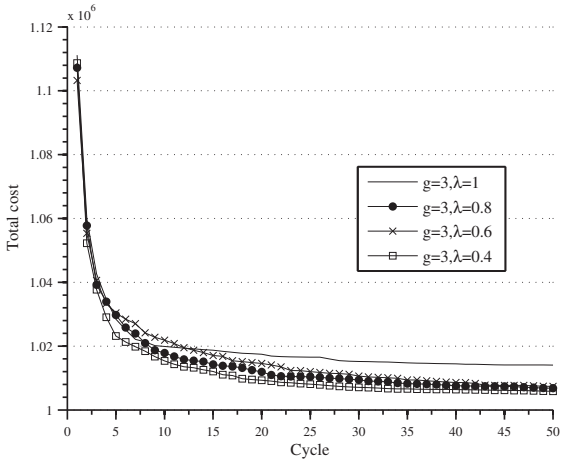
implies that the λ value should be set neither too large nor too small to achieve the best tradeoff between the exploitation within the current neighborhood and the exploration in the broader neighborhoods.

When $g = 2$, the compared RDG-MAENS $(2, 10)$ is the state-of-the-art version. However, when $g = 3$, the RDG-MAENS $(3, 10)$ is not the state-of-the-art [16]. Thus, it is necessary to compare the VND with the state-of-the-art RDG-MAENS version for $g = 3$, i.e., $(3, 5)$ [16]. Table IV shows the mean and standard deviation of the results obtained by the state-of-the-art versions of RDG-MAENS and VND with $g = 2$ and 3 on the EGL-G test set. The best mean is marked with †, and the significantly better mean is marked in bold. It is seen that the VND with $g = 3$ and $\lambda = 0.6$ obtained best mean total cost on 4 instances, better than the two state-of-the-art RDG-MAENS versions ($(2, 10)$ and $(3, 5)$, obtained the best mean on 3 and 2 instances, respectively). In addition, the VND with $g = 3$ and $\lambda = 0.6$ obtained significantly better total cost on G2-C. In short, the VND with $g = 3$ and $\lambda = 0.6$ showed better performance than the state-of-the-art versions of RDG-MAENS.

Table V shows the best results of all the compared VNDs on the EGL-G test set, along with the best known results, which were obtained from [16]. The new best known results are marked in bold. The best known results are updated on



(a) Curves of average α



(b) Curves of average total cost

Fig. 4: Curves of the average α and total cost of the VNDs with $g = 3$ on EGL-G1-A.

6 out of the total 10 instances, 5 of which were obtained by the VNDs with $g = 3$. This shows a potential of using large g value (small subcomponent size) and variable neighborhood structure to obtain better solutions.

Table VI shows the average computational time over the 30 independent runs of the compared VNDs on the EGL-G test set. It can be seen that with the same g value, the VNDs with different λ values have similar computational time to each other. This implies that the computational time of the algorithm largely depends on the g value, which is the most important factor in determining the subcomponent size.

In summary, the experimental results show a great potential of combining variable neighborhood structures and a small subcomponent size (a large g value) to search more effectively in the huge solution space of LSCARP.

V. Conclusion

A Variable Neighborhood Decomposition (VND) method is proposed for tackling the LSCARP more effectively. A set of neighborhood structures with different properties are defined based on the RDG decomposition scheme [16]. Briefly

TABLE V: The best results of the compared VNDs on the EGL-G test set. The new best known results are marked in bold.

Name	($\|Z\|$,τ)	BK	$g=2$				$g=3$			
			$\lambda=1$	$\lambda=0.8$	$\lambda=0.6$	$\lambda=0.4$	$\lambda=1$	$\lambda=0.8$	$\lambda=0.6$	$\lambda=0.4$
G1-A	(347,20)	998777	998777	998611	998020	1001289	1002154	1001013	**997055**	997313
G1-B	(347,25)	1111971	1118030	1117058	1115659	1118030	1118030	1117708	1114120	1116929
G1-C	(347,30)	1241762	1243403	1242704	1244475	1245092	1241762	**1241586**	1243808	1244115
G1-D	(347,35)	1371443	1373389	1375029	1374914	1373510	1376280	1375002	1373480	1375149
G1-E	(347,40)	1512584	1517424	1514639	1518090	1517507	1513648	1514687	1517772	1515838
G2-A	(375,22)	1094912	1097578	1096797	**1094827**	1095933	1100422	1099298	1098454	1100024
G2-B	(375,27)	1208326	1209694	1214490	1209562	1212192	1210276	**1203579**	1211759	1210266
G2-C	(375,32)	1341519	1342637	1347103	1345840	1345156	1341519	1345228	1344184	1344574
G2-D	(375,37)	1481181	1483558	1483541	1481666	1481746	1481649	1481218	1481045	**1479814**
G2-E	(375,42)	1618899	1620692	1618832	1617295	1620381	1620992	1618338	**1616119**	1616857

TABLE IV: The mean and standard deviation (in the parenthesis) of the results obtained by the state-of-the-art versions of RDG-MAENS and VND with $g=2$ and 3 on the EGL-G test set. The best mean is marked with †, and the significantly better mean is marked in bold.

Name	$g=2$		$g=3$	
	RDG-MAENS ($\alpha=10$)	VND ($\lambda=0.4$)	RDG-MAENS ($\alpha=5$)	VND ($\lambda=0.6$)
G1-A	1007619 (4449)	1006694† (3637)	1007223 (4734)	1007393 (4289)
G1-B	1122863 (4587)	1125488 (5950)	1124751 (6190)	1122077† (4496)
G1-C	1250174† (5918)	1252728 (4144)	1251718 (5954)	1253789 (5706)
G1-D	1386120 (6590)	1384463 (5659)	1383619† (6419)	1383997 (7321)
G1-E	1525629 (5716)	1525420 (5302)	1524393† (5627)	1525994 (6491)
G2-A	1104944† (4781)	1106198 (5835)	1108916 (6828)	1105870 (5306)
G2-B	1221429 (6812)	1221117 (4753)	1222183 (6013)	1220012† (5152)
G2-C	1355548 (7329)	1353811 (4845)	1353118 (4881)	**1351845**† (4073)
G2-D	1492063 (5652)	1490923 (6629)	1489723 (5284)	1489500† (5597)
G2-E	1629002† (5056)	1631074 (4824)	1630132 (6926)	1630048 (7242)
Avg.	1309539	1309792	1309578	1309052

speaking, different neighborhoods are generated by the RDG with different α values, which determine different degrees of greediness when grouping the routes together. The resultant VND with different switching rates between neighborhood structures (controlled by the parameter $\lambda < 1$) was compared to the state-of-the-art RDG-MAENS counterpart ($\lambda = 1$) on the EGL-G LSCARP test set. The experimental results show that the VND performs better than the RDG-MAENS, and the improvement increases with the increase of g (the decrease of subcomponent size). The VND with $g = 3$ and $\lambda = 0.6$ performed better than the best version of RDG-MAENS so far. The VND has a great potential to be combined with small subcomponent size.

TABLE VI: The average computational time over the 30 independent runs of the compared VNDs on the EGL-G test set.

Name	$g=2$				$g=3$			
	$\lambda=1$	$\lambda=0.8$	$\lambda=0.6$	$\lambda=0.4$	$\lambda=1$	$\lambda=0.8$	$\lambda=0.6$	$\lambda=0.4$
G1-A	1628	1685	1633	1667	985	1065	1083	1022
G1-B	1390	1444	1436	1438	888	959	942	922
G1-C	1264	1304	1290	1292	846	914	883	882
G1-D	1145	1183	1171	1191	822	867	839	857
G1-E	1068	1119	1109	1119	775	858	837	831
G2-A	1675	1748	1709	1730	1073	1110	1114	1139
G2-B	1511	1556	1558	1550	1002	1059	1020	1028
G2-C	1373	1431	1405	1441	947	1022	992	990
G2-D	1268	1337	1331	1374	920	1015	975	948
G2-E	1181	1249	1245	1278	893	968	954	930
Avg.	1350	1406	1389	1408	915	984	964	955

The basic idea of VND is to start with the smallest but most promising neighborhood structure, and gradually extend to broader neighborhood structures when the search is getting stuck in a local optimum of the current neighborhood. The RDG-based definition of the neighborhood structures is based on the assumption that the neighborhood generated by RDG with the largest α value is the most promising one, which has not been verified. Besides, in contrast to the traditional VND method, one cannot guarantee full enumeration of the current neighborhood structure before moving to the next one. It is obvious that the neighborhoods generated by larger subcomponents need more cycles to be fully exploited. In future, more investigations on the neighborhood structures are to be conducted to ensure that the more promising neighborhoods are examined first. Also, an adaptive exploitation scheme of each neighborhood is to be developed by setting the number of cycles or generations for each neighborhood based on its subcomponent sizes.

ACKNOWLEDGMENT

This work was supported by an ARC Discovery grant (No. DP120102205), an NSFC grant (No. 61329302), and an

EPSRC grant (No. EP/I010297/1). Xin Yao was supported by a Royal Society Wolfson Research Merit Award.

REFERENCES

[1] M. Dror, *Arc routing: theory, solutions and applications*. Boston: Kluwer Academic Publishers, 2000.

[2] H. Handa, L. Chapman, and X. Yao, "Robust Salting Route Optimization Using Evolutionary Algorithms," in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, vol. 1. Springer, 2006, pp. 10 455–10 462.

[3] ——, "Robust route optimization for gritting/salting trucks: a CERCIA experience," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 6–9, 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," *European Journal of Operational Research*, 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," *European Journal of Operational Research*, vol. 169, no. 2, pp. 586–605, 2006.

[7] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Evolutionary algorithms for periodic arc routing problems," *European Journal of Operational Research*, 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 Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 41, no. 6, pp. 1654–1667, 2011.

[9] J. Campbell and A. Langevin, "Roadway snow and ice control," *Arc routing: theory, solutions and applications. Boston, MA: Kluwer*, pp. 389–418, 2000.

[10] M. Polacek, K. Doerner, R. Hartl, and V. Maniezzo, "A variable neighborhood search for the capacitated arc routing problem with intermediate facilities," *Journal of 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," *Computers and Operations Research*, 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 Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 39, no. 3, pp. 723–734, 2009.

[13] X. Chen, "MAENS+: A Divide-and-Conquer Based Memetic Algorithm for Capacitated Arc Routing Problem," in *2011 Fourth International Symposium on Computational Intelligence and Design (ISCID)*, vol. 1. IEEE, 2011, pp. 83–88.

[14] R. Martinelli, M. Poggi, and A. Subramanian, "Improved Bounds for Large Scale Capacitated Arc Routing Problem," *Computers & Operations Research*, vol. 40, no. 8, pp. 2145–2160, 2013.

[15] Y. Mei, X. Li, and X. Yao, "Decomposing large-scale capacitated arc routing problems using a random route grouping method," in *2013 IEEE Congress on, Evolutionary Computation (CEC)*. IEEE, 2013.

[16] ——, "Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation, Preprinted, DOI: 10.1109/TEVC.2013.2281503*, 2013.

[17] K. Tang, Y. Mei, and X. Yao, "Memetic Algorithm with Extended Neighborhood Search for Capacitated Arc Routing Problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, 2009.

[18] P. Hansen, N. Mladenović, and J. A. M. Pérez, "Variable neighbourhood search: methods and applications," *Annals of Operations Research*, vol. 175, no. 1, pp. 367–407, 2010.

[19] X. Yao, "Simulated annealing with extended neighbourhood," *International Journal of Computer Mathematics*, vol. 40, no. 3, pp. 169–189, 1991.

[20] ——, "Dynamic neighbourhood size in simulated annealing," in *Proc. of Int'l Joint Conf. on Neural Networks (IJCNN'92)*, vol. 1, 1992, pp. 411–416.

[21] P. Hansen, N. Mladenović, and D. Perez-Britos, "Variable neighborhood decomposition search," *Journal of Heuristics*, vol. 7, no. 4, pp. 335–350, 2001.

[22] R. Baldacci and V. Maniezzo, "Exact methods based on node routing formulations for arc routing problems," *Networks*, vol. 47, pp. 52–60, 2006.

[23] J. Belenguer and E. Benavent, "A cutting plane algorithm for the capacitated arc routing problem," *Computers and Operations Research*, vol. 30, no. 5, pp. 705–728, 2003.

[24] E. Bartolini, J.-F. Cordeau, and G. Laporte, "Improved lower bounds and exact algorithm for the capacitated arc routing problem," *Mathematical Programming*, vol. 137, no. 1-2, pp. 409–452, 2013.

[25] R. Krishnapuram, A. Joshi, and L. Yi, "A fuzzy relative of the k-medoids algorithm with application to web document and snippet clustering," in *1999 IEEE International Fuzzy Systems Conference Proceedings*, vol. 3. IEEE, 1999, pp. 1281–1286.

[26] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.