Memetic Algorithm with Extended Neighborhood Search for Capacitated Arc Routing Problems

Ke Tang, *Member, IEEE,* Yi Mei, *Student Member, IEEE,* and Xin Yao, *Fellow, IEEE*

***Abstract*— The capacitated arc routing problem (CARP) has attracted much attention during the last few years due to its wide applications in real life. Since CARP is NP-hard and exact methods are only applicable to small instances, heuristic and metaheuristic methods are widely adopted when solving CARP. In this paper, we propose a memetic algorithm, namely memetic algorithm with extended neighborhood search (MAENS), for CARP. MAENS is distinct from existing approaches in the utilization of a novel local search operator, namely Merge-Split (MS). The MS operator is capable of searching using large step sizes, and thus has the potential to search the solution space more efficiently and is less likely to be trapped in local optima. Experimental results show that MAENS is superior to a number of state-of-the-art algorithms, and the advanced performance of MAENS is mainly due to the MS operator. The application of the MS operator is not limited to MAENS. It can be easily generalized to other approaches.**

***Index Terms*— Capacitated arc routing problem (CARP), evolutionary optimization, local search, memetic algorithm, metaheuristic search.**

I. Introduction

The ARC routing problem is a classic problem with many applications in the real world, such as urban waste collection, post delivery, sanding or salting the streets [1], [2], etc. It is a combinatorial optimization problem that requires determining the least cost routing plan for vehicles subject to some constraints [3]. The capacitated arc routing problem (CARP), which is the most typical form of the arc routing problem, is considered in this paper. It can be described as follows: a mixed graph $G = (V, E, A)$, with a set of vertices denoted by V, a set of edges denoted by E and a set of arcs (i.e., directed edges) denoted by A, is given. There is a central depot vertex $dep \in V$, where a set of vehicles are based. A subset $E_R \subseteq E$ composed of all the edges required to be served and a subset $A_R \subseteq A$ composed of all the arcs required to be served are also given. The elements of these two subsets are called edge tasks and arc tasks, respectively. Each edge or arc in the graph is associated with a demand, a serving cost, and a deadheading cost (the cost of a vehicle traveling along the edge/arc without serving it). Both the demand and the serving cost are zero for the edges and arcs that do not require service. A solution to the problem is a routing plan that consists of a number of routes for the vehicles, and the objective is to minimize the total cost of the routing plan subject to the following constraints:

1) each route starts and ends at the depot;
2) each task is served in exactly one route;
3) the total demand of each route must not exceed the vehicle's capacity Q.

Since CARP is NP-hard [4], exact methods are only applicable to small-size instances. As a result, heuristics and metaheuristics are often considered in the literature. For example, Augment–Merge [4], Path-Scanning [5], the "route first-cluster second" type heuristic proposed in [6], and Ulusoy's Heuristic [7] are typical heuristic methods. In 2000, Hertz *et al.* proposed a tabu search for CARP (CARPET) [8]. In CARPET, a solution is represented as a set of routes, each of which is an ordered list of vertices. Every vertex is associated with a 0-1 variable indicating whether the edge between this vertex and the successive vertex is served. Based on CARPET, a variable neighborhood descent (VND) algorithm was later proposed by replacing the tabu search process in CARPET with a variable neighborhood search process [9]. In 2003, Beullens *et al.* developed a guided local search (GLS) algorithm for CARP [10]. GLS adopts an edge marking scheme, which marks (unmarks) edges based on the information of a previous search procedure. Local search operators are only applied to those marked edges so as to make the search process more effective. After that, Lacomme *et al.* proposed a memetic algorithm (LMA[1]), which combines the genetic algorithm (GA) with local search [11]. LMA employs a genotype encoding scheme. That is, a solution is represented as a sequence of tasks, and the intermediate vertices between two subsequent tasks are omitted. To obtain the complete routing plan from a solution, one needs to connect every two subsequent tasks with the shortest path between them, and then apply Ulusoy's heuristic to separate the sequence into a number of routes.

Manuscript received November 18, 2008; revised February 14, 2009; accepted May 5, 2009. First version published August 11, 2009; current version published September 30, 2009. This work was supported in part by the Engineering and Physical Sciences Research Council under Grant EP/E058884/1 on "Evolutionary Algorithms for Dynamic Optimization Problems: Design, Analysis and Applications," by the National Natural Science Foundation of China under Grants 60533020 and U0835002, and by the Fund for Foreign Scholars in University Research and Teaching Programs, Grant B07033.

K. Tang and Y. Mei are with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: ketang@ustc.edu.cn; meiyi@mail.ustc.edu.cn).

X. Yao is with the Center 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.2009.2023449

[1]In the literature, MA is usually referred to as a general framework rather than any specific algorithm. For the sake of clarity, we denote the MA proposed by Lacomme *et al.* [10] as LMA in this paper.

In 2006, Handa *et al.* proposed an evolutionary algorithm (EA) [1], [2] for a salting route optimization problem in the U.K. This EA adopts a similar encoding scheme as LMA, i.e., a solution is represented by a sequence of tasks. The difference is that a solution obtained by the EA can be naturally separated into different routes, and thus it is unnecessary to utilize Ulusoy's heuristic any more. Besides, EA and LMA also employ different evolutionary operators and evaluation schemes. Recently, Brandão and Eglese proposed a deterministic tabu search algorithm (TSA) [12] for CARP. In their work, two variants of TSA were described, namely TSA1 and TSA2. TSA1 is a rather standard tabu search algorithm, and TSA2 can be viewed as applying TSA1 with a number of different initial solutions. Experimental studies showed that TSA2 is superior to TSA1 and outperformed both CARPET and LMA on three sets of benchmark instances. In [13], we proposed a global repair operator (GRO), which aims to amend low-cost infeasible solutions. It has been shown that combining GRO with TSA1 can lead to significant improvement in terms of solution quality, and may even accelerate convergence of the algorithm.

So far, we have introduced both heuristic and metaheuristic methods. It can be observed that the former was popular in early years while the latter is attracting more and more interest recently. Moreover, many existing metaheuristic methods incorporate previous heuristic methods in their framework. For example, CARPET, LMA, and TSA all employ one or more aforementioned heuristics to obtain initial solutions, and then try to further improve them. Therefore, it is unsurprising that metaheuristic approaches usually outperform heuristic methods in terms of solution quality, although they are computationally more expensive. Fortunately, powerful modern computers can easily afford the additional computational cost.

In this paper, we investigate CARP within the framework of MA. As an emerging area of evolutionary computation, MAs are population-based metaheuristic search methods that combine global search strategies (e.g., crossover) with local search heuristics, and have been studied under a number of different names, such as Baldwinian EAs, Lamarckian EAs, cultural algorithms, genetic local search, etc. They are reported to not only converge to high-quality solutions, but also search more efficiently than conventional EAs. The successes of MAs have been revealed on a wide variety of real-world problems [14]–[16], including CARP, as demonstrated by LMA [11]. Compared to conventional EAs, there are two key issues for the success of MAs. One is an appropriate balance between global and local search, and the other is a cost effective coordination of local search. Hence, the local search procedure, which is usually designed to utilize the domain knowledge of the problem of interest, plays the most important role in MAs. In the context of CARP, local search is often conducted via some traditional move operators, such as single insertion, double insertion, swap, etc. [11]. These move operators modify only a small part of the current solution. More intuitively, they can be said to have small search step size and search within a small neighborhood of the current solution. With such characteristics, these operators can be expected to perform well on simple problems that have a small number of local optima and a small solution space. However, they may no longer work when the solution space becomes large or contains many local optima, or in the case that the solution space consists of separated feasible regions. In such cases, a large-step-size local search may be more desirable, either for jumping out of the local optimum or from one feasible region to another, or to conduct the search more efficiently. From a general optimization viewpoint, the benefits of large step size have been theoretically addressed in [17], where it is proved that simulated annealing (SA) with a larger neighborhood is better than SA with a smaller neighborhood. Unfortunately, the step size issue itself has rarely been addressed in the context of CARP, let alone the design of a refined memetic approach.

Motivated by the above consideration, we propose a new move operator for CARP, named the Merge-Split (MS) operator, in this paper. Compared to traditional move operators, the MS operator has a larger search step size, which is variable. Thus, it can flexibly conduct a local search within a large neighborhood of a solution. We have incorporated the MS operator into the MA framework, and developed the memetic algorithm with extended neighborhood search (MAENS) for CARP. MAENS has been evaluated on four sets of CARP benchmark instances (a total of 181 instances) and compared with five existing metaheuristic algorithms, i.e., CARPET [8], VND [9], GLS [10], LMA [11], and TSA2 [12]. Experimental results showed that MAENS outperformed all the five existing algorithms on difficult instances with many local optima, and performed almost the same as the existing algorithms on simple instances since they all reach the global optimum.

The rest of this paper is organized as follows. Section II introduces preliminary background of this paper, including the formal problem definition of CARP, the general framework of MA, and the traditional move operators for local search. Section III describes the MS operator in detail. After that, MAENS is proposed in Section IV. Section V presents the experimental studies, which include empirical comparison between MAENS and other algorithms and the demonstration of the efficacy of the MS operator. Finally, conclusions and future work will be presented in Section VI.

II. Background

In this section, the background of the paper is presented. We start from the notations, solution representation, and mathematical representation of CARP, and then briefly describe the general framework of MA and traditional move operators for CARP.

A. Notations and Problem Definition

CARP involves seeking a minimum cost routing plan for vehicles to serve all the required edges $E_R \subseteq E$ and required arcs $A_R \subseteq A$ of a given graph $G = (V, E, A)$, subject to some constraints. Each arc task is assigned a unique ID, say t. Each edge (i, j) is considered as a pair of arcs $<i, j>$ and $<j, i>$, one for each direction. Thus, each edge task is assigned two IDs. For the sake of convenience, the IDs are set to positive integers. Each ID t is associated with five features, namely $tail(t)$, $head(t)$, $sc(t)$, $dc(t)$, and $dem(t)$, standing for the tail and head vertices, serving cost, deadheading cost, and

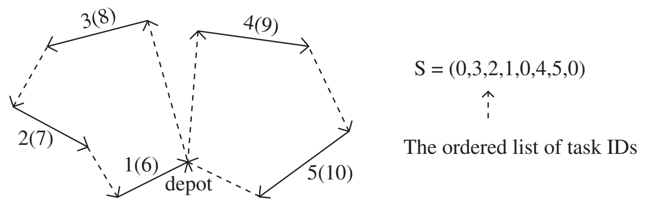


Fig. 1. Illustration of the solution representation. IDs in parentheses represent inversions of the current directions.

```
1 Initialization: Generate an initial population
2 while stopping criteria are not satisfied do
3     Evaluate all individuals in the population
4     Evolve a new population using evolutionary operators
5     for each individual do
6         Perform local search around it with probability $P_{ls}$;
7     end
8 end
```

Fig. 2. General framework of MAs.

demand of the corresponding task, respectively. If t belongs to an edge task, let *inv*(t) denote the inversion of task t. The serving cost, deadheading cost, and demand of task *inv*(t) are the same as *sc*(t), *dc*(t), and *dem*(t), respectively. But note that each edge task should be served only once, in either direction (i.e., only one of tasks t and *inv*(t) is served). To separate different routes in a solution, we also define the *dummy task*. Both the tail and head vertices of a dummy task are the depot vertex *dep*, and its ID is set to 0. Like many other existing metaheuristic approaches, we represent a solution to CARP as an ordered list of tasks (IDs), denoted by $S = (S_1, S_2, \ldots, K)$, where S_i is the ith element (task) of S. Fig. 1 presents a simple illustration of such a solution representation. Given a solution S, the corresponding routing plan can be obtained by connecting every two subsequent tasks with the shortest path between them (i.e., finding a shortest path from the tail vertex of the former task to the head vertex of the subsequent task), which can be easily found by Dijkstra's algorithm [18]. In the context of CARP, the term "shortest path" is equivalent to the path with minimum deadheading cost. Let $sp(S_i,\ S_{i+1})$ denote the total deadheading cost of the shortest path between S_i and S_{i+1}, then the total cost of S can be written as

$$TC(S) = \sum_{i=1}^{length(S)-1} [sc(S_i) + sp(S_i,\ S_{i+1})] \tag{1}$$

where $length(S)$ stands for the length of the sequence S.

From Fig. 1, we may see that each solution S may consist of multiple routes (e.g., the example in Fig. 1 has two routes), and each starts from and ends at the depot. Hence, we can further write S in the form

$$\begin{aligned} S &= (R_1,\ R_2, K, R_m) \\ &= (0,\ \underbrace{R_{11},\ R_{12}, K,}_{R_1}\ 0,\ \underbrace{R_{21},\ R_{22}, K,}_{R_2}\ 0,\ \underbrace{R_{m1},\ R_{m2}, K,}_{R_m}\ 0) \end{aligned} \tag{2}$$

where m is the number of routes in S and each R_i denotes a single route. Obviously, every R_i also consists of a subsequence of tasks, and the load (i.e., total demand) of it is

$$load(R_i) = \sum_{k=1}^{length(R_i)} dem(R_{ik}). \tag{3}$$

Given all the above notations and the aforementioned three constraints of CARP, we now arrive at the following representation of CARP:

$$\begin{aligned} \min_S\ & TC(S) = \sum_{i=1}^{length(S)-1} [sc(S_i) + sp(S_i,\ S_{i+1})] \\ s.t.:\ & app(S_i) = 1, \forall S_i \in A_R \\ & app(S_i) + app(inv(S_i)) = 1, \forall S_i \in E_R \\ & m \le nveh \\ & load(R_i) \le Q \end{aligned} \tag{4}$$

where $app(S_i)$ counts the times that task S_i appears in the whole sequence, $nveh$ is the number of vehicles available at the depot, and Q is the vehicle's capacity.

B. General Framework of Memetic Algorithms (MAs)

First introduced by Moscato in 1989 [19], MAs were inspired by both Darwinian principles of natural evolution and Dawkins' notion of memes [20]. From the evolutionary computation perspective, MAs can be viewed as a form of population-based EAs hybridized with individual learning procedures that are capable of performing local refinements [19]. Without loss of generality, the framework of MAs can be summarized by Fig. 2.

From Fig. 2, we can see that one major difference between MAs and conventional EAs is that the mutation operators of EAs are replaced by local search in MAs. Hence, the success of MAs is largely due to the appropriate adoption of local search operators, and it is not surprising that much important work in the incremental development of MAs is centered around the local search procedure [21]–[24]. Unlike the evolutionary operators, which are usually very general and applicable to various problems, the local search operators are usually expected to incorporate some domain specific heuristics, so that the MAs can balance well between generality and problem specificity. To name a few, local heuristics or conventional exact enumerative methods, such as the Simplex method, Newton/Quasi-Newton method, conjugate gradient method, and line search, are typical local search strategies for numerical optimization. In the context of combinatorial optimization, local search methods are often specifically designed to serve a problem of interest well, e.g., k-gene exchange, the k-opt algorithm for the traveling salesman problem, and many others. In the next section, some traditional local search operators for CARP will be briefly introduced.

C. Traditional Move Operators for Local Search

Recall that a solution to CARP is encoded as a sequence of task IDs, and thus local search around a candidate solution is often conducted by applying move operators to it. In the

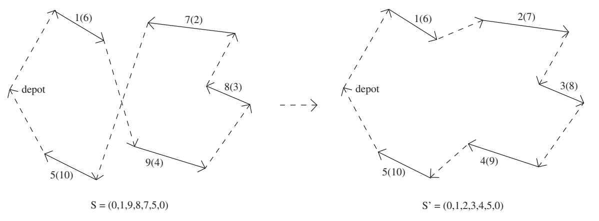


Fig. 3. 2-opt for a single route.

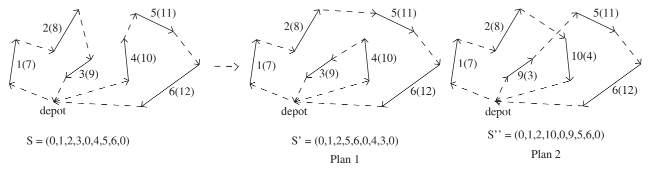


Fig. 4. 2-opt for double routes.

literature, there are four commonly used move operators for CARP, namely single insertion, double insertion, swap, and 2-opt [11].

1) Single Insertion: In the single insertion move, a task is removed from its current position and re-inserted into another position of the current solution or a new empty route. If the selected task belongs to an edge task, both its directions will be considered when inserting the task into the "target position." The direction leading to a better solution will be chosen.

2) Double Insertion: The double insertion move is similar to the single insertion except that two consecutive tasks are moved instead of a single task. Similar to the single insertion, both directions are considered for edge tasks.

3) Swap: In the swap move, two candidate tasks are selected and their positions are exchanged. Similar to the single insertion, both directions are considered for edge tasks.

4) 2-opt: There are two types of 2-opt move operators, one for a single route and the other for double routes. In the 2-opt move for a single route, a subroute (i.e., a part of the route) is selected and its direction is reversed. When applying the 2-opt move to double routes, each route is first cut into two subroutes, and new solutions are generated by reconnecting the four subroutes. Figs. 3 and 4 illustrate the two 2-opt move operators, respectively. In Fig. 3, given a solution $S = (0, 1, 9, 8, 7, 5, 0)$, the subroute from task 9 to 7 is selected and its direction is reversed. In Fig. 4, given a solution $S = (0, 1, 2, 3, 0, 4, 5, 6, 0)$, the first route is cut between tasks 2 and 3, and the second route is cut between tasks 4 and 5. A new solution can be obtained either by connecting task 2 with task 5, and task 4 with task 3, or by linking task 2 to the inversion of task 4, and task 5 with inversion of task 3. In practice, one may choose the one with the smaller cost. Unlike the previous three operators, the 2-opt operator is only applicable to edge tasks. Although it can be easily modified to cope with arc tasks, such work remains absent in the literature.

All the above move operators were first proposed to address the vehicle routing problem (VRP) [25], and were then extended and widely used in CARP [1], [2], [8]–[12], [26]. They adopt rather simple schemes to generate new solutions, and thus are likely to generate new solutions that are quite similar to the current solutions. Intuitively speaking, we may say that these traditional move operators have "small" step size and thus are only capable of searching within a "small" neighborhood. However, a small step-size local search operator might not perform well in a case where a CARP has a large solution space, or the capacity constraints are tight. In the former case, it may take much longer time for a traditional move operator to find the global optimum, i.e., the search process will become inefficient as the solution space enlarges. In the latter case, the solution space will become more rugged and contain more local optima as the capacity constraints become tighter. Consequently, it is likely that the feasible regions in the solution space are isolated by infeasible regions. A small step-size local search might easily be trapped in local optima, and might not be able to "jump" from one feasible solution to another. Therefore, it may never search all the feasible regions appropriately.

Obviously, for both the above cases, a local search with large search step size is more desirable. At first glance, it appears that a large search step size can be obtained with little effort, i.e., we may simply apply the traditional move operators for multiple times. Such an idea can be found in [27], where Liang *et al.* tackled a different type of combinatorial problem—the cutting stock problem. But taking a closer look at the traditional move operators for CARP, we found that all of them define a neighborhood of size $\Theta(n^2)$, where n is the number of tasks. For example, the single insertion selects one task out of n, and there are $n + m - 1$ possible positions for the task to be inserted in, where m is the number of routes. Thus, the single insertion can at most generate $n(n + m - 1)$ different solutions. The swap operator requires selecting two tasks out of n, and there exist $n(n - 1)/2$ different choices. Similar observations can be made upon the double insertion and 2-opt, too. Therefore, consecutively applying single move operators for k times defines a neighborhood of size $O(n^{2k})$, which increases exponentially with k. In consequence, it is prohibitive to enumerate all the possible solutions when k becomes large. One simple solution to this problem is to randomly sample a part of the huge neighborhood. However, it is often the case that some regions in the solution space are more promising than the others. Hence, random sampling is a bit blind and might waste a lot of computational resource. To summarize, although a large step-size local search can be beneficial, it cannot be implemented by simply extending the traditional move operators, and a more refined approach is required. For this purpose, we developed the MS operator.

III. Merge-Split Operator for Local Search

The MS operator aims to improve a given solution by modifying multiple routes of it. As indicated by its name (Merge-Split), this operator is composed of two components, i.e., the Merge and Split. Given a solution, the Merge component randomly selects $p (p > 1)$ routes of it and, combines them together to form an unordered list of task IDs, which contains all the tasks of the selected routes. The Split component directly operates on the unordered list generated by the Merge

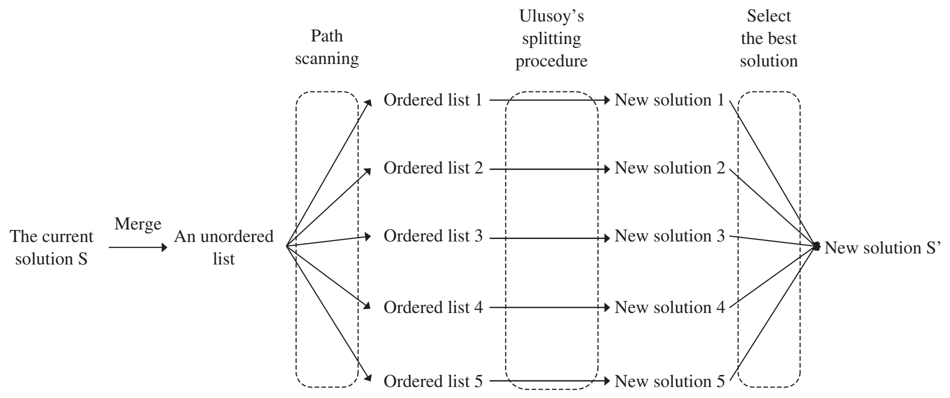


Fig. 5. Merge-Split operator.

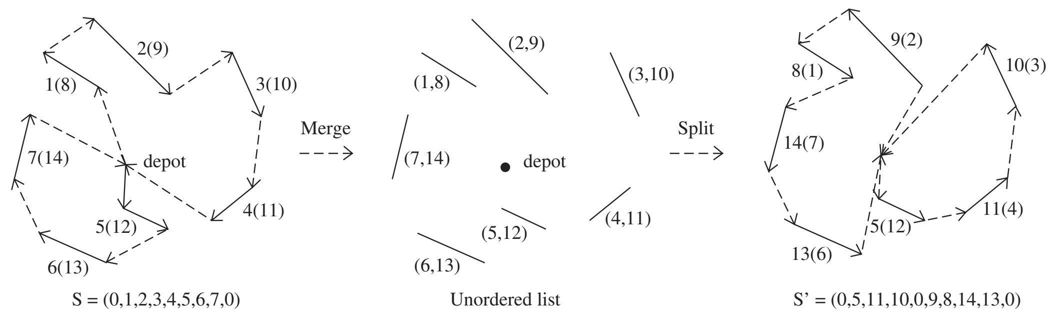


Fig. 6. Demonstration of the Merge-Split operator.

component. First, the path scanning (PS) heuristic [5] is applied. PS starts by initializing an empty path. At each iteration, PS finds out the tasks that do not violate the capacity constraints. If no task satisfies the constraints, it connects the end of the current path to the depot with the shortest path between them to form a route, and then initializes a new empty path. If a unique task satisfies the constraints, PS connects that task to the end of the current path (again, with the shortest path between them). If multiple tasks satisfy the constraints, the one closest to the end of the current path is chosen. If multiple tasks not only satisfy the capacity constraints but are also the closest to the end of the current path, five rules are further adopted to determine which to choose: 1) maximize the distance from the head of task to the depot; 2) minimize the distance from the head of task to the depot; 3) maximize the term $dem(t)/sc(t)$, where $dem(t)$ and $sc(t)$ are demand and serving cost of task t, respectively; 4) minimize the term $dem(t)/sc(t)$; 5) use rule 1) if the vehicle is less than half-full, otherwise use rule 2). If multiple tasks still remain, ties are broken arbitrarily. PS terminates when all the tasks in the unordered list have been selected. Note that PS does not use the five rules alternatively. Instead, it scans the unordered list of tasks for five times. In each scan, only one rule is used. Hence, PS will generate five ordered lists of tasks in total. In the Split component, PS is followed by Ulusoy's splitting procedure [7]. In other words, Ulusoy's splitting method is applied to all the five ordered lists obtained by PS to further improve them. Given an ordered list of tasks, Ulusoy's splitting procedure is an exact algorithm that seeks the optimal way to split the ordered list into different routes. Since Ulusoy's splitting procedure is an exact algorithm and has been well known for years, we omit its detailed steps in this paper. Interested readers may refer to the original publication [7].

To summarize, the MS operator first merges multiple routes to obtain an unordered list of tasks, and then employs PS to sort the unordered list. After that, Ulusoy's splitting procedure is used to split the ordered lists into new routes in the optimal way. Finally, we may obtain five new solutions of CARP by embedding the new routes back into the original solution, and the best one is chosen as the output of the MS operator. Fig. 5 demonstrates the whole process of MS operator.

One main advantage of the MS operator is its capability of generating new solutions that are significantly different from the current solution. As illustrated in Fig. 6, the MS operator generates a solution $S' = (0, 5, 11, 10, 0, 9, 8, 14, 13, 0)$ based on the solution $S = (0, 1, 2, 3, 4, 0, 5, 6, 7, 0)$. If using traditional move operators, S' may be achieved by applying single insertion and double insertion consecutively, but cannot be reached by applying any of the traditional move operators only once. Hence, we may say that the MS operator has a larger search step size than the traditional move operators. In general, the larger the p (i.e., the number of routes involved in MS), the more distant the new solution is from the current solution. Another appealing property of the MS operator is that it is likely to generate high-quality new solutions. This is due to the adoption of PS and Ulusoy's splitting procedure, both of which are known to be capable of generating relatively good solutions. The major drawback of the MS operator is its

```
    Input: A CARP instance, $psize$, $opsize$, $ubtrial$, $P_{ls}$
    Output: A feasible solution $S_{bf}$
    // Initialization:
 1  Set the current population $pop = \emptyset$;
 2  while $|pop| < psize$ do
 3      Set the trial counter $ntrial = 0$;
 4      repeat
 5          Generate an initial solution $S_{init}$;
 6          $ntrial \leftarrow ntrial + 1$;
 7      until $S_{init}$ is not a clone of any solution $S \in pop$ or $ntrial = ubtrial$;
 8      if $S_{init}$ is a clone of some $S \in pop$ then
 9          break;
10      end
11      $pop \leftarrow pop \cup S_{init}$;
12  end
13  $psize = |pop|$;
    // Main Loop:
14  while stopping criterion is not met do
15      Set an intermediate population $pop_t = pop$;
16      for $i = 1 \rightarrow opsize$ do
17          Randomly select two different solutions $S_1$ and $S_2$ as the parents from $pop$;
18          Apply the crossover operator to $S_1$ and $S_2$ to generate $S_x$;
19          Sample a random number $r$ from the uniform distribution between 0 and 1;
20          if $r < P_{ls}$ then
21              Apply local search to $S_x$ to generate $S_{ls}$;
22              if $S_{ls}$ is not a clone of any $S \in pop_t$ then
23                  $pop_t = pop_t \cup S_{ls}$
25 24           else if $S_x$ is not a clone of any $S \in pop_t$ then
26                  $pop_t = pop_t \cup S_x$
27              end
29 28       else if $S_x$ is not a clone of any $S \in pop_t$ then
30              $pop_t = pop_t \cup S_x$
31          end
32      end
33      Sort the solutions in $pop_t$ using stochastic ranking;
34      Set $pop$ = {the best $psize$ solutions in $pop_t$};
35  end
36  return the best feasible solution $S_{bf}$ in $pop$;
```

Fig. 7. Pseudocode of MAENS.

high computational complexity compared to traditional move operators. Fortunately, such a drawback may be more or less alleviated by a careful coordination of the MS operator and other search operators. In the next section, we will propose the MAENS, in which the MS and traditional move operators are integrated to form the local search operator.

IV. Memetic Algorithm with Extended Neighborhood Search for CARP

In this section, we first briefly summarize MAENS. Then the algorithmic details, including initialization, crossover, and local search, will be described.

Like most other MAs, MAENS starts by initializing a population of solutions. At each iteration, crossover and local search are conducted to generate new candidate solutions, i.e., the offspring population. Then, the parent population and offspring population are combined and individuals are sorted using stochastic ranking [28], which is commonly used when applying EAs to constrained optimization problems. Identical solutions, also called clones, are not allowed to appear in a population simultaneously, in order to maintain diversity of the population. This rule applies to both the initial population and all intermediate populations during the search. The Pseudocode of MAENS is provided in Fig. 7, where *psize* is the population size, *opsize* is the number of offspring generated in each generation, *ubtrial* is the maximum number of trials allowed for generating a nonclone initial solution, and P_{ls} is the probability of carrying out local search on an individual solution.

1) Initialization: During the initialization phase, the initial population *pop* is set to empty first. Then nonclone individual solutions are generated and inserted into *pop* one by one. The initialization phase terminates when *psize* nonclone solutions have been generated or no eligible solution has been generated for *ubtrial* consecutive trials. In the latter case, *psize* is reset to the number of solutions finally obtained. Lines 1 to 13 in Fig. 7 demonstrate the initialization phase.

2) Crossover: At each iteration of MAENS, crossover is implemented by applying the sequence based crossover (SBX) operator to two parent individuals randomly selected from the current population. Each pair of parent individuals leads to a single offspring individual. Originally proposed for VRP [29], SBX was designed to work on a sequence of vertex IDs. However, similar to CARP, the aim of VRP is also to sort the elements of the sequence in the optimal order and split them into different groups. Hence, it actually does not matter whether the elements (IDs) represent vertices or edges (arcs) of a graph, and SBX can be applied to CARP with little effort. Given two parent solutions S_1 and S_2, SBX randomly selects two routes R_1 and R_2 from them, respectively. Then, both R_1 and R_2 are further randomly split into two subroutes, say $R_1 = (R_{11}, R_{12})$ and $R_2 = (R_{21}, R_{22})$. After that, a

TABLE I
SUMMARY OF THE PARAMETERS OF MAENS

Name	Description	Value
psize	Population size	30
ubtrial	Maximum trials for generating initial solutions	50
opsize	No. of offspring generated in each generation	6**psize*
P_{ls}	Probability of carrying out local search (mutation)	0.2
p	Number of routes involved in Merge-Split operator	2 (the first set of experiments)
		2, 3, and 4 (the second set of experiments)
G_m	Maximum number of generations	500

TABLE II
RESULTS ON THE *gdb* BENCHMARK TEST SET IN TERMS OF COSTS OF SOLUTIONS. "BEST" AND "AVERAGE" STAND FOR THE BEST AND AVERAGE RESULTS OBTAINED FROM 30 INDEPENDENT RUNS. "NS" STANDS THE NUMBER OF SUCCESSFUL RUNS

Name	$\|V\|$	$\|E\|$	LB	CARPET	VND	LMA	TSA2	MAENS				TSA_{best}
								Average	Std	NS	Best	
1	12	22	316	316	316	316	316	**316.0**	0.0	30	**316**	316
2	12	26	339	339	339	339	339	**339.0**	0.0	30	**339**	339
3	12	22	275	275	275	275	275	**275.0**	0.0	30	**275**	275
4	11	19	287	287	287	287	287	**287.0**	0.0	30	**287**	287
5	13	26	377	377	377	377	377	**377.0**	0.0	30	**377**	377
6	12	22	298	298	298	298	298	**298.0**	0.0	30	**298**	298
7	12	22	325	325	325	325	325	**325.0**	0.0	30	**325**	325
8	27	46	348	352	350	350	348	348. 7	1.0	20	**348**	348
9	27	51	303	317	315	303	303	303.0	0.0	29	**303**	303
10	12	25	275	275	275	275	275	**275.0**	0.0	30	**275**	275
11	22	45	395	395	395	395	395	**395.0**	0.0	30	**395**	395
12	13	23	458	458	458	458	458	**458.0**	0.0	30	**458**	458
13	10	28	536	544	544	536	540	**536.0**	0.0	30	**536**	536
14	7	21	100	100	100	100	100	**100.0**	0.0	30	**100**	100
15	7	21	58	58	58	58	58	**58.0**	0.0	30	**58**	58
16	8	28	127	127	127	127	127	**127.0**	0.0	30	**127**	127
17	8	28	91	91	91	91	91	**91.0**	0.0	30	**91**	91
18	9	36	164	164	164	164	164	**164.0**	0.0	30	**164**	164
19	8	11	55	55	55	55	55	**55.0**	0.0	30	**55**	55
20	11	22	121	121	121	121	121	**121.0**	0.0	30	**121**	121
21	11	33	156	156	156	156	156	**156.0**	0.0	30	**156**	156
22	11	44	200	200	200	200	200	**200.0**	0.0	30	**200**	200
23	11	55	233	235	235	233	235	**233.0**	0.0	30	**233**	233
Mean	—	—	253.8	255.0	254.8	253.9	254.0	253.8	—	29.5	253.8	253.8
No. best	—	—	—	19	19	22	21	21	—	—	**23**	**23**
APD	—	—	—	0.35	0.30	0.03	0.07	0.01	—	—	**0.00**	**0.00**

new route is obtained by replacing R_{12} with R_{22}. Finally, it is possible that some tasks appear more than once in the new route, or some tasks in R_1 are no longer served in the new route. In the former case, the duplicated tasks are removed from the new route. In the latter case, the missing tasks are re-inserted into the new route. The re-insertions may induce additional cost and violation of the capacity constraints. Hence, each missing task is re-inserted into such a position that re-insertion into any other position will not induce both lower additional cost and smaller violation of the capacity constraints. If multiple positions satisfy this condition, one of them will be chosen arbitrarily.

3) Local Search: After generating the offspring population using SBX, each offspring will be further improved by a local search with probability P_{ls}. In the previous section, we have introduced some traditional move operators for local search, and proposed the novel MS operator. Now, the question is, how to make use of these operators to conduct local search effectively. As mentioned before, traditional move operators have small search step size, while the step size of the MS operator is relatively large and generally increases with the number of routes involved in it. Numerous previous papers on EAs have shown that neither type of search operators is universally the best. A small step-size operator may be easily trapped in local optima, while a large step size might miss the global optimum when the current solution is close to it. Hence, a natural idea is to search in a small region around the current solution first. When a local optimum is reached, we extend the search step size, trying to jump out from it. If successful, the step size will be reduced again to exploit the new local region to the full extent. Following this idea, we developed the local search procedure of MAENS as below.

TABLE III
RESULTS ON THE *val* BENCHMARK TEST SET IN TERMS OF COSTS OF SOLUTIONS. "BEST" AND "AVERAGE" STAND FOR THE BEST AND AVERAGE RESULTS OBTAINED FROM 30 INDEPENDENT RUNS. "NS" STANDS THE NUMBER OF SUCCESSFUL RUNS

Name	$\|V\|$	$\|E\|$	LB	CARPET	VND	LMA	TSA2	MAENS				TSA_{best}
								Average	Std	NS	Best	
1A	24	39	173	173	173	173	173	**173.0**	0.0	30	**173**	173
1B	24	39	173	173	173	173	173	**173.0**	0.0	30	**173**	173
1C	24	39	245	245	245	245	245	**245.0**	0.0	30	**245**	245
2A	24	34	227	227	227	227	227	**227.0**	0.0	30	**227**	227
2B	24	34	259	260	259	259	259	**259.0**	0.0	30	**259**	259
2C	24	34	457	494	457	457	457	457.2	1.1	29	**457**	457
3A	24	35	81	81	81	81	81	**81.0**	0.0	30	**81**	81
3B	24	35	87	87	87	87	87	**87.0**	0.0	30	**87**	87
3C	24	35	138	138	140	138	138	**138.0**	0.0	30	**138**	138
4A	41	69	400	400	400	400	400	**400.0**	0.0	30	**400**	400
4B	41	69	412	416	414	412	412	**412.0**	0.0	30	**412**	412
4C	41	69	428	453	428	428	428	431.1	3.1	14	**428**	428
4D	41	69	526	556	544	541	530	532.9	3.3	15	**530**	530
5A	34	65	423	423	423	423	423	**423.0**	0.0	30	**423**	423
5B	34	65	446	448	449	446	446	**446.0**	0.0	30	**446**	446
5C	34	65	473	476	474	474	474	**474.0**	0.0	30	**474**	474
5D	34	65	573	607	599	583	583	582.9	2.2	21	**577**	577
6A	31	50	223	223	223	223	223	**223.0**	0.0	30	**223**	223
6B	31	50	233	241	233	233	233	**233.0**	0.0	30	**233**	233
6C	31	50	317	329	325	317	317	**317.0**	0.0	30	**317**	317
7A	40	66	279	279	279	279	279	**279.0**	0.0	30	**279**	279
7B	40	66	283	283	283	283	283	**283.0**	0.0	30	**283**	283
7C	40	66	334	343	335	334	334	**334.0**	0.0	30	**334**	334
8A	30	63	386	386	386	386	386	**386.0**	0.0	30	**386**	386
8B	30	63	395	401	403	395	395	**395.0**	0.0	30	**395**	395
8C	30	63	518	533	543	527	529	525.9	1.7	29	**521**	521
9A	50	92	323	323	323	323	323	323.0	0.0	29	**323**	323
9B	50	92	326	329	326	326	326	**326.0**	0.0	30	**326**	326
9C	50	92	332	332	336	332	332	**332.0**	0.0	30	**332**	332
9D	50	92	385	409	399	391	391	**391.0**	0.0	30	**391**	391
10A	50	97	428	428	428	428	428	**428.0**	0.0	30	**428**	428
10B	50	97	436	436	436	436	436	436.0	0.0	29	**436**	436
10C	50	97	446	451	446	446	446	**446.0**	0.0	30	**446**	446
10D	50	97	525	544	538	530	530	533.6	1.5	0	531	528
Mean	—	—	343.2	350.8	347.5	345.2	344.9	345.1	—	27.8	344.5	344.4
No. best	—	—	—	16	22	30	32	26	—	—	33	**34**
APD	—	—	—	1.60	0.82	0.27	0.22	0.26	—	—	0.15	**0.13**

Given an offspring individual S_x generated by crossover, three traditional move operators, i.e., the single insertion, double insertion, and swap, are separately applied to the individual to conduct a best improvement local search. That is, for each move operator, all solutions that can be reached by it from S_x are examined, and the current solution will only be updated when a better solution has been found. This procedure terminates when the current solution can no longer be improved. By this means, we will obtain three new solutions, and the best one will be chosen as the output of this stage. Since this stage conducts exhaustive search within the neighborhoods defined by the three operators, its output is assured to be a local optimum. After that, the MS operator is applied to this local optimal solution, forming the second stage of local search. Assume that the current solution consists of m routes and the MS operator works on p of them, where p is the predefined parameter of the MS operator. Then, the MS operator may at most generate C_m^p different solutions. This number can be really huge for a large m. Besides, generating a single solution using MS is already much more time consuming than the traditional move operators. Hence, it might be prohibitive to enumerate all the C_m^p possible solutions when C_m^p is too large. For this reason, we restrict the MS operator to generating 100 solutions at most in this stage, which is implemented by random sampling among the C_m^p possibilities (all the solutions will be enumerated if $C_m^p \leq 100$). Again, this stage is a best improvement procedure. The current solution will be updated only when the MS operator has generated a better solution. If the MS operator manages to find a better solution, we further apply the three traditional move operators in exactly the same way as the first stage to exploit the new local region. Otherwise, the second stage

TABLE IV

RESULTS ON THE *egl* BENCHMARK TEST SET IN TERMS OF COSTS OF SOLUTIONS. "BEST" AND "AVERAGE" STAND FOR THE BEST AND AVERAGE RESULTS OBTAINED FROM 30 INDEPENDENT RUNS. "NS" STANDS THE NUMBER OF SUCCESSFUL RUNS

Name	$\|V\|$	$\|R\|$	$\|E\|$	LB	LMA	TSA2	MAENS				TSA_{best}
							Average	Std	NS	Best	
E1-A	77	51	98	3548	3548	3548	3548.0	0.0	30	**3548**	3548
E1-B	77	51	98	4498	4498	4533	4516.5	17.6	13	**4498**	4498
E1-C	77	51	98	5566	5595	5595	5601.6	9.9	20	**5595**	5595
E2-A	77	72	98	5018	5018	5018	5018.0	0.0	30	**5018**	5018
E2-B	77	72	98	6305	6340	6343	6341.4	12.0	6	**6317**	6317
E2-C	77	72	98	8243	8415	8347	8355.7	35.9	22	**8335**	8335
E3-A	77	87	98	5898	5898	5902	5898.8	2.9	27	**5898**	5898
E3-B	77	87	98	7704	7822	7816	7802.9	27.3	23	**7775***	7777
E3-C	77	87	98	10163	10433	10309	10321.9	18.0	11	**10292***	10305
E4-A	77	98	98	6408	6461	6473	6475.2	10.3	1	**6456**	6456
E4-B	77	98	98	8884	9021	9063	9023.0	18.7	17	**8998***	9000
E4-C	77	98	98	11427	11779	11627	11645.8	46.7	9	**11561***	11601
S1-A	140	75	190	5018	5018	5072	5039.8	35.9	20	**5018**	5018
S1-B	140	75	190	6384	6435	6388	6433.4	8.6	1	**6388**	6388
S1-C	140	75	190	8493	8518	8535	8518.3	1.5	28	**8518**	8518
S2-A	140	147	190	9824	9995	10038	9959.2	34.6	28	**9895***	9956
S2-B	140	147	190	12968	13174	13178	13231.6	63.2	7	**13147***	13165
S2-C	140	147	190	16353	16795	16505	16509.8	51.8	16	**16430***	16505
S3-A	140	159	190	10143	10296	10451	10312.7	26.5	6	**10257***	10260
S3-B	140	159	190	13616	14053	13981	13876.6	67.8	29	**13749***	13807
S3-C	140	159	190	17100	17297	17346	17305.8	41.4	7	**17207***	17234
S4-A	140	190	190	12143	12442	12462	12419.2	33.2	24	**12341**	12341
S4-B	140	190	190	16093	16531	16490	16441.2	38.1	28	**16337***	16442
S4-C	140	190	190	20375	20832	20733	20767.2	74.6	8	**20538***	20591
Mean	–	–	–	9673.8	9842.3	9823	9806.8	–	17.1	9756.8	9773.9
No. best	–	–	–	–	7	4	2	–	–	**24**	12
APD	–	–	–	–	1.38	1.3	1.14	–	–	**0.70**	0.84

terminates. This accomplishes the local search procedure of MAENS.

In addition to the above descriptions, two issues of the local search procedure still remain to be further elaborated. The first one is how to choose the parameter p of the MS operator. Generally speaking, p can be set to any integer larger than 1. But recall that C_m^p increases rapidly with p, thus the chance of finding the best one among C_m^p possibilities by random sampling will decrease with p and eventually deteriorate the performance of the operator. Besides, a larger step size never guarantees a better solution. Therefore, we recommend $p = 2$ as the default choice. As will be demonstrated by our experimental results, $p = 2$ performs much better than other values such as 3 and 4.

The final important issue that must be addressed is how to evaluate a solution and determine whether it is better than the current solution. In particular, since CARP is a constrained optimization problem, it is inevitable that the search process will encounter infeasible solutions. As stated at the beginning of this section, in each generation of MAENS, the parent and offspring populations are combined together after crossover and local search. Then, the stochastic ranking [28] will be applied to obtain the parent population for the next generation. Let lower ranks indicate better individuals and stochastic ranking sort individuals through a bubble-sort-like procedure. Each pair of adjacent individuals is compared and their ranks will be swapped if the individual with higher rank is better. If the two compared individuals are both feasible, comparison will be made solely according to the fitness. Otherwise, the two individuals will be compared either according to their fitness or according to their constraint violations, with a predefined probability. Although the stochastic ranking can be readily applied to the local search procedure, it is inappropriate to do so due to the high computational cost involved. For each individual, using the stochastic ranking in local search involves a time complexity of $O(2n^2N_{iter})$, where n is the number of tasks and N_{iter} is the number of iterations of the traditional move operators. Given that n is typically larger than 50 for most CARP instances studied in the literature, it may take quite a long time to carry out the stochastic ranking for a single individual at each iteration of the local search. Hence, we resort to a much simpler method here. That is, evaluating the quality of a solution by a weighted sum of the extent it violates the constraints and its cost, as given in (5)

$$f(S) = TC(S) + \lambda^* TV(S) \qquad (5)$$

where $TC(S)$ is the total cost of S and $TV(S)$ is the total violation of it, which can be calculated by summing up the violations of all routes in S. λ is a penalty parameter that balances the tradeoff between cost and violation. The solution with the smallest value of $f(S)$ is considered as the best solution obtained by local search. The use of (5) reduces the

TABLE V

RESULTS ON SET C OF THE BENCHMARK SETS OF BEULLENS ET AL. IN TERMS OF COSTS OF SOLUTIONS. "BEST" AND "AVERAGE" STAND FOR THE BEST AND AVERAGE RESULTS OBTAINED FROM 30 INDEPENDENT RUNS. "NS" STANDS THE NUMBER OF SUCCESSFUL RUNS

Name	$\|V\|$	$\|R\|$	$\|E\|$	LB	GLS	TSA2	MAENS				TSA$_{\text{best}}$
							Average	Std	NS	Best	
C1	69	79	98	1590	1660	1660	1707.0	23.9	5	**1660**	1660
C2	48	53	66	1095	1095	1095	1095.7	3.7	29	**1095**	1095
C3	46	51	64	875	925	925	927.8	3.9	18	**925**	925
C4	60	72	84	1285	1340	1340	1342.7	4.5	22	**1340**	1340
C5	56	65	79	2410	2475	2470	2522.3	30.0	4	**2470**	2470
C6	38	51	55	855	895	895	907.5	3.4	1	**895**	895
C7	54	52	70	1735	1795	1795	**1795.0**	0.0	30	**1795**	1795
C8	66	63	88	1640	1730	1730	1732.3	4.3	23	**1730**	1730
C9	76	97	117	1775	1825	1830	1852.8	21.5	7	**1820**	1820
C10	60	55	82	2190	2290	2270	2317.8	43.8	10	**2270**	2270
C11	83	94	118	1725	1815	1815	1853.7	34.1	7	**1815**	1815
C12	62	72	88	1510	1610	1610	**1610.0**	0.0	30	**1610**	1610
C13	40	52	60	1050	1110	1110	1122.0	21.4	17	**1110**	1110
C14	58	57	79	1620	1680	1680	1687.3	10.8	15	**1680**	1680
C15	97	107	140	1765	1860	1865	1896.5	16.3	2	**1860**	1860
C16	32	32	42	580	585	585	585.2	0.9	29	**585**	585
C17	43	42	56	1590	1610	1610	1618.3	17.8	24	**1610**	1610
C18	93	121	133	2315	2410	2415	2411.7	18.9	20	**2390***	2410
C19	62	61	84	1345	1395	1400	1425.7	19.1	1	**1395**	1395
C20	45	53	64	665	665	665	668.5	6.7	21	**665**	665
C21	60	76	84	1705	1725	1725	1725.2	0.9	29	**1725**	1725
C22	56	43	76	1070	1070	1070	**1070.0**	0.0	30	**1070**	1070
C23	78	92	109	1620	1690	1700	1724.3	30.8	8	**1690**	1700
C24	77	84	115	1330	1360	1360	1368.5	6.2	7	**1360**	1360
C25	37	38	50	905	905	905	907.0	7.6	28	**905**	905
Mean	—	—	—	1449.8	1500.8	1501.0	1515.0	—	16.7	**1498.8**	1500.0
No. best	—	—	—	—	21	20	3	—	—	**25**	23
APD	—	—	—	—	3.30	3.32	4.19	—	—	**3.21**	3.27

computational complexity to $O(2nN_{iter})$ and thus alleviates the computational cost involved in local search. As stated in the constraint handling literature, determining an appropriate value of λ is a nontrivial task [30]. Intuitively, since $TV(S)$ is usually much smaller than $TC(S)$, normalization is required to make the two terms within the same magnitude, so that $TV(S)$ will not be neglected. Furthermore, λ should decrease with the total cost of the current solution while increase with the total violation, so as to provide different biases to different solutions. Based upon these considerations, we have designed an adaptive λ for MAENS. When conducting local search around solution S, λ is first initialized as

$$\lambda = \frac{TC(S_{\text{best}})}{Q} * \left(\frac{TC(S_{\text{best}})}{TC(S)} + \frac{TV(S)}{Q} + 1\right) \quad (6)$$

where S_{best} represents the best feasible solution found so far. In (6), $TC(S_{\text{best}})/Q$ functions as the normalization factor. The term $TC(S_{\text{best}})/TC(S)$ makes λ decrease with the total cost of S, and the term $TV(S)/Q$ makes λ increase with the constraint violation of S. The term "1" is included to ensure a sufficiently large λ in case S is a feasible solution with a very large cost. During the local search procedure, λ is halved if feasible solutions have been reached for five consecutive iterations and is doubled if infeasible solutions have been reached for five consecutive iterations.

V. EXPERIMENTAL STUDIES

To evaluate the efficacy of the MS operator and MAENS, two sets of experiments have been carried out. In the first set, we comprehensively compared MAENS to a number of state-of-the-art algorithms. After that, the effect of the MS operator was investigated. In particular, the performance of MAENS was studied in four scenarios, that is, removing the MS operator from MAENS and setting p to 2, 3, and 4.

A. Experimental Setup

All the experiments were carried out on four benchmark test sets of CARP instances, referred to as the *gdb* set [31], the *val* set [32], the *egl* set [33]–[35], and the Beullens *et al.*'s sets [10]. The *gdb* set was generated by DeArmon in [31] and consists of 23 instances. The *val* set was generated by Benavent *et al.* in [32]. It contains 34 instances based on 10 different graphs. Different instances based on each graph were generated by changing the capacity of the vehicles. The *egl* set was generated by Eglese based on data from a winter gritting application in placeLancashire [33]–[35]. It consists of 24 instances based on two graphs, each with a distinct set of required edges and capacity constraints. The test set generated by Beullens *et al.* in [10] is based on the intercity road network in Flanders. It further contains four

TABLE VI

RESULTS ON SET *D* OF THE BENCHMARK SETS OF BEULLENS ET AL. IN TERMS OF COSTS OF SOLUTIONS. "BEST" AND "AVERAGE" STAND FOR THE BEST AND AVERAGE RESULTS OBTAINED FROM 30 INDEPENDENT RUNS. "NS" STANDS THE NUMBER OF SUCCESSFUL RUNS

Name	$\|V\|$	$\|R\|$	$\|E\|$	LB	GLS	TSA2	MAENS				TSA_{best}
							Average	Std	NS	Best	
D1	69	79	98	725	725	740	745.0	0.0	0	745	740
D2	48	53	66	480	480	480	**480.0**	0.0	30	**480**	480
D3	46	51	64	415	415	415	415.2	0.9	29	**415**	415
D4	60	72	84	615	615	615	616.0	3.8	28	**615**	615
D5	56	65	79	1040	1040	1040	**1040.0**	0.0	30	**1040**	1040
D6	38	51	55	485	485	485	493.0	15.6	23	**485**	485
D7	54	52	70	735	835	835	847.3	17.7	13	**835**	835
D8	66	63	88	615	685	685	704.2	15.5	7	**685**	685
D9	76	97	117	680	680	680	**680.0**	0.0	30	**680**	680
D10	60	55	82	900	910	910	**910.0**	0.0	30	**910**	910
D11	83	94	118	920	930	960	935.2	6.2	4	**920***	940
D12	62	72	88	680	680	680	**680.0**	0.0	30	**680**	680
D13	40	52	60	690	690	695	691.0	2.0	24	**690**	690
D14	58	57	79	920	930	940	931.0	3.1	27	**930**	930
D15	97	107	140	910	910	950	919.0	3.1	3	**910**	950
D16	32	32	42	170	170	170	**170.0**	0.0	30	**170**	170
D17	43	42	56	675	675	675	**675.0**	0.0	30	**675**	675
D18	93	121	133	930	930	930	934.2	8.7	23	**930**	930
D19	62	61	84	650	680	690	**680.0**	0.0	30	**680**	680
D20	45	53	64	415	415	415	415.2	0.9	29	**415**	415
D21	60	76	84	695	805	825	834.2	18.8	0	810	815
D22	56	43	76	690	690	690	**690.0**	0.0	30	**690**	690
D23	78	92	109	715	735	735	748.2	8.4	3	**735**	735
D24	77	84	115	620	670	670	683.5	19.3	18	**670**	670
D25	37	38	50	410	410	410	**410.0**	0.0	30	**410**	410
mean	–	–	–	671.2	687.6	693.2	693.1	–	21.2	688.2	690.6
No. best	–	–	–	–	**24**	18	10	–	–	23	21
APD	–	–	–	–	**2.38**	3.02	3.18	–	–	2.48	2.74

subsets, namely the sets C, D, E, and F, each of which contains 25 instances. Instances of sets D and F share the same networks with instances of sets C and E, respectively, but with a larger capacity of vehicles. In total, 181 instances were used in our studies. Although MAENS can be directly applied to the mixed CARP without any modification, all these benchmark test sets are undirected CARP only, i.e., all the tasks are edge tasks. It is well known that mixed CARP is more challenging than undirected CARP. However, few (if any) benchmark instances of mixed CARP are publicly available. Hence, we had to restrict our experimental study to undirected CARP.

Throughout the experiments, MAENS adopted the same parameters. The algorithm was terminated when a predefined number of generations were reached. Table I summarizes the parameter settings of MAENS used in the experiments. All the experiments were conducted for 30 independent runs, and the best and average results obtained are reported in this paper.

B. Comparing MAENS to Existing Algorithms

We consider five existing algorithms in the comparative study: including CARPET [8], VND [9], GLS [10], LMA [11], and TSA2 [12]. Results of these algorithms are directly obtained from the original publications. Despite the existence of some other algorithms for CARP, it has been shown [12] that the above algorithms are state of the art. For each CARP instance we have studied (except for the instance 10D of the *val* set), at least one of the five algorithms has been reported to obtain the best known solution. Hence, comparison to these algorithms would be sufficient for our evaluation. On the *gdb* set and the *val* set, results of CARPET, VND, LMA, and TSA2 are available. LMA and TSA have been applied to the *egl* set only. GLS and TSA have been compared on the Beullens *et al.*'s sets. Tables II to VIII present the cost of the final solutions obtained by the compared algorithms on all the 4 test sets. A brief description of the contents in the tables is given as below.

1) The columns headed $|V|$, $|R|$, and $|E|$ indicate the number of vertices, required edges, and total edges, respectively. Since all edges are required to be served in the *gdb* and *val* sets, the column $|R|$ is omitted from Tables II and III.
2) The columns headed LB give the lower bounds found so far for the instances, which were collected from [10], [12], [36]–[39].
3) For MAENS, the columns headed Average and Std provide the average results and standard deviations calculated over the 30 runs. The columns headed NS stand for the number of successful runs of MAENS. Here, for each instance, a run is considered to be successful

TABLE VII

RESULTS ON SET E OF THE BENCHMARK SET OF BEULLENS ET AL. IN TERMS OF COSTS OF SOLUTIONS. "BEST" AND "AVERAGE" STAND FOR THE BEST AND AVERAGE RESULTS OBTAINED FROM 30 INDEPENDENT RUNS. "NS" STANDS THE NUMBER OF SUCCESSFUL RUNS

Name	$\|V\|$	$\|R\|$	$\|E\|$	LB	GLS	TSA2	MAENS				TSA_{best}
							Average	Std	NS	Best	
E1	73	85	105	1855	1940	1935	1967.8	35.0	3	**1935**	1935
E2	58	58	81	1580	1610	1610	1615.5	14.2	24	**1610**	1610
E3	46	47	61	750	750	750	752.0	4.1	24	**750**	750
E4	70	77	99	1580	1610	1615	1684.3	21.1	2	**1610**	1615
E5	68	61	94	2130	2170	2160	2228.7	49.0	3	**2160**	2160
E6	49	43	66	670	670	670	**670.0**	0.0	30	**670**	670
E7	73	50	94	1780	1900	1900	**1900.0**	0.0	30	**1900**	1900
E8	74	59	98	2080	2150	2155	2150.5	1.5	27	**2150**	2150
E9	93	103	141	2160	2250	2300	2327.7	38.2	1	**2235***	2295
E10	56	49	76	1690	1690	1690	1691.5	5.7	28	**1690**	1690
E11	80	94	113	1810	1850	1855	1932.0	44.5	2	1850	1840
E12	74	67	103	1580	1710	1730	1764.3	17.9	0	1710	1705
E13	49	52	73	1300	1325	1325	1335.3	22.2	24	**1325**	1325
E14	53	55	72	1780	1810	1810	1817.0	10.9	20	**1810**	1810
E15	85	107	126	1555	1610	1610	1617.8	13.0	12	**1595***	1610
E16	60	54	80	1785	1825	1825	**1825.0**	0.0	30	**1825**	1825
E17	38	36	50	1290	1290	1290	1294.3	6.3	18	**1290**	1290
E18	78	88	110	1600	1610	1610	1612.3	6.3	26	**1610**	1610
E19	77	66	103	1400	1435	1435	1437.0	3.1	20	**1435**	1435
E20	56	63	80	950	990	990	**990.0**	0.0	30	**990**	990
E21	57	72	82	1700	1705	1705	1755.5	22.9	2	**1705**	1705
E22	54	44	73	1155	1185	1185	1187.5	6.3	25	**1185**	1185
E23	93	89	130	1395	1430	1445	1469.0	13.1	0.0	1435	1435
E24	97	86	142	1695	1785	1785	1822.2	26.3	6	**1785**	1785
E25	26	28	35	655	655	655	**655.0**	0.0	30	**655**	655
Mean	—	—	—	1517.0	1558.2	1561.6	1580.1	—	16.7	1556.6	1559.2
No. best	—	—	—	—	19	18	5	—	—	**22**	21
APD	—	—	—	—	2.51	2.69	3.77	—	—	**2.41**	2.54

only if the obtained solution is not worse than the solutions obtained by the compared algorithms. Finally, the columns headed "best" present the cost of the best solutions obtained among the 30 runs.

4) For TSA2, two types of solutions were reported in [12]. The columns headed TSA2 present the results obtained by TSA2 using the same standard parameter settings for all instances, while results given in the columns headed TSA_{best} were obtained by fine-tuning the parameters of TSA2 for each instance so as to get the best solutions.
5) For each table, three additional rows are included at the bottom. The first row presents the average values calculated for each algorithm over all the instances in one test set. The average values of lower bounds were also calculated for reference. The second row summarizes the number of instances on which the algorithm has achieved the best solution among the compared methods. The third row calculates for each algorithm the average percentage deviation (APD) from the lower bounds.
6) In all the tables, results are highlighted in bold for the instances on which MAENS achieved the best solutions among the compared methods. Results with "*" indicate that MAENS has attained new best known solutions (i.e., never found before) for the corresponding instances.
7) In [10], results on Beullens *et al.*'s four test subsets were reported in terms of the cost of deadheading only. Hence, we present the results of MAENS in the same form in Tables V–VIII.

The efficacy of MAENS can be evaluated from two perspectives, i.e., the best and the average performance it has achieved in the 30 independent runs. Since MAENS is a stochastic algorithm while most of the compared algorithms (except LMA) are deterministic, it might be unfair to make comparisons based on the best performance of MAENS. Nevertheless, such comparison does provide some information on MAENSs search capability. Considering the columns headed "best," it can be found that MAENS obtained the best solutions on 175 out of 181 CARP instances, which is significantly better than the compared algorithms. In particular, even after fine-tuning the parameters for each instance, TSA2 performed the best one on 152 instances, not to mention the remaining algorithms. More importantly, MAENS managed to find **new** best known solutions on 12 instances of the *egl* set and 4 instances of the benchmark sets of Beullens *et al.* In terms of APD, MAENS outperformed the other algorithms on the *egl* set and the sets C and E of Beullens *et al.* It also obtained the lowest APD on the *gdb* set and the set F of Beullens *et al.*, while being only slightly inferior to the best algorithms

TABLE VIII

RESULTS ON SET F OF THE BENCHMARK SETS OF BEULLENS ET AL. IN TERMS OF COSTS OF SOLUTIONS. "BEST" AND "AVERAGE" STAND FOR THE BEST AND AVERAGE RESULTS OBTAINED FROM 30 INDEPENDENT RUNS. "NS" STANDS THE NUMBER OF SUCCESSFUL RUNS

Name	$\|V\|$	$\|R\|$	$\|E\|$	LB	GLS	TSA2	MAENS Average	Std	NS	Best	TSA_{best}
F1	73	85	105	1065	1065	1085	1071.0	7.9	13	**1065**	1070
F2	58	58	81	920	920	920	**920.0**	0.0	30	**920**	920
F3	46	47	61	400	400	400	**400.0**	0.0	30	**400**	400
F4	70	77	99	930	940	960	963.5	8.4	1	**940**	945
F5	68	61	94	1180	1180	1180	1180.3	1.3	28	**1180**	1180
F6	49	43	66	490	490	490	**490.0**	0.0	30	**490**	490
F7	73	50	94	1080	1080	1080	1090.7	24.5	24	**1080**	1080
F8	74	59	98	1135	1145	1145	**1145.0**	0.0	30	**1145**	1145
F9	93	103	141	1145	1145	1170	1197.8	27.6	2	**1145**	1155
F10	56	49	76	1010	1010	1010	**1010.0**	0.0	30	**1010**	1010
F11	80	94	113	1015	1015	1015	1037.5	16.3	7	**1015**	1015
F12	74	67	103	900	910	910	939.5	33.0	11	**910**	910
F13	49	52	73	835	835	835	**835.0**	0.0	30	**835**	835
F14	53	55	72	1025	1025	1035	1065.5	14.4	3	**1025**	1035
F15	85	107	126	945	945	990	951.7	9.9	20	**945**	965
F16	60	54	80	775	775	775	**775.0**	0.0	30	**775**	775
F17	38	36	50	605	605	630	**605.0**	0.0	30	**605**	605
F18	78	88	110	835	850	850	861.2	23.3	18	**850**	850
F19	77	66	103	685	725	740	**725.0**	0.0	30	**725**	725
F20	56	63	80	610	610	610	614.8	0.9	1	**610**	610
F21	57	72	82	905	905	905	**905.0**	0.0	30	**905**	905
F22	54	44	73	790	790	790	**790.0**	0.0	30	**790**	790
F23	93	89	130	705	725	730	736.3	14.3	12	**725**	730
F24	97	86	142	975	975	1010	1001.3	17.1	4	**975**	1010
F25	26	28	35	430	430	430	**430.0**	0.0	30	**430**	430
Mean	—	—	—	855.6	859.8	867.8	869.6	—	20.2	859.8	863.4
No. best	—	—	—	—	**25**	16	12	—	—	**25**	18
APD	—	—	—	—	**0.54**	1.44	1.55	—	—	**0.54**	0.91

TABLE IX

RUNTIME (IN CPU SECONDS) OF THE COMPARED ALGORITHMS ON THE TEST SETS (AVERAGED OVER 30 RUNS FOR MAENS)

Test set	CARPET	VND	GLS	LMA	TSA2	MAENS
gdb	4.9	0.5	—	2.6	1.7	6.3
val	34.6	2.1	—	21.6	14.1	68.1
egl	—	—	—	263.5	204.0	702.1
Set C of Beullens *et al.*	—	—	59.0	—	60.1	233.1
Set D of Beullens *et al.*	—	—	24.1	—	26.9	309.2
Set E of Beullens *et al.*	—	—	56.5	—	64.4	226.6
Set F of Beullens *et al.*	—	—	14.8	—	29.5	235.0

on the *val* set and the set D of Beullens *et al.* All of the above observations are evidence that MAENS is capable of achieving better solutions than the state-of-the-art algorithms over a wide range of different test problem instances, although such capability may not be guaranteed due to its stochastic nature.

Instead of using the best performance, comparing MAENS to other algorithms based upon its average performance might be fairer. For a similar reason, it would be inappropriate to take the results of TSA_{best} into account, because it tunes parameters for individual instances. Therefore, only the standard TSA2 should be adopted. From the tables, one may find that the average performance of MAENS is also very competitive. MAENSs average results are comparable to the best solutions in all cases. In fact, the average performance of MAENS is still better than CARPET, VND, and LMA (as shown in Tables II–IV). Furthermore, MAENS has always obtained the best solutions for most instances from the *gdb* and *val* sets. Since LMA is the only population-based (and the only stochastic) algorithm among the five compared algorithms, comparison between MAENS and LMA might be of particular interest. Unfortunately, it is not clearly stated in [11] whether the results reported were averaged over a number of independent runs, or just the best ones obtained. Hence, a truly fair comparison is difficult. Nevertheless, given the fact that the average results of MAENS are better than the results reported for LMA, we may conclude that MAENS is superior in terms of solution quality. In addition to the quality of solutions, it is

TABLE X
AVERGE APD OF MAENS ON THE BENCHMARK SETS WITH DIFFERENT VALUE OF "p"

Test set	Best of the 30 runs				Average of the 30 runs			
	Without MS	$P = 2$ for MS	$P = 3$ for MS	$P = 4$ for MS	Without MS	$P = 2$ for MS	$P = 3$ for MS	$P = 4$ for MS
gdb	**0.00**	**0.00**	**0.00**	**0.00**	0.16	0.01	0.01	**0.00**
val	0.23	**0.15**	**0.14**	0.20	0.56	**0.26**	0.33	0.38
egl	1.51	**0.70**	0.77	0.82	2.53	**1.14**	1.20	1.35
Set C of Beullens *et al.*	3.39	**3.21**	3.27	3.38	5.45	**4.19**	4.27	4.53
Set D of Beullens *et al.*	2.54	**2.48**	2.62	2.75	4.04	**3.18**	3.61	3.83
Set E of Beullens *et al.*	2.94	**2.41**	2.62	2.71	5.15	**3.77**	3.82	4.12
Set F of Beullens *et al.*	0.71	**0.54**	0.65	0.79	2.49	**1.55**	1.82	2.08

natural to ask how stable MAENS is, i.e., how likely MAENS would perform well. This issue can be evaluated by counting the number of successful runs out of 30. Here, a run is regarded as successful only when MAENS obtains a solution with the lowest cost among all the compared algorithms. As shown in the tables, the average number of successful runs of MAENS is larger than 15 for all the benchmark test sets. In particular, MAENS always obtained the best solution (i.e., succeed in all the 30 runs) on 79 out of 181 instances. Such observation more or less demonstrates the stability of MAENS.

However, MAENS suffers from a high computational cost introduced by the MS operator. In Table IX, the average runtimes of each compared algorithm on the four test sets are presented. In our experiments, MAENS was coded in C language and run using an Intel(R) Xeon(R) E5335 2.00 GHz. Since the compared algorithms were implemented on different computers, normalization has been carried out to make fair comparisons on the runtimes. That is, all the runtimes presented in this paper were obtained by dividing the runtimes in the original publications by some factors. To be specific, CARPET and VND were implemented on the Graphics Indigo2 (195 MHz); therefore the runtimes presented in [8] and [9] were divided by 10. The results of GLS in [10] were obtained using a Pentium II 500 MHz, and therefore we divided the runtimes there by 4. LMA was implemented on a Pentium III 1 GHz [11] and the TSA on a Pentium Mobile 1.4 GHz [12]. Hence, the runtimes of LMA and TSA given in the corresponding papers have been divided by 2 and 10/7, respectively. It should be noted that the comparison between the CPU time is meant to be indicative, because we do not have access to other information that influences the computation time, such as the operating systems, compilers, coding skills of the programmer, etc. From Table IX, it can be found that MAENS is much more time consuming than all the compared algorithms. We will address this issue in our future work. Some potential ideas will be discussed in the next section.

C. Further Analysis of the Effects and Settings of the Merge-Split Operator

As mentioned before, four scenarios were considered in this set of experiments. To investigate the contributions of the MS operator to MAENS, we removed it from the algorithm while keeping all the other parts unchanged. Besides, we also set the parameter p of the MS operator to 3 and 4 to investigate how it influences the performance of MAENS. The resultant three algorithms were then run on all the instances for 30 times and compared to the MAENS with $p = 2$. Table X summarizes the results, where the best and average results obtained are given. For the sake of brevity, only the average APD for each benchmark set is given. The best results are highlighted in bold. We may see clearly that MAENS performed the best when $p = 2$, and its performance generally deteriorated with the increase of p. On the other hand, though setting p to 3 or 4 led to inferior performance; they still outperformed the MA without MS operator. For example, if we consider the average performance, incorporation of MS operator with $p = 3$ and 4 both resulted in enhanced results on all benchmark sets. In summary, experimental results show that the MS operator is indeed crucial for our algorithm and $p = 2$ is a reasonably good default choice if no prior knowledge suggests other alternatives.

Finally, we can observe that incorporation of MS resulted in the most significant improvement of MAENS on the *egl* set, while it is not that obvious on the *gdb* and *val* sets. Taking a closer look at the CARP instances, it can be found that the *egl* set consists of quite a few challenging instances, which have 190 edge tasks. On the other hand, instances of the *gdb* and *val* sets contain no more than 100 edge tasks. That means, instances of the *egl* set generally have larger solution spaces. Therefore, the success of MAENS on the *egl* set seems to validate our hypothesis that a large step-size local search operator, such as the MS operator, would benefit those problems with large solution space.

VI. CONCLUSION AND FUTURE WORK

In this paper, we investigated CARP within the framework of MA. First, a novel local search operator, the MS operator, was proposed. Unlike existing local search operators, which only modify a small part of the candidate solutions, the MS operator may result in significantly different solutions. Hence, the MS operator is capable of searching using large step size. For this reason, the MS operator is less likely to be trapped in local optima. Second, we proposed the MAENS algorithm. MAENS combines the advantages of both the traditional and MS operators in its local search procedure so as to attain a good tradeoff between exploitation and exploration. Based upon our comprehensive experimental studies, two main conclusions can be drawn. First of all, MAENS is capable of achieving better solutions than the state-of-the-art algorithms. MAs are very promising approaches to CARP and deserve more in-depth investigation. Second, removing the MS operator from MAENS led to significant deterioration of the quality of solutions, and thus the MS operator plays an important role in the success of MAENS. The same conclusion can also be drawn from the comparison between MAENS and LMA, since LMA employs the same framework and traditional local search operators as MAENS, but does not contain the MS operator.

The importance of large search step size has been clearly demonstrated for numerical optimization for a long time [40], [41] and it is well acknowledged that a good combination of large and small step sizes can lead to better solutions to numerical optimization problems. The success of the MS operator showed that similar insight also holds for combinatorial problems. It is worth noting that the MS operator is not specifically proposed for MAENS, but is applicable to ANY existing algorithms for CARP. Given a candidate solution obtained by any existing algorithm, the MS operator can be applied to it.

Although MAENS has shown excellent performance in our experimental studies, it is by no means perfect. The main disadvantage of MAENS, i.e., the high computational cost, remains to be addressed in the future. To reduce the time complexity of MAENS, one may consider a more refined scheme of carrying out local search. It has been commonly stated in the literature that not all individuals in the population deserve a local search. Hence, heuristics can be included in MAENS to apply local search to only those "promising" (e.g., low cost and feasible) individuals. Another potential improvement to MAENS might be to incorporate a time-variant parameter p. To be specific, one may design a scheme to monitor the search process of MAENS and decide which value of p to use based upon the status of the current population. Such a scheme might make MAENS converge faster to even better solutions.

References

[1] H. Handa, D. Lin, L. Chapman, and X. Yao, "Robust solution of salting route optimisation using evolutionary algorithms," in *Proc. IEEE Congr. Evol. Comput. 2006*, Vancouver, BC, Canada, pp. 3098–3105.

[2] 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.

[3] M. Dror, *Arc Routing, Theory, Solutions, and Applications*. Boston, MA: Kluwer, 2000.

[4] B. L. Golden and R. T. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, no. 3, pp. 305–315, 1981.

[5] B. L. Golden, J. S. DeArmon, and E. K. Baker, "Computational experiments with algorithms for a class of routing problems," *Comput. Oper. Res.*, vol. 10, no. 1, pp. 47–59, 1983.

[6] H. I. Stern and M. Dror, "Routing electric meter readers," *Comput. Oper. Res.*, vol. 6, no. 4, pp. 209–223, 1979.

[7] G. Ulusoy, "The fleet size and mix problem for capacitated arc routing," *Eur. J. Oper. Res.*, vol. 22, no. 3, pp. 329–337, 1985.

[8] A. Hertz, G. Laporte, and M. Mittaz, "A tabu search heuristic for the capacitated arc routing problem," *Oper. Res.*, vol. 48, no. 1, pp. 129–135, 2000.

[9] A. Hertz and M. Mittaz, "A variable neighborhood descent algorithm for the undirected capacitated arc routing problem," *Transport. Sci.*, vol. 35, no. 4, pp. 425–434, 2001.

[10] P. Beullens, L. Muyldermans, D. Cattrysse, and D. V. Oudheusden, "A guided local search heuristic for the capacitated arc routing problem," *Eur. J. Oper. Res.*, vol. 147, no. 3, pp. 629–643, 2003.

[11] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Competitive memetic algorithms for arc routing problem," *Ann. Oper. Res.*, vol. 131, no. 1–4, pp. 159–185, 2004.

[12] J. Brandao 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.

[13] 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.

[14] E. Ozcan and E. Onbasioglu, "Memetic algorithms for parallel code optimization," *Int. J. Parallel Programm.*, vol. 35, no. 1, pp. 33–61, 2006.

[15] M. Tang and X. Yao, "A memetic algorithm for VLSI floorplanning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 62–69, Feb. 2007.

[16] Z. Zhu, Y. S. Ong, and M. Dash, "Wrapper-filter feature selection algorithm using a memetic framework," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 70–76, Feb. 2007.

[17] X. Yao, "Simulated annealing with extended neighbourhood," *Int. J. Comput. Math.*, vol. 40, no. 3, pp. 169–189, 1991.

[18] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[19] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms," Caltech Concurrent Comput. Program, CalTech, Pasadena, CA, Rep. 826, 1989.

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

[21] K.-H. Liang, X. Yao, and C. Newton, "Evolutionary search of approximated N-dimensional landscapes," *Int. J. Knowledge-Based Intell. Eng. Syst.*, vol. 4, no. 3, pp. 172–183, 2000.

[22] K. W. C. Ku, M. W. Mak, and W. C. Siu, "A study of the lamarckian evolution of recurrent neural networks," *IEEE Trans. Evol. Comput.*, vol. 4, no. 1, pp. 31–42, Apr. 2000.

[23] Y. S. Ong and A. J. Keane, "Meta-lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, Apr. 2004.

[24] N. K. Bambha, S. S. Bhattacharyya, J. Teich, and E. Zitzler, "Systematic integration of parameterized local search into evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 137–155, Apr. 2004.

[25] M. Dror and L. Levy, "A vehicle routing improvement algorithm comparison of a 'greedy' and a matching implementation for inventory routing," *Comput. Oper. Res.*, vol. 13, no. 1, pp. 33–45, 1986.

[26] P. A. Mullaseril, M. Dror, and J. Leung, "Split-delivery routing heuristic in livestock feed distribution," *J. Oper. Res. Soc.*, vol. 48, no. 2, pp. 107–116, 1997.

[27] K.-H. Liang, X. Yao, C. S. Newton, and D. Hoffman, "A new evolutionary approach to cutting stock problems with and without contiguity," *Comput. Oper. Res.*, vol. 29, no. 12, pp. 1641–1659, 2002.

[28] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, no. 3, pp. 284–294, Sep. 2000.

[29] J. Y. Potvin and S. Bengio, "The vehicle routing problem with time windows part II: Genetic search," *Informs J. Comput.*, vol. 8, no. 2, pp. 165–172, 1996.

[30] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Comput. Methods Appl. Mech. Eng.*, vol. 191, no. 11–12, pp. 1245–1287, 2002.

[31] J. S. DeArmon, "A comparison of heuristics for the capacitated Chinese postman problems," M.S. thesis, Univ. Maryland, College Park, MD, 1981.

[32] E. Benavent, V. Campos, E. Corberan, and E. Mota, "The capacitated arc routing problem. Lower bounds," *Networks*, vol. 22, no. 4, pp. 669–690, 1992.

[33] R. W. Eglese, "Routing winter gritting vehicles," *Discrete Appl. Math.*, vol. 48, no. 3, pp. 231–244, 1994.

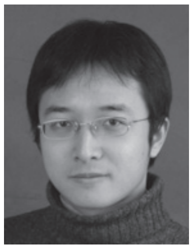
[34] R. W. Eglese and L. Y. O. Li, "A tabu search based heuristic for arc routing with a capacity constraint and time deadline," in *Metaheuristics: Theory and Applications*, Boston, MA: Kluwer, 1996, pp. 633–650.

[35] L. Y. O. Li and R. W. Eglese, "An interactive algorithm for vehicle routing for winter-gritting," *J. Oper. Res. Soc.*, vol. 47, no. 2, pp. 217–228, 1996.

[36] J. M. 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.

[37] D. Ahr, "Contributions to multiple postmen problems," Ph.D. dissertation, Rupercht-Karls-Univer., Heidelberg, Germany, 2004.

[38] R. Baldacci and V. Maniezzo, “Exact methods based on node-routing formulations for undirected arc-routing problems,” *Networks*, vol. 47, no. 1, pp. 52–60, 2006.
[39] H. Longo, D. A. M. Poggi, 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.
[40] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, Jul. 1999.
[41] C. Y. Lee and X. Yao, “Evolutionary programming using the mutations based on the Lévy probability distribution,” *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 1–13, Feb. 2004.



Ke Tang (S’05–M’07) received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002 and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007.

He is currently an Associate Professor with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. He is the coauthor of more than 30 refereed publications. His major research interests include machine learning, pattern analysis, evolutionary computation, data mining, metaheuristic algorithms, and real-world applications.

Dr. Tang is an editorial board member of three international journals and the Chair of IEEE Task Force on Large Scale Global Optimization.

Yi Mei (S’09) received the B.S. degree in mathematics from the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei, in 2005, and is currently working toward the Ph.D. degree.

His current research interests include memetic algorithm, tabu search, and other metaheuristics for solving arc routing problems.

Xin Yao (M’91–SM’96–F’03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, China, in 1985, and the Ph.D. degree from USTC in 1990.

From 1985 to 1990, he was an Associate Lecturer and Lecturer with USTC, while working toward the Ph.D. degree in simulated annealing and evolutionary algorithms. In 1990, he was a Postdoctoral Fellow with the Computer Sciences Laboratory, Australian National University, Canberra, Australia, where he continued his work on simulated annealing and evolutionary algorithms. In 1991, he was with the Knowledge-Based Systems Group, Commonwealth Scientific and Industrial Research Organization, Division of Building, Construction and Engineering, Melbourne, Australia, where he worked primarily on an industrial project on automatic inspection of sewage pipes. In 1992, he returned to Canberra to take up a lectureship in the School of Computer Science, University College, University of New South Wales, Australian Defense Force Academy, Sydney, Australia, where he was later promoted to a Senior Lecturer and Associate Professor. Since April 1999, he has been a Professor (Chair) of computer science in the University of Birmingham, Birmingham, U.K. He is currently the Director of the Center of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, Birmingham, U.K. and also a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, USTC. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. He has more than 300 referenced publications. His major research interests include evolutionary artificial neural networks, automatic modularization of machine-learning systems, evolutionary optimization, constraint-handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner’s dilemma, data mining, and real-world applications.

Dr. Yao was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008, an Associate Editor or editorial board member of 12 other journals, and the Editor of the World Scientific Book Series on Advances in Natural Computation. He was the recipient of the President’s Award for Outstanding Thesis by the Chinese Academy of Sciences for his Ph.D. work on simulated annealing and evolutionary algorithms in 1989. He was the recipient of the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.