

# A Hybrid Ant Colony Optimization Algorithm for the Extended Capacitated Arc Routing Problem

Li-Ning Xing, Philipp Rohlfshagen, Ying-Wu Chen, and Xin Yao, *Fellow, IEEE*

**Abstract**—The capacitated arc routing problem (CARP) is representative of numerous practical applications, and in order to widen its scope, we consider an extended version of this problem that entails both total service time and fixed investment costs. We subsequently propose a hybrid ant colony optimization (ACO) algorithm (HACOA) to solve instances of the extended CARP. This approach is characterized by the exploitation of heuristic information, adaptive parameters, and local optimization techniques: Two kinds of heuristic information, arc cluster information and arc priority information, are obtained continuously from the solutions sampled to guide the subsequent optimization process. The adaptive parameters ease the burden of choosing initial values and facilitate improved and more robust results. Finally, local optimization, based on the two-opt heuristic, is employed to improve the overall performance of the proposed algorithm. The resulting HACOA is tested on four sets of benchmark problems containing a total of 87 instances with up to 140 nodes and 380 arcs. In order to evaluate the effectiveness of the proposed method, some existing capacitated arc routing heuristics are extended to cope with the extended version of this problem; the experimental results indicate that the proposed ACO method outperforms these heuristics.

**Index Terms**—Ant colony optimization (ACO), capacitated arc routing problem (CARP), combinatorial optimization, memetic algorithms (MAs).

## I. INTRODUCTION

THE capacitated arc routing problem (CARP) is concerned with the routing of entities (such as vehicles) across some given topology (e.g., a road network) in order to carry out some task as efficiently as possible. This problem naturally arises in many industrial sectors, i.e., the routing of street sweepers [1], snow plows [2], household refuse collection vehicles [3] or gritting (salting) trucks [4], the planning of mail delivery [5] or school bus service [6], and the inspection of electric power lines [7], gas pipelines [8], or oil pipelines [9].

Manuscript received January 31, 2010; revised July 20, 2010 and January 4, 2011; accepted January 15, 2011. This work was supported in part by the National Natural Science Foundation of China under Grants 70971131 and 70801062 and in part by Engineering and Physical Sciences Research Council Grant EP/E058884/1 on “Evolutionary Algorithm for Dynamic Optimisation Problems: Design, Analysis and Applications.” This paper was recommended by Associate Editor J. Wang.

L.-N. Xing and Y.-W. Chen are with the Department of Management Science and Engineering, College of Information System and Management, National University of Defense Technology, Changsha 410073, China (e-mail: xinglining@gmail.com; chen.nudt@gmail.com).

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

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSMCB.2011.2107899

Lacomme *et al.* [10] recently introduced an extended CARP (ECARP) that takes additional practical constraints into consideration. In particular, the authors proposed the following four extensions: 1) a mixed multigraph with two kinds of links and parallel links; 2) two distinct costs per link; 3) prohibited turns and turn penalties; and 4) a maximum trip length. These extensions allow for a more realistic model and hence may afford solutions that are applicable in a wider set of scenarios. However, neither total service time nor fixed investment costs were considered by Lacomme *et al.*

In this paper, a new ECARP is proposed that considers these aspects which we deem critical with respect to practical requirements: With the development of service-oriented governments [11], each government office is responsible to provide a timely service to the public. At the same time, a strong competition necessitates a flexible and dynamic company model that facilitates agile delivery. Furthermore, the service architecture (e.g., infrastructure) should be optimized to satisfy the ever-changing requirements of the customers. Whereas the traditional service mission is “can do it” (i.e., provide the optimal service within the existing service architecture), the current service mission is “do it best” (i.e., provide the optimal service within an optimized service architecture). In terms of the ECARP, both the service scheduling (a set of vehicle routes) and the service architecture (a configuration of depots and vehicles) need to be optimized simultaneously to improve the quality of service. For this reason, both the maximum total service time and the fixed investment cost (which is applied to optimize the service architecture) are considered in this paper.

Taking additional requirements into account inevitably increases the complexity of what is an already complex problem. The application of metaheuristics—which tend to find solutions of satisfactory quality for difficult optimization problems within reasonable time—is thus a viable option. One such approach is ant colony optimization (ACO) which has gained popularity in recent years because of numerous success stories [12]–[15]. For this reason, a novel hybrid ACO algorithm (HACOA) is proposed to tackle the new and extended version of ECARP. This approach, which is described in detail in Section IV, is characterized by the exploitation of heuristic information, adaptive parameters, and local optimization techniques. The combination of these techniques allows the algorithm to perform reliably across the benchmark instances considered in this paper.

The remainder of this paper is organized as follows. Section II reviews some recent work related to the CARP. Section III formulates this ECARP. The HACOA is outlined in Section IV. Section V analyzes the performance of the algorithm, and Section VI gives some concluding remarks as well as directions for future work.

## II. LITERATURE REVIEW

The CARP in an NP-hard optimization problem and exact approaches, such as branch-and-bound [16], can deal with only very small instances of 20–30 edges [10]. Subsequently, heuristics are required for solving larger problem instances. Examples include Augment-Merge [17], Path-Scanning [18], Construct-and-Strike [19], Ulusoy's tour splitting algorithm [20], and Augment-Insert [21]. Although these heuristics generally produce fairly good results, better results can be obtained by metaheuristics [22]–[25].

Beullens *et al.* [26] proposed the guided local search (GLS) method for the CARP. The GLS was extended by some new moves (relocate, exchange, and cross) and also made use of lists of neighbors and edge marking. Experiments on standard benchmark problems demonstrate that the GLS finds all the best known upper bounds and also improved the bounds for some of the instances. On a set of new benchmark problems, the GLS consistently produced high-quality solutions and, in many cases, located the optimal solution.

Doerner *et al.* [27] employed an ACO algorithm to deal with the CARP. Experimental results obtained from 23 instances generated by DeArmon [28] show that the ACO algorithm is superior to Tabu Scatter Search [29] and the Tabu search proposed by Hertz *et al.* [30] but inferior to Lacomme's memetic algorithm (MA) [10]. Hertz and Mittaz [31] developed the variable neighborhood descent (VND) method for the CARP and tested it on 23 instances generated by DeArmon [28], 34 instances by Benavent *et al.* [32], and 270 instances produced by Hertz *et al.* [30]. The results demonstrate that CARPET and VND are superior to a range of simple heuristics. Polacek *et al.* [33] proposed a variable neighborhood search (VNS) for the CARP with intermediate facilities. The VNS is a simple and robust method which tackles the CARP as well as some of its extensions. This VNS displays an excellent performance on four different sets of benchmarks. Hertz *et al.* [30] constructed a Tabu Search (TS) heuristic for the CARP, while Brandão and Eglese [34] developed a deterministic Tabu Search Algorithm (TSA) for the CARP. Mei *et al.* [22] proposed an operator called global repair operator (GRO) to address the major disadvantage encountered by most of the traditional algorithms: They inserted GRO to the TSA and apply the resultant Repair-based Tabu Search (RTS) algorithm to five well-known benchmark test sets. Experimental results suggest that RTS is competitive with a number of state-of-the-art approaches.

Lacomme *et al.* proposed several different evolutionary algorithms for the CARP [10], [35]–[37]. As mentioned in the introduction, Lacomme *et al.* proposed an ECARP [10] which considered attributes such as mixed networks, parallel arcs, and turn penalties. The authors also proposed an MA [10], and experimental results demonstrate that the MA outperforms a range of other heuristics on 81 instances with up to 140 nodes and 190 edges. The authors explain the algorithm's performance as follows: 1) Individual solutions are strongly improved by local search; 2) small populations of distinct solutions tend to avoid premature convergence; 3) some good solutions are obtained using classical heuristics; and 4) the partial replacement technique used for restarts improves the solution quality of population.

Tang *et al.* [23] proposed an MA with extended neighborhood search (MAENS). Unlike existing approaches, MAENS employs a novel local search operator called Merge-Split that is capable of large step sizes and thus has the potential to search the solution space more efficiently. Experimental results suggest that MAENS is superior to a number of state-of-the-art algorithms.

The amount of research directed toward the multidepot CARP (MCARP) is considerably less. Amberg *et al.* [38] proposed a TS using capacitated trees to solve the multiple-center CARP. Li *et al.* [39] proposed an improved genetic algorithm to the MCARP in 2009. Xing *et al.* [40] constructed a novel evolutionary approach to the MCARP that integrates ECARP heuristics into a novel evolutionary framework. Kansou and Yassine deal the MCARP with two different approaches: ACO combined with an insertion heuristic and MA based on a special crossover [41], [42]. Their computational results on benchmark instances suggest the superiority of the MA compared to the ACO [41].

The other extended versions of CARPs include the following: periodic CARPs [43]–[45], split-delivery CARPs [46], CARPs with intermediate facilities [33], CARPs with time windows [47], CARPs with refill points [48], CARPs with stochastic demands [49], and so on.

All the existing methods developed for the CARP assume a predefined and fixed service architecture (i.e., the number and position of the depots and the number of vehicles are fixed). Motivated by practical requirements (as outlined in Section I), an extended version of the traditional CARP is proposed in the next section where both the service scheduling (a set of vehicle routes) and the service architecture (a configuration of depots and vehicles) need to be optimized simultaneously to improve the quality of service.

## III. PROBLEM FORMULATION

In comparison with the ECARP proposed by Lacomme *et al.* [10], two important extensions are addressed in this work. The first extension considers the maximum service time: Service time is defined as the vehicle's traveling time which is equivalent to the total time required by the vehicle to carry out the tasks assigned to it. The consideration of this aspect may lead to shorter overall service times. The second extension deals with the optimization of the service architecture, which means that both the number and the position of depots are variable. This extension would guarantee the availability of services (the restriction on the service time places hard constraints on the problem: If the road network is large or the service time is small, some roads cannot be serviced because of their distance from the depot).

Our ECARP is modeled as the directed connected graph in which each arc has a deadheading (traveling without service) cost. A subset of arcs with known demands and additional service costs must be serviced by the vehicles. A penalty may be imposed depending on the sequence of arcs traversed. For instance, U-turns and left turns may incur a penalty. A fleet of identical trucks, each with the same purchase cost, capacity, and speed, is employed to service the arcs. The number of

TABLE I  
MATHEMATICAL SYMBOLS USED IN THE PROBLEM FORMULATION

symbol	meaning
$G_U = (V, E)$	An undirected connected graph.
$G_D = (V, A)$	A directed connected graph.
$V$	A set of $N_V$ nodes.
$E$	A set of $N_E$ edges.
$A$	A set of $N_A$ arcs.
$R$	A set of $N_R$ required links $R \subseteq A$ ( $R \subseteq E$ ).
$dd(u)$	The cost of link $u$ (the monolithic cost).
$d(u)$	The deadheading cost of link $u$ .
$s(u)$	The additional service cost of required link $u$ .
$q(u)$	The gritting demand of required link $u$ .
$pen(u, v)$	The penalty occurs from arc $u$ to arc $v$ .
$N_1$	The number of depots.
$N_2$	The number of trucks.
$Q$	The capacity of trucks.
$S$	The average speed of trucks.
$T$	The maximal service time.
$C$	The maximal trip length.
$\alpha$	Deadheading cost conversion coefficient.
$L$	A set of vehicle routes $L = \{l_1, l_2, \dots, l_{N_2}\}$ .
$l_j$	A vehicle route $l_j = \{\mu_{1j}, \mu_{2j}, \dots, \mu_{(m_j)j}\}$ .
$m_j$	The number of links in route $l_j$ .
$Start(l_j)$	The start depot of route $l_j$ .
$\mu_{ij}$	The $i^{th}$ link in vehicle route $l_j$ .
$f_j(\mu_{ij})$	The service indicator of link $\mu_{ij}$ .
$D(l_j)$	The total deadheading cost of route $l_j$ .
$S(l_j)$	The total service cost of route $l_j$ .
$P(l_j)$	The total penalty of route $l_j$ .
$M$	The same task is executed $M$ times in a given period.
$Depot$	The positions of depots, $Depot = \{d_1, d_2, \dots, d_{N_V}\}$ , where $d_i = 1$ indicates that node $i$ should be selected to be a depot.
$DT$	The distribution of trucks, $DT = \{dt_1, dt_2, \dots, dt_{N_1}\}$ , where $dt_i$ denotes the number of trucks to be stored at depot $i$ .
$DR$	The distribution of resources, $DR = \{dr_1, dr_2, \dots, dr_{N_1}\}$ , where $dr_i$ corresponds to the amount of resources which should be reserved at depot $i$ .

vehicles and depots is considered to be a decision variable. Every required arc is serviced by one single trip, and each trip starts and ends at the same depot. The total demand of a trip must not exceed the vehicle's capacity. All services must finish within the maximum service time, and no partial services are allowed. At any depot, the dispatched truck and resource must not exceed its own (the service capacity of each depot is considered). A solution consists of determining a configuration of depots and vehicles as well as a set of vehicle routes; the goal is to minimize the sum of fixed and running costs.

Here, we will formally describe the standard CARP, the Lacomme's ECARP [10], and our ECARP and elaborate further on the attributes unique to our ECARP. These problems are characterized using inputs, outputs, objectives, and constraints. We have listed the attributes of different problems in itemized form to allow a direct comparison among these three problem variants.

### A. Preliminary Knowledge

The symbols used in this section are summarized in Table I. The following concepts are illustrated for the sake of clarity.

- 1) *Mixed connected graph*: The mixed graph allows the existence of two kinds of links: undirected edges and directed arcs. In this paper, a mixed connected graph is transformed into a directed connected graph through replacing each edge by two arcs with opposite directions.

- 2) *Type of arcs*: Each edge can be regarded as a bidirectional arc, and every arc can be regarded as one unidirectional arc.
- 3) *Deadheading cost*: It will incur the deadheading cost no matter whether a link in the given route is serviced or not.
- 4) *Service time*: In this paper, for simplification, only the traveling time of vehicles is considered as the service time. There is a linear relationship between the deadheading cost and the service time. Here, the deadheading cost is transformed into the traveling distance of trucks by multiplying a coefficient  $\alpha$ .
- 5) *Service indicator*:  $f_j(\mu_{ij})$  is a Boolean variable. If arc  $\mu_{ij}$  is serviced in route  $l_j$ , then  $f_j(\mu_{ij}) = 1$ ; else,  $f_j(\mu_{ij}) = 0$ .
- 6) *Different costs of each route*: The total deadheading cost  $D(l_j)$ , the total service cost  $S(l_j)$ , and the total penalty  $P(l_j)$  of route  $l_j$  can be computed as follows:

$$D(l_j) = \sum_{i=1}^{m_j} d(\mu_{ij}) \quad (1)$$

$$S(l_j) = \sum_{i=1}^{m_j} s(\mu_{ij}) f_j(\mu_{ij}) \quad (2)$$

$$P(l_j) = \sum_{i=1}^{m_j} pen(\mu_{ij}, \mu_{(i+1)j}). \quad (3)$$

- 7) *Different constraints*: Six different kinds of constraints are summarized as follows.
  - a) Start and return the same depot. Any route must start and return the same depot. That is, to each route  $l_j$  ( $1 \leq j \leq N_2$ ), the initial node of the first arc must be the same as the end node of the last arc.
  - b) Demand constraints. No demand exceeds the vehicle's capacity  $Q$ . For every route  $l_j$

$$\sum_{i=1}^{m_j} q(\mu_{ij}) f_j(\mu_{ij}) \leq Q, 1 \leq j \leq N_2. \quad (4)$$

- c) No partial service. An arc cannot be serviced by more than one truck, and each required arc must be serviced exactly once.
- d) Trip length constraints. The total length of any trip must not exceed the maximum trip length  $C$

$$\max_{1 \leq j \leq N_2} \{D(l_j) + S(l_j)\} \leq C. \quad (5)$$

- e) Service time constraints. The total service time of any truck must not exceed the maximum service time  $T$

$$\max_{1 \leq j \leq N_2} \left\{ \sum_{i=1}^{m_j} d(\mu_{ij}) \right\} \leq \frac{S \times T}{\alpha} \quad (6)$$

where  $S \times T/\alpha$  denotes the maximum deadheading cost of trips.

- f) Resource constraints. If there are  $dt_i$  trucks stored at depot  $i$ , then, at most,  $dt_i$  trucks can be dispatched from depot  $i$ . Also, if there are  $dr_i$  resources reserved at depot  $i$ , then, at most,  $dr_i$  resources can be dispatched from depot  $i$ .

### B. Problem Formulation of CARP

#### 1) Inputs

- a) An undirected connected graph  $G_U = (V, E)$ .
- b) Each edge  $u \in E$  incurs a cost  $dd(u)$ .
- c) Each edge  $u \in R$  has a demand  $q(u)$ .
- d) One depot is available.
- e)  $N_2$  identical trucks are available.
- f) The capacity of trucks is  $Q$ .

#### 2) Outputs

- a) An optimal set of vehicle routes  $L$ .
- b) Service indicators  $f_j(\mu_{ij})$ .

#### 3) Objectives

$$\min \sum_{j=1}^{N_2} D(l_j). \quad (7)$$

#### 4) Constraints

- a) Start and return the same depot.
- b) Demand constraints.
- c) No partial service.

### C. Problem Formulation of Lacomme's ECARP

#### 1) Inputs

- a) A directed connected graph  $G_D = (V, A)$ .
- b) Each arc  $u \in A$  incurs a deadheading cost  $d(u)$ .
- c) Each arc  $u \in R$  incurs an additional service cost  $s(u)$ .
- d) Each arc  $u \in R$  has a demand  $q(u)$ .
- e) A penalty  $pen(u, v)$  may occur when traversing any two successive arcs  $u, v \in A$ .
- f) One depot is available.
- g)  $N_2$  identical trucks are available.
- h) The capacity of trucks is  $Q$ .
- i) The maximum trip length  $C$ .

#### 2) Outputs

- a) An optimal set of vehicle routes  $L$ .
- b) Service indicators  $f_j(\mu_{ij})$ .

#### 3) Objectives

$$\min \sum_{j=1}^{N_2} \{D(l_j) + S(l_j) + P(l_j)\}. \quad (8)$$

#### 4) Constraints

- a) Start and return the same depot.
- b) Demand constraints.
- c) No partial service.
- d) Trip length constraints.

### D. Problem Formulation of Our ECARP

#### 1) Inputs

- a) A directed connected graph  $G_D = (V, A)$ .
- b) Each arc  $u \in A$  incurs a deadheading cost  $d(u)$ .
- c) Each arc  $u \in R$  incurs an additional service cost  $s(u)$ .
- d) Each arc  $u \in R$  has a demand  $q(u)$ .
- e) A penalty  $pen(u, v)$  may occur when traversing any two successive arcs  $u, v \in A$ .
- f) The cost of depots is denoted by  $C_1$ .
- g) The cost of trucks is denoted by  $C_2$ .

h) The capacity of trucks is  $Q$ .

i) The average speed of trucks is  $S$ .

j) The maximal service time is  $T$ .

k) Deadheading cost conversion coefficient  $\alpha$ .

l) In a given period, the same task is executed  $M$  times.<sup>1</sup>

#### 2) Outputs

A solution to the ECARP needs to specify both the decision variables regarding the system architecture as well as a set of routes that satisfies the constraints.

a) The number of depots  $N_1$ .

b) The positions of depots  $Depot$ .

c) The number of trucks  $N_2$ .

d) The distribution of trucks  $DT$ .

e) The distribution of resources  $DR$ .

f) An optimal set of vehicle routes  $L$ .

g) Service indicators  $f_j(\mu_{ij})$ .

h) The start depot  $Start(l_j)$  of route  $l_j$ .

#### 3) Objectives

The objective of the ECARP is to minimize the total cost

$$F = CF + CR \quad (9)$$

where  $CF$  denotes the fixed cost

$$CF = C_1 N_1 + C_2 N_2 \quad (10)$$

and  $CR$  denotes the running cost

$$CR = M \sum_{j=1}^{N_2} Cost(l_j) \quad (11)$$

with  $Cost(l_j)$  denoting the total cost of route  $l_j$

$$Cost(l_j) = D(l_j) + S(l_j) + P(l_j). \quad (12)$$

#### 4) Constraints

a) Start and return the same depot.

b) Demand constraints.

c) No partial service.

d) Service time constraints.

e) Resource constraints.

According to problem formulations, significant differences between our ECARP and Lacomme's ECARP are evident. In Lacomme's ECARP, the number and positions of the depots as well as the number of the vehicles are all predefined. In our ECARP, these quantities need to be optimized to improve the overall quality of service. Furthermore, Lacomme's ECARP instances only have a single depot, whereas, here, we consider the possibility of multiple depots. These differences prevent the application of Lacomme's MA.

## IV. HYBRID ACO

The ACO framework was introduced by Dorigo *et al.* and presents a novel nature-inspired metaheuristic that has been used successfully in the past to obtain high-quality solutions

<sup>1</sup>To make a tradeoff between fixed costs and running costs, the same task should be executed for many times in the given period.

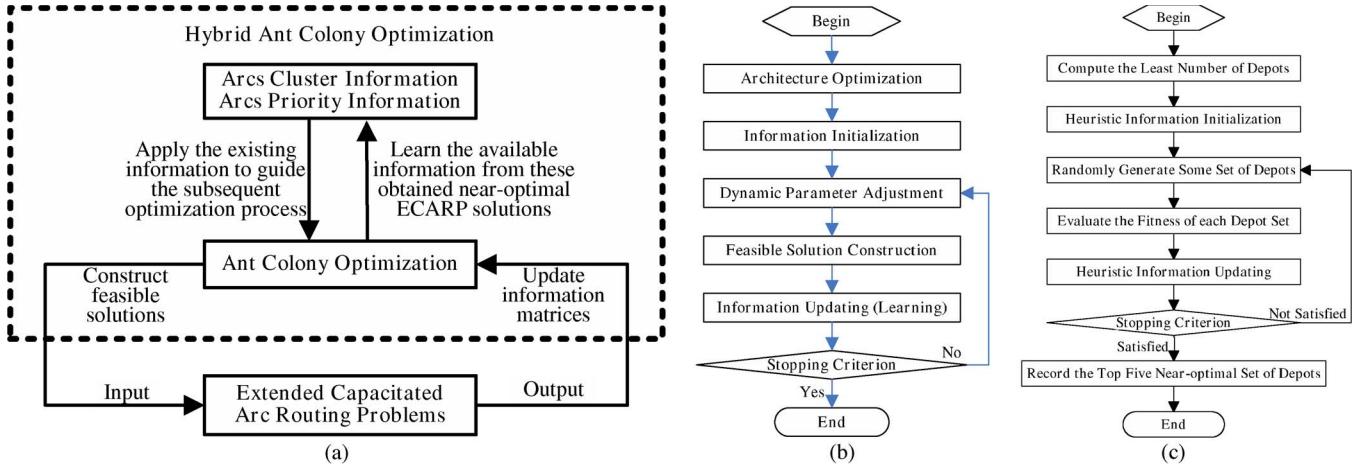


Fig. 1. Workings of HACO. (a) The overall architecture of the algorithm, (b) the computational flow of the algorithm, and (c) the computational flow of the architecture optimization module.

to complex optimization problems in a reasonable amount of computational time [50], [51]. In particular, ACO has been employed successfully to obtain high-quality solutions to the travelling salesman problem which is closely related to the routing of vehicles in the ECARP. There are numerous different variants of ACO as surveyed by Dorigo and Blum [52], [53].

In order to effectively deal with the ECARP, an HACO is proposed. The HACO is divided into two functional modules: a heuristic search module (the ant colony system) and a *knowledge module*. The heuristic search module takes charge of searching through the vast solution space and identifies optimal solutions. The knowledge module obtains useful information throughout the ongoing optimization process and applies this knowledge to guide the subsequent search. The working mechanism of HACO is shown in Fig. 1(a), and the algorithm's computational flow is shown in Fig. 1(b). It is evident from this figure that the algorithm implements five components across the two functional modules. These may be outlined as follows.

- 1) **Architecture optimization:** It obtains near-optimal sets of depots (the number and position of depots are decided in this step).
- 2) **Information initialization:** It initializes the information (e.g., arc cluster or arc priority information).
- 3) **Dynamic parameter adjustment:** It dynamically selects a specific combination of parameters according to their past optimization performances.
- 4) **Feasible solution construction:** It constructs a group of solutions using ACO. This step makes use of the heuristic information obtained from the best solutions found so far.
- 5) **Information updating:** The information is updated using information extracted from the near-optimal solutions obtained.

Steps 3)–5) are repeated iteratively until a stopping criterion is satisfied. In the following, we outline each component in detail.

#### A. Architecture Optimization

The principal goal of optimizing the problem's architecture is to decide the number of depots required and their positions. Its computational flow is described in detail next.

*Compute the Least Number of Depots:* Since all required arcs should be serviced within the maximal service time  $T$  and vehicle routes must start and end at the same depot, the least number of depots is decided by the maximum travel distance of trucks,  $\beta = S \times T$ . The distance between node  $i$  ( $1 \leq i \leq N_V$ ) and the depots can be defined as

$$dis(i) = \min_{1 \leq j \leq N_V, d_j=1} SPL(i, j) \quad (13)$$

where  $d_j = 1$  denotes that node  $j$  is selected as a depot and  $SPL(i, j)$  denotes the deadheading cost of the shortest path from node  $i$  to node  $j$  in  $G_D$ . The least number of depots  $N^*$  is determined such that  $dis(i) \leq \beta/2$ ,  $1 \leq i \leq N_V$ .

*Heuristic Information Initialization:* The depot selection information, used to improve the quality of the architecture optimization, is the accumulative knowledge (refined iteratively) of assigning a given node to be a depot. An array  $DSI$  with size  $N_V$  is used for the depot selection information, where  $DSI(i)$  denotes the number of times node  $i$  was designated as a depot among the near-optimal solutions obtained. The more frequently node  $i$  has been assigned to be a depot, the larger  $DSI(i)$  will be, and the probability of assigning node  $i$  as a depot will subsequently be larger too. At the beginning of each run,  $DSI(i)$ ,  $1 \leq i \leq N_V$ , is initialized as one.

*Randomly Generate Some Set of Depots:* A set of depots may be considered as an individual in the algorithm's population. The length  $Len$  of each individual is determined by

$$Len = \begin{cases} N^*, & 0 \leq r < 0.5 \\ N^* + R_P, & 0.5 \leq r < 1 \end{cases} \quad (14)$$

where  $N^*$  denotes the least number of depots,  $R_P \in [1, N^*]$  is a random integer drawn from a uniform distribution, and  $r \in [0, 1)$  is a random real number, again drawn from a uniform distribution. Each node  $i$  will be selected as a depot with the following probability:

$$P(i) = \frac{DSI(i)}{\sum_{k=1}^{N_V} DSI(k)}. \quad (15)$$

TABLE II  
EXAMPLE OF ARC CLUSTER INFORMATION MATRIX

	Arc 1	Arc 2	Arc 3	Arc 4	Arc 5	Arc 6
Arc 1	10	6	2	1	1	0
Arc 2	6	10	1	1	1	1
Arc 3	2	1	10	3	3	1
Arc 4	1	1	3	10	2	3
Arc 5	1	1	3	2	10	3
Arc 6	0	1	1	3	3	10

*Evaluate the Fitness of Each Depot Set:* The fitness of each set of depots is computed using the Extended Random Path-Scanning (ERPS) heuristic [40]. When constructing a tour, ERPS appends the most promising arc to the existing sequence of arcs until the capacity  $Q$  or service time  $T$  is exhausted.

*Heuristic Information Updating:* If the best solution found so far is improved at the end of an iteration, then the depot selection information will be updated as follows:

$$DSI(i) = \begin{cases} DSI(i) + 1, & i \in GBS \\ DSI(i), & \text{otherwise} \end{cases} \quad (16)$$

where  $i \in GBS$  denotes that node  $i$  is selected as a depot.

*Stopping Criterion:* The execution of the architecture optimization process will be terminated when the best solution found does not improve over ten consecutive iterations.

### B. Information Initialization

In ACO, the artificial ants will deposit pheromone on the paths traversed. Similar to the pheromone definition described earlier in this paper, this extends the concept of pheromone deposition, making use of arc cluster information and arc priority information. In HACO, the artificial ants will construct feasible solutions based on this information, reinforcing (via deposition) the information along the traveled paths.

*Arc Cluster Information:* This information used to assign required arcs to existing clusters is extracted from the best solutions found throughout the search. A matrix  $ACI$  with size  $N_R \times N_R$  is defined to contain this information, where  $ACI(i, j)$  denotes the probability of assigning arc  $i$  and arc  $j$  to the same cluster;  $N_R$  denotes the number of required arcs. An example of an  $ACI$  matrix is shown in Table II, where  $ACI(1, 2) = 6$  indicates that arcs 1 and 2 can be assigned to the same cluster with 60% ( $6/6 + 2 + 1 + 1$ ) probability, while  $ACI(1, 3) = 2$  means that arcs 1 and 3 can be assigned to the same cluster with 20% ( $2/6 + 2 + 1 + 1$ ) probability. In the initialization phase, each element of  $ACI$  is initialized as  $\tau_0$ .

*Arc Priority Information:* This information used to establish an appropriate processing priority for the different arcs is stored in a matrix  $API$  with size  $N_R \times N_R$ , where  $API(i, j)$  denotes the probability of servicing arc  $i$  before arc  $j$ . An example of an  $API$  matrix is shown in Table III, where  $API(1, 2) = 5$  indicates that the probability of servicing arc 1 before arc 2 is 50% ( $5/5 + 2 + 1 + 1 + 1$ ), while  $API(1, 3) = 2$  indicates that the probability of servicing arc 1 before arc 3 is 20% ( $2/5 + 2 + 1 + 1 + 1$ ). At the beginning, each element of  $API$  is initialized as  $\tau_0$ .

TABLE III  
EXAMPLE OF ARC PRIORITY INFORMATION MATRIX

	Arc 1	Arc 2	Arc 3	Arc 4	Arc 5	Arc 6
Arc 1	1	5	2	1	1	1
Arc 2	5	1	1	1	1	1
Arc 3	2	1	1	3	3	1
Arc 4	1	1	3	1	2	3
Arc 5	1	1	3	2	1	3
Arc 6	1	1	1	3	3	1

TABLE IV  
EXAMPLE OF DYNAMIC PARAMETER ADJUSTMENT

Parameter Combination	First Iteration		Second Iteration	
	Optimisation Performance	Probability of Selection	Optimisation Performance	Probability of Selection
ParaComb 1	1	25%	1	20%
ParaComb 2	1	25%	2	40%
ParaComb 3	1	25%	1	20%
ParaComb 4	1	25%	1	20%

### C. Dynamic Parameter Adjustment

In order to improve the overall performance of HACO and decrease the impact of initial parameter choices on the results obtained, parameters are dynamically adjusted according to the performance of different parameter combinations. If the best solution found so far improves after an iteration, then this iteration will be called a successful iteration. The number of successful iterations under a given parameter combination can be regarded as its optimization performance. In the initialization phase, many parameter combinations are generated following the orthogonal design [57], and the optimization performance of each parameter combination is initialized as one. In the beginning of each iteration, one parameter combination will be selected randomly according to its optimization performance. An example of dynamic parameter adjustment is displayed in Table IV.

As indicated in Table IV, each parameter combination is selected with 25% probability at the beginning. Suppose that the second parameter combination (ParaComb 2) is selected for the first iteration and the best solution found is improved after the first iteration. Then, the optimization performance of ParaComb 2 will be increased by one. At the beginning of the second iteration, the selection probability of ParaComb 2 is 40%, while the selection probability of the other parameter combinations is 20%.

### D. Feasible Solution Construction

In this phase, solutions are constructed with guidance from the heuristic information described earlier.

*Allowed Selecting Arc Set:* This set determines which of the required arcs (i.e., those that require service) should be used to construct the current solution. Let  $B(i) \in \{0, 1\}$  be the service indicator of arc  $i$  such that  $B(i) = 1$  means that arc  $i$  has been serviced.  $Allow(\mu, B)$  then denotes the allowed selecting arc set after servicing arc  $\mu$ . The pseudocode for the entire set construction is shown in Fig. 2.

*State Transition Rule:* This rule specifies how to select the next required arc from the neighborhood of the current required

Step 1.	Initialize the local best solution and set its objective (summation of fixed costs and running costs) as infinite.
Step 2.	Construct the vehicle routes of current ant (current solution) with the guidance of obtained heuristic information.
Step 2.1.	Select the depot of current route (trip). The depot closest to any required arcs which has not yet serviced is selected as the depot of current route. If there are many satisfied depots, then select one randomly.
Step 2.2.	Determine the <b>Allowed Selecting Arc Set</b> .
Step 2.3.	Select the next required arc from the <b>Allowed Selecting Arc Set</b> using the <b>State Transition Rule</b> .
Step 2.4.	Join the selected arc of Step 2.3 to current route.
Step 2.5.	If both capacity and maximal service time are not exhausted, go to Step 2.2.
Step 2.6.	If some required arcs are still not serviced, go to Step 2.1.
Step 3.	If the current solution is better than the local best solution, then update the local best solution.
Step 4.	If not all of ants have finished constructing the current solution, then go to Step 2.
Step 5.	Improve the local best solution using the local optimisation methods ( <b>Local Optimisation on One Trip</b> and <b>Local Optimisation on Two Trips</b> ).
Step 6.	If the local best solution is better than the global best solution, then update the global best solution.

Fig. 2. Pseudocode of feasible solution construction.

arc. In HACO, each ant makes use of the probability distribution [see (17)] that follows from the allowed selecting arc set and randomly selects the next arc  $\mu_j$  (when finished with the current arc  $\mu_i$ ) according to

$$\Pr(\mu_i, \mu_j, B) = \begin{cases} \frac{TF(\mu_i, \mu_j)}{\sum_{\mu_k \in Allow(\mu_i, B)} TF(\mu_i, \mu_k)}, & \mu_j \in Allow(\mu_i, B) \\ 0, & \mu_j \notin Allow(\mu_i, B) \end{cases} \quad (17)$$

where

$$TF(\mu_i, \mu_j) = [TFA(\mu_i, \mu_j)]^a \times [TFB(\mu_i, \mu_j)]^b \times [TFC(\mu_i, \mu_j)]^c. \quad (18)$$

$TFA(\mu_i, \mu_j)$  denotes the normalized result of the static heuristic information (e.g., the demand of arcs) and

$$TFA(\mu_i, \mu_j) = \frac{DI(\mu_i, \mu_j)}{\max_{\mu_k \in Allow(\mu_i, B)} \{DI(\mu_i, \mu_k)\}} \quad (19)$$

$TFB(\mu_i, \mu_j)$  denotes the normalized result of the arc cluster information (accumulative) and

$$TFB(\mu_i, \mu_j) = \frac{ACI(\mu_i, \mu_j)}{\max_{\mu_k \in Allow(\mu_i, B)} \{ACI(\mu_i, \mu_k)\}} \quad (20)$$

$TFC(\mu_i, \mu_j)$  denotes the normalized result of the arc priority information (accumulative) and

$$TFC(\mu_i, \mu_j) = \frac{API(\mu_i, \mu_j)}{\max_{\mu_k \in Allow(\mu_i, B)} \{API(\mu_i, \mu_k)\}} \quad (21)$$

and  $a, b, c$  denote the weights of the different heuristic information.

The static heuristic information is characterized by

$$DI(\mu_i, \mu_j) = \begin{cases} q(\mu_j), & 0 \leq r < \frac{1}{6} \\ \frac{1}{q(\mu_j)}, & \frac{1}{6} \leq r < \frac{1}{3} \\ d(\mu_j) + s(\mu_j) - pen(\mu_i, \mu_j), & \frac{1}{3} \leq r < \frac{1}{2} \\ \frac{1}{d(\mu_j) + s(\mu_j) + pen(\mu_i, \mu_j)}, & \frac{1}{2} \leq r < \frac{2}{3} \\ \frac{q(\mu_j)}{d(\mu_j) + s(\mu_j) + pen(\mu_i, \mu_j)}, & \frac{2}{3} \leq r < \frac{5}{6} \\ \frac{d(\mu_j) + s(\mu_j) - pen(\mu_i, \mu_j)}{q(\mu_j)}, & \frac{5}{6} \leq r \leq 1 \end{cases} \quad (22)$$

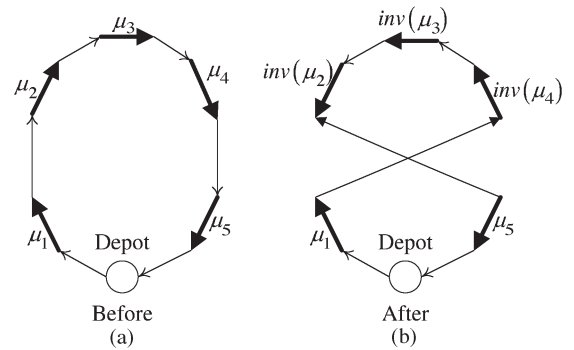


Fig. 3. Local optimization on one trip using two-opt. Here, thick lines denote the required arcs, while thin lines denote the shortest deadheading paths.

where  $r \in [0, 1]$  denotes a uniformly random float. In (22), the first branch denotes that the arc with the largest demand should be serviced with the top priority, while the second one means that the arc with the smallest demand should be serviced with the top priority. The third branch denotes that the arc with the highest cost should be serviced with the top priority, while the fourth one means that the arc with the cheapest cost should be serviced with the top priority. The fifth branch denotes that the arc with the largest yield should be serviced with the top priority, while the fourth one means that the arc with the small yield should be serviced with the top priority. These rules are all learned from the Path-Scanning [18] heuristic and are very effective to tackle the ECARP. In (18), both the intrinsic information (e.g., the demand of each arc) and the obtained information (i.e., arc cluster information) are integrated into the heuristic information.

*Local Optimization:* HACO is hybridized with the two-opt heuristic [10], [58] to further improve the performance. At the end of each iteration, the best solution obtained from this iteration is improved via the two-opt. There are two phases: *local optimization on one trip* (Fig. 3) and *local optimization on two trips* (Fig. 4). The local search implements these two phases which scan all possible pairs of required arcs. Each phase is terminated as soon as an improvement has been found. If arc  $\mu$  is a bidirectional arc, then  $inv(\mu)$  denotes the opposite arc corresponding to arc  $\mu$ ; if arc  $\mu$  is a single-direction arc, then  $inv(\mu) = \mu$ .

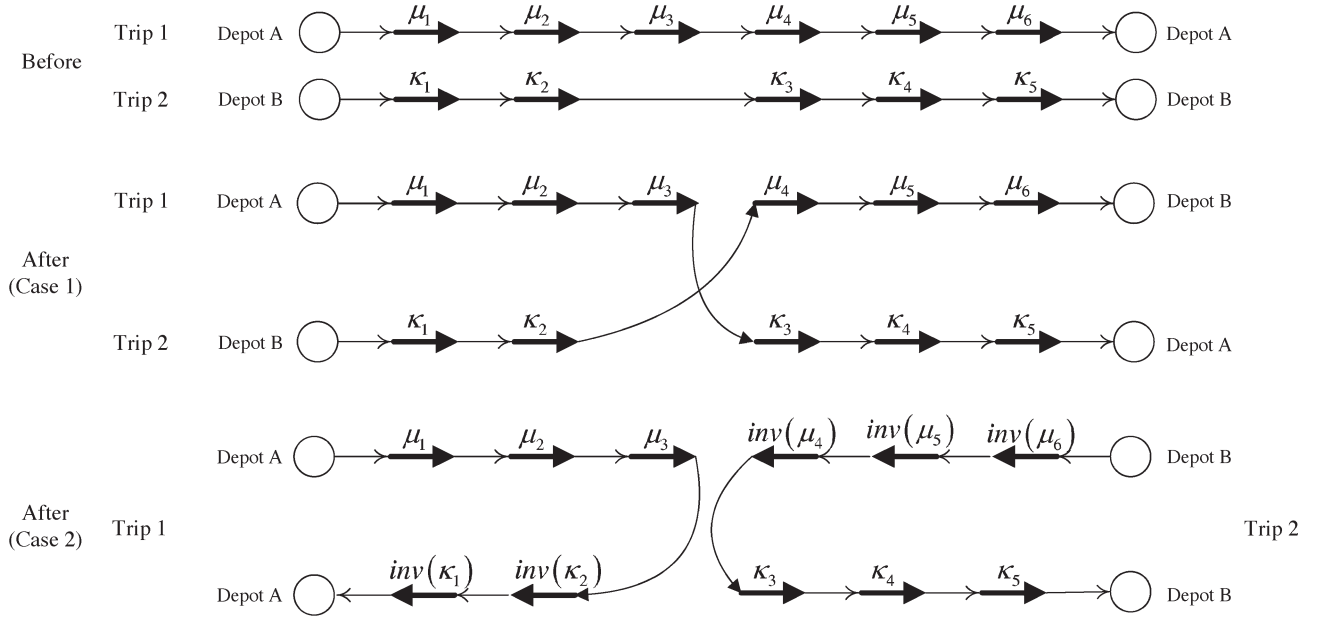


Fig. 4. Local optimization on two trips using two-opt. Here, thick lines denote the required arcs, while thin lines denote the shortest deadheading paths.

### E. Information Updating

In ACO, the amount of deposited pheromones will be updated according to different rules. Likewise, the accumulated heuristic information in HACO will also be updated based on different rules.

**Global Updating Rule:** When the best solution found so far improves, the information amount is updated via the global updating rule of (23) and (24). In HACO, only the ant which constructed the global best solution (the best solution found so far) is allowed to deposit information employing the global updating rule

$$ACI(i, j) = \begin{cases} ACI(i, j) + Q_G, & \text{if } i, j \in GBS_T \\ ACI(i, j), & \text{Other} \end{cases} \quad (23)$$

$$API(i, j) = \begin{cases} API(i, j) + Q_G, & \text{if } (i, j \in GBS_T) \text{ and } (i \rightarrow j) \\ API(i, j), & \text{Other} \end{cases} \quad (24)$$

where  $i, j \in GBS_T$  denotes that arcs  $i$  and  $j$  are serviced in the same tour (corresponding to the best solution found so far). Furthermore,  $i \rightarrow j$  denotes that arc  $i$  and arc  $j$  are serviced in succession, and  $Q_G$  denotes the incremental amount at the global updating phase.

**Local Updating Rule:** This rule is executed after each iteration. Only the ant which constructed the local best solution (i.e., the best solution of each iteration) is allowed to deposit information by the local updating rule

$$ACI(i, j) = \begin{cases} ACI(i, j) + Q_L, & \text{if } i, j \in LBS_T \\ ACI(i, j), & \text{otherwise} \end{cases} \quad (25)$$

$$API(i, j) = \begin{cases} API(i, j) + Q_L, & \text{if } (i, j \in LBS_T) \text{ and } (i \rightarrow j) \\ API(i, j), & \text{otherwise} \end{cases} \quad (26)$$

where  $i, j \in LBS_T$  denotes that arcs  $i$  and  $j$  are serviced in the same tour (corresponding to the local best solution),  $i \rightarrow j$  denotes that arc  $i$  and arc  $j$  are serviced in succession, and  $Q_L$  denotes the incremental amount at the local updating phase.

**Information Evaporating Rule:** This rule is employed following each iteration. The information amount is updated by the information evaporating rule [see (27) and (28)]. In HACO, the information amount of each solution component is limited to an interval  $[\tau_{\min}, \tau_{\max}]$  to avoid stagnation

$$ACI(i, j) = \max\{\tau_{\min}, \min\{\tau_{\max}, (1 - \rho)ACI(i, j)\}\} \quad (27)$$

$$API(i, j) = \max\{\tau_{\min}, \min\{\tau_{\max}, (1 - \rho)API(i, j)\}\} \quad (28)$$

where  $(0 < \rho < 1)$  is the information evaporating parameter.

## V. EXPERIMENTAL RESULTS

In order to test the validity of HACO, it has been evaluated on four different sets of ECARP problem containing a total of 87 instances with up to 140 nodes and 380 arcs. As the global optimum of these ECARP instances is unknown, a lower bound is used instead. The relative performance of HACO is established by a comparison to the following five approaches.

- 1) Two classical heuristics for the CARP, Path-Scanning [18] and Ulusoy's heuristic [20], were extended by Xing *et al.* to make them applicable to the ECARP. The resulting extended heuristics, ERPS and Extended Random Ulusoy's Heuristic (ERUH), have been shown to perform well on an extended version of the traditional CARP [40].
- 2) The MA [10] by Lacomme *et al.* outperformed all known heuristics on three sets of benchmark problems.

TABLE V  
PARAMETER SYMBOLS USED IN THIS PAPER

Symbol	Meaning
<i>AntSize</i>	The size of artificial ants
$\tau_0$	The initial level of heuristic information
$a$	The weight of determinate information
$b$	The weight of Arc Cluster Information
$c$	The weight of Arc Priority Information
$\tau_{min}$	The minimal level of heuristic information
$\tau_{max}$	The maximal level of heuristic information
$Q_L$	The incremental level at the global updating phase
$Q_G$	The incremental level at the local updating phase
$\rho$	The information evaporating parameter

- 3) HHEA [40] hybridizes evolutionary algorithms with heuristic information. Experimental results suggest that the integrated evolutionary framework significantly outperforms individual extended heuristics.
- 4) Since the approach presented in this paper is an ant colony algorithm, it is also necessary to compare it with a typical ACO algorithm. For this reason, we also consider the Max-Min Ant System (MMAS) [55].

Each of these techniques is evaluated on 87 ECARP instances. These instances were generated<sup>2</sup> based on the following sets of readily available benchmark problems:

- 1) GDB: It contains 23 undirected instances presented in [28], with 7–27 nodes and 11–55 edges. For each instance, all edges are required.
- 2) KESH: It contains six undirected instances proposed by Kiuchi *et al.* [59], with 6–10 nodes and 15 edges. For each instance, all edges are required.
- 3) VAL: It contains 34 undirected instances designed in [60]. These instances have 24–50 nodes and 34–97 edges. For each instance, all edges are required.
- 4) EGLESE: It contains 24 undirected instances constructed by Belenguer and Benavent [60]. Generally, all instances are very interesting because of their authenticity and large size (77–140 nodes and 98–190 edges).

#### A. Parameter Settings

HACO A was implemented in Visual C++, and all codes were executed on a personal computer with an Intel Pentium 2-GHz processor and 2-GB RAM. The parameters required are elaborated in Table V, and the parameter combinations investigated are summarized in Table VI. Among these parameter combinations, the following parameters are predefined as follows:  $AntSize = 50$ ,  $\tau_{min} = 0.1$ ,  $\tau_{max} = 10$ , and  $\tau_0 = 1$ . The final experimental results were averaged over 50 trials. The Wilcoxon signed ranks test [61] is employed for the statistical comparisons of the different approaches.

Each algorithm will be terminated when the predefined number of fitness evaluations is exhausted. This limit, set to 5000 function evaluations, is the same for all algorithms to ensure a fair comparison. The average computational error is computed as follows:

$$E_A = \frac{F_A - LB}{LB} \quad (29)$$

<sup>2</sup>The original CARP instances may be downloaded from <http://www.uv.es/belengue/carp.html>.

TABLE VI  
PARAMETER COMBINATIONS USED IN THIS PAPER

Parameter	$a$	$b$	$c$	$Q_L$	$Q_G$	$\rho$
Para-Comb 1	1	1	1	0.01	0.15	0.03
Para-Comb 2	1	1	1	0.01	0.2	0.02
Para-Comb 3	1	1	1	0.01	0.25	0.01
Para-Comb 4	1	3	3	0.03	0.15	0.03
Para-Comb 5	1	3	3	0.03	0.2	0.02
Para-Comb 6	1	3	3	0.03	0.25	0.01
Para-Comb 7	1	5	5	0.05	0.15	0.03
Para-Comb 8	1	5	5	0.05	0.2	0.02
Para-Comb 9	1	5	5	0.05	0.25	0.01
Para-Comb 10	3	1	3	0.05	0.15	0.02
Para-Comb 11	3	1	3	0.05	0.2	0.01
Para-Comb 12	3	1	3	0.05	0.25	0.03
Para-Comb 13	3	3	5	0.01	0.15	0.02
Para-Comb 14	3	3	5	0.01	0.2	0.01
Para-Comb 15	3	3	5	0.01	0.25	0.03
Para-Comb 16	3	5	1	0.03	0.15	0.02
Para-Comb 17	3	5	1	0.03	0.2	0.01
Para-Comb 18	3	5	1	0.03	0.25	0.03
Para-Comb 19	5	1	5	0.03	0.15	0.01
Para-Comb 20	5	1	5	0.03	0.2	0.03
Para-Comb 21	5	1	5	0.03	0.25	0.02
Para-Comb 22	5	3	1	0.05	0.15	0.01
Para-Comb 23	5	3	1	0.05	0.2	0.03
Para-Comb 24	5	3	1	0.05	0.25	0.02
Para-Comb 25	5	5	3	0.01	0.15	0.01
Para-Comb 26	5	5	3	0.01	0.2	0.03
Para-Comb 27	5	5	3	0.01	0.25	0.02

TABLE VII  
FOUR VERSIONS OF HACO AS

Version	Heuristic Information	Parameters	Local Optimisation
HACO A-1	ACI	Fixed	No
HACO A-2	ACI+API	Fixed	No
HACO A-3	ACI+API	Dynamic	No
HACO A-4	ACI+API	Dynamic	Yes

where  $E_A$  denotes the average computational error,  $F_A$  denotes the average experimental result of the given instance, and  $LB$  denotes the lower bound of the given instance.

#### B. Performances of Different Components

HACO A incorporates several different components to deal with the complexity of the ECARP. In this section, four different versions of HACO A are tested in order to evaluate the impact of these components on the algorithm's overall performance. The principle goal of this experiment is to gain a better understanding of how and why HACO A works. The configuration of each variant is summarized in Table VII: "ACI" denotes that only arc cluster information is applied in the optimization process, while "ACI + API" denotes that both arc cluster information and arc priority information are used throughout the optimization process. "Fixed parameters" denote the optimal parameter combination among the parameter combinations shown in Table VI (this was precomputed for the different instances). Nine different ECARP instances were applied to validate the performance of different HACO As, and the average computational error of each variant is listed in Table VIII.

In Table VIII, numbers in bold indicate that the performance of the method to the right is significantly better (confidence degree of 0.95) than the one nearest to the left. The experimental results demonstrate that HACO A-2 consistently outperforms

TABLE VIII  
AVERAGE COMPUTATIONAL ERROR (%) OF HACO A APPROACHES

<i>SN</i>	<i>HACO A</i> <sub>1</sub>	<i>HACO A</i> <sub>2</sub>	<i>HACO A</i> <sub>3</sub>	<i>HACO A</i> <sub>4</sub>
C79	4.129	<b>3.628</b>	<b>2.817</b>	<b>1.807</b>
C80	4.257	<b>3.893</b>	<b>2.985</b>	<b>2.208</b>
C81	4.267	<b>3.967</b>	<b>3.451</b>	<b>2.519</b>
C82	2.902	2.896	<b>2.172</b>	<b>1.195</b>
C83	3.979	<b>3.506</b>	<b>2.738</b>	<b>1.689</b>
C84	4.525	<b>4.213</b>	<b>3.419</b>	<b>2.302</b>
C85	3.792	<b>3.319</b>	<b>2.632</b>	<b>1.572</b>
C86	4.503	<b>4.118</b>	<b>3.382</b>	<b>2.290</b>
C87	4.487	<b>4.205</b>	<b>3.461</b>	<b>2.272</b>

TABLE IX  
AVERAGE COMPUTATIONAL ERROR (%) OF ERPS AND HACO A

<i>SN</i>	<i>ERPS</i>	<i>HACO A</i>	<i>SN</i>	<i>ERPS</i>	<i>HACO A</i>
C1	1.130	<b>0.254</b>	C45	1.534	<b>0.354</b>
C2	0.087	0.077	C46	2.526	<b>0.631</b>
C3	0.727	<b>0.177</b>	C47	1.490	<b>0.354</b>
C4	1.228	<b>0.308</b>	C48	1.269	<b>0.323</b>
C5	0.072	0.062	C49	2.492	<b>0.655</b>
C6	0.056	0.046	C50	1.564	<b>0.424</b>
C7	1.856	<b>0.469</b>	C51	1.677	<b>0.454</b>
C8	2.653	<b>0.754</b>	C52	4.023	<b>0.962</b>
C9	4.287	<b>0.992</b>	C53	1.669	<b>0.439</b>
C10	0.426	<b>0.108</b>	C54	2.239	<b>0.554</b>
C11	1.457	<b>0.354</b>	C55	3.990	<b>1.093</b>
C12	1.223	<b>0.285</b>	C56	0.584	<b>0.139</b>
C13	0.615	<b>0.139</b>	C57	1.734	<b>0.447</b>
C14	0.653	<b>0.154</b>	C58	1.714	<b>0.393</b>
C15	0.054	0.069	C59	1.441	<b>0.331</b>
C16	1.082	<b>0.277</b>	C60	1.421	<b>0.347</b>
C17	0.017	0.015	C61	0.983	<b>0.246</b>
C18	0.716	<b>0.162</b>	C62	0.744	<b>0.169</b>
C19	0.092	0.085	C63	2.061	<b>0.477</b>
C20	1.555	<b>0.354</b>	C64	6.222	<b>1.501</b>
C21	0.064	0.046	C65	6.913	<b>1.601</b>
C22	0.097	0.085	C66	4.506	<b>1.083</b>
C23	0.927	<b>0.215</b>	C67	5.882	<b>1.531</b>
C24	1.271	<b>0.362</b>	C68	4.282	<b>1.130</b>
C25	0.811	<b>0.223</b>	C69	8.977	<b>2.337</b>
C26	0.970	<b>0.262</b>	C70	6.580	<b>1.631</b>
C27	3.444	<b>0.931</b>	C71	5.225	<b>1.236</b>
C28	3.915	<b>0.954</b>	C72	6.704	<b>1.760</b>
C29	2.844	<b>0.754</b>	C73	8.096	<b>1.866</b>
C30	0.969	<b>0.262</b>	C74	10.443	<b>2.567</b>
C31	3.111	<b>0.885</b>	C75	13.306	<b>3.438</b>
C32	3.758	<b>0.885</b>	C76	3.909	<b>0.930</b>
C33	0.667	<b>0.169</b>	C77	5.216	<b>1.289</b>
C34	1.024	<b>0.231</b>	C78	6.059	<b>1.536</b>
C35	6.536	<b>1.648</b>	C79	7.580	<b>1.807</b>
C36	0.784	<b>0.200</b>	C80	9.100	<b>2.208</b>
C37	0.969	<b>0.223</b>	C81	10.819	<b>2.519</b>
C38	6.352	<b>1.578</b>	C82	5.326	<b>1.195</b>
C39	1.426	<b>0.385</b>	C83	6.794	<b>1.689</b>
C40	1.026	<b>0.246</b>	C84	10.083	<b>2.302</b>
C41	1.805	<b>0.416</b>	C85	5.774	<b>1.572</b>
C42	1.327	<b>0.377</b>	C86	10.259	<b>2.290</b>
C43	1.514	<b>0.362</b>	C87	8.569	<b>2.272</b>
C44	1.048	<b>0.270</b>			

HACO A-1, suggesting that employing two kinds of heuristic information is more powerful than employing just one. The experimental results also demonstrate that HACO A-3 is significantly better than HACO A-2, leading to the conclusion that the dynamic parameter technique outperforms the fixed parameter one. Finally, the last set of experimental results demonstrates that HACO A-4 is significantly better than HACO A-3. This suggests that local optimization can largely improve the performance of the HACO A framework. Overall, these results highlight the following: 1) usefulness of the individual components and 2) their combined contribution to an overall improvement in performance of the algorithm.

TABLE X  
AVERAGE COMPUTATIONAL ERROR (%) OF ERUH AND HACO A

<i>SN</i>	<i>ERUH</i>	<i>HACO A</i>	<i>SN</i>	<i>ERUH</i>	<i>HACO A</i>
C1	1.080	<b>0.254</b>	C45	1.644	<b>0.354</b>
C2	0.075	0.077	C46	2.726	<b>0.631</b>
C3	0.839	<b>0.177</b>	C47	1.756	<b>0.354</b>
C4	1.274	<b>0.308</b>	C48	1.527	<b>0.323</b>
C5	0.089	0.062	C49	2.890	<b>0.655</b>
C6	0.055	0.046	C50	2.012	<b>0.424</b>
C7	1.969	<b>0.469</b>	C51	1.938	<b>0.454</b>
C8	3.241	<b>0.754</b>	C52	4.271	<b>0.962</b>
C9	4.624	<b>0.992</b>	C53	2.166	<b>0.439</b>
C10	0.463	<b>0.108</b>	C54	2.595	<b>0.554</b>
C11	1.582	<b>0.354</b>	C55	4.604	<b>1.093</b>
C12	1.159	<b>0.285</b>	C56	0.673	<b>0.139</b>
C13	0.693	<b>0.139</b>	C57	2.069	<b>0.447</b>
C14	0.706	<b>0.154</b>	C58	1.625	<b>0.393</b>
C15	0.065	0.069	C59	1.393	<b>0.331</b>
C16	1.251	<b>0.277</b>	C60	1.599	<b>0.347</b>
C17	0.015	0.015	C61	1.139	<b>0.246</b>
C18	0.718	<b>0.162</b>	C62	0.739	<b>0.169</b>
C19	0.089	0.085	C63	2.182	<b>0.477</b>
C20	1.621	<b>0.354</b>	C64	6.682	<b>1.501</b>
C21	0.081	0.046	C65	6.474	<b>1.601</b>
C22	0.086	0.085	C66	4.361	<b>1.083</b>
C23	0.998	<b>0.215</b>	C67	6.603	<b>1.531</b>
C24	1.524	<b>0.362</b>	C68	4.535	<b>1.130</b>
C25	0.977	<b>0.223</b>	C69	10.245	<b>2.337</b>
C26	1.253	<b>0.262</b>	C70	7.638	<b>1.631</b>
C27	4.358	<b>0.931</b>	C71	5.059	<b>1.236</b>
C28	4.256	<b>0.954</b>	C72	7.102	<b>1.760</b>
C29	3.444	<b>0.754</b>	C73	8.607	<b>1.866</b>
C30	1.256	<b>0.262</b>	C74	11.830	<b>2.567</b>
C31	3.592	<b>0.885</b>	C75	13.806	<b>3.438</b>
C32	4.074	<b>0.885</b>	C76	3.735	<b>0.930</b>
C33	0.685	<b>0.169</b>	C77	5.401	<b>1.289</b>
C34	1.020	<b>0.231</b>	C78	7.046	<b>1.536</b>
C35	7.095	<b>1.648</b>	C79	7.332	<b>1.807</b>
C36	0.975	<b>0.200</b>	C80	9.644	<b>2.208</b>
C37	0.895	<b>0.223</b>	C81	11.667	<b>2.519</b>
C38	7.524	<b>1.578</b>	C82	5.638	<b>1.195</b>
C39	1.914	<b>0.385</b>	C83	7.926	<b>1.689</b>
C40	1.228	<b>0.246</b>	C84	9.402	<b>2.302</b>
C41	1.992	<b>0.416</b>	C85	7.002	<b>1.572</b>
C42	1.673	<b>0.377</b>	C86	10.172	<b>2.290</b>
C43	1.628	<b>0.362</b>	C87	9.891	<b>2.272</b>
C44	1.138	<b>0.270</b>			

In the following sections, HACO A is compared with ERPS, ERUH, MA, HHEA, and MMAS. The average computational errors of these methods are summarized in Tables IX and XIII. In these tables, numbers in bold indicate that the performance of the algorithm to the right is significantly better (confidence degree of 0.95) than the one to the left.

### C. HACO A Versus ERPS and ERUH

According to the experimental results summarized in Tables IX and X, the performance of HACO A is significantly better than that of either ERPS or ERUH for almost all instances considered. It should be noted that both ERPS and ERUH are simple heuristics (i.e., stochastic algorithms with simple rules), whereas HACO A is an advanced metaheuristic. It is therefore expected that HACO A performs superior, and this is indeed demonstrated by the experimental results.

### D. HACO A Versus MA

Lacomme's MA cannot be employed directly to solve the ECARP, and we therefore introduced some required extensions. Based on the experimental results in Table XI, it may be observed that HACO A is significantly better in the majority of cases. We believe the reasons for this discrepancy in

TABLE XI  
AVERAGE COMPUTATIONAL ERROR (%) OF MA AND HACO A

<i>SN</i>	<i>MA</i>	<i>HACO A</i>	<i>SN</i>	<i>MA</i>	<i>HACO A</i>
C1	0.869	<b>0.254</b>	C45	1.127	<b>0.354</b>
C2	0.085	0.077	C46	1.276	<b>0.631</b>
C3	0.639	<b>0.177</b>	C47	1.115	<b>0.354</b>
C4	0.874	<b>0.308</b>	C48	0.928	<b>0.323</b>
C5	0.069	0.062	C49	1.895	<b>0.655</b>
C6	0.045	0.046	C50	1.131	<b>0.424</b>
C7	0.983	<b>0.469</b>	C51	1.183	<b>0.454</b>
C8	1.546	<b>0.754</b>	C52	2.127	<b>0.962</b>
C9	2.522	<b>0.992</b>	C53	1.166	<b>0.439</b>
C10	0.319	<b>0.108</b>	C54	1.559	<b>0.554</b>
C11	0.823	<b>0.354</b>	C55	2.046	<b>1.093</b>
C12	1.002	<b>0.285</b>	C56	0.537	<b>0.139</b>
C13	0.594	<b>0.139</b>	C57	1.013	<b>0.447</b>
C14	0.403	<b>0.154</b>	C58	1.016	<b>0.393</b>
C15	0.073	0.069	C59	0.784	<b>0.331</b>
C16	0.651	<b>0.277</b>	C60	0.971	<b>0.347</b>
C17	0.015	0.015	C61	0.872	<b>0.246</b>
C18	0.552	<b>0.162</b>	C62	0.593	<b>0.169</b>
C19	0.087	0.085	C63	1.128	<b>0.477</b>
C20	1.312	<b>0.354</b>	C64	3.268	<b>1.501</b>
C21	0.051	0.046	C65	3.669	<b>1.601</b>
C22	0.088	0.085	C66	2.136	<b>1.083</b>
C23	0.667	<b>0.215</b>	C67	3.695	<b>1.531</b>
C24	0.942	<b>0.362</b>	C68	3.355	<b>1.130</b>
C25	0.683	<b>0.223</b>	C69	6.542	<b>2.337</b>
C26	0.874	<b>0.262</b>	C70	4.863	<b>1.631</b>
C27	2.152	<b>0.931</b>	C71	3.019	<b>1.236</b>
C28	2.233	<b>0.954</b>	C72	4.013	<b>1.760</b>
C29	1.692	<b>0.754</b>	C73	4.763	<b>1.866</b>
C30	0.825	<b>0.262</b>	C74	6.803	<b>2.567</b>
C31	1.582	<b>0.885</b>	C75	8.856	<b>3.438</b>
C32	2.012	<b>0.885</b>	C76	3.138	<b>0.930</b>
C33	0.593	<b>0.169</b>	C77	4.014	<b>1.289</b>
C34	0.823	<b>0.231</b>	C78	5.017	<b>1.536</b>
C35	3.289	<b>1.648</b>	C79	5.248	<b>1.807</b>
C36	0.762	<b>0.200</b>	C80	6.572	<b>2.208</b>
C37	0.659	<b>0.223</b>	C81	7.329	<b>2.519</b>
C38	3.245	<b>1.578</b>	C82	4.362	<b>1.195</b>
C39	1.281	<b>0.385</b>	C83	5.219	<b>1.689</b>
C40	0.992	<b>0.246</b>	C84	6.437	<b>2.302</b>
C41	1.129	<b>0.416</b>	C85	5.249	<b>1.572</b>
C42	1.037	<b>0.377</b>	C86	6.712	<b>2.290</b>
C43	1.086	<b>0.362</b>	C87	5.198	<b>2.272</b>
C44	0.729	<b>0.270</b>			

performance to be as follows: In MA, some solutions were randomly generated in both the initialization and the restart phase. However, in many highly constrained combinatorial optimization problems, it is very difficult to obtain feasible solutions by random generation. In particular, we found that the population initialization can largely affect the convergence speed of the algorithm as well as the quality of the final solution.

#### E. HACO A Versus HHEA

The experimental results in Table XII suggest that there is a small difference between HHEA and HACO A when solving smaller instances (C1–C23) and a large difference for larger instances (C64–C87). This difference is introduced by the different mechanism of HHEA and HACO A. In general, the ACO algorithm can obtain some near-optimal solutions relatively quickly, while the improvement of high-quality solutions requires considerably more time. To the contrary, the EA method is able to improve the quality of solutions continuously but requires numerous iterations to obtain the high-quality solutions. In fact, the limit on the number of fitness

TABLE XII  
AVERAGE COMPUTATIONAL ERROR (%) OF HHEA AND HACO A

<i>SN</i>	<i>HHEA</i>	<i>HACO A</i>	<i>SN</i>	<i>HHEA</i>	<i>HACO A</i>
C1	0.285	0.254	C45	0.710	<b>0.354</b>
C2	0.082	0.077	C46	1.240	<b>0.631</b>
C3	0.164	0.177	C47	0.723	<b>0.354</b>
C4	0.646	<b>0.308</b>	C48	0.789	<b>0.323</b>
C5	0.063	0.062	C49	1.206	<b>0.655</b>
C6	0.049	0.046	C50	0.806	<b>0.424</b>
C7	0.984	<b>0.469</b>	C51	0.821	<b>0.454</b>
C8	1.253	<b>0.754</b>	C52	1.839	<b>0.962</b>
C9	2.311	<b>0.992</b>	C53	0.784	<b>0.439</b>
C10	0.116	0.108	C54	1.049	<b>0.554</b>
C11	0.742	<b>0.354</b>	C55	2.189	<b>1.093</b>
C12	0.295	0.285	C56	0.138	0.139
C13	0.121	0.139	C57	0.807	<b>0.447</b>
C14	0.163	0.154	C58	0.433	0.393
C15	0.072	0.069	C59	0.643	<b>0.331</b>
C16	0.276	0.277	C60	0.682	<b>0.347</b>
C17	0.019	0.015	C61	0.373	0.246
C18	0.164	0.162	C62	0.166	0.169
C19	0.091	0.085	C63	1.061	<b>0.477</b>
C20	0.826	<b>0.354</b>	C64	2.674	<b>1.501</b>
C21	0.075	0.046	C65	2.821	<b>1.601</b>
C22	0.865	0.085	C66	2.392	<b>1.083</b>
C23	0.528	<b>0.215</b>	C67	3.497	<b>1.531</b>
C24	0.396	0.362	C68	2.809	<b>1.130</b>
C25	0.529	<b>0.223</b>	C69	4.612	<b>2.337</b>
C26	0.595	<b>0.262</b>	C70	3.919	<b>1.631</b>
C27	1.810	<b>0.931</b>	C71	2.412	<b>1.236</b>
C28	2.023	<b>0.954</b>	C72	4.056	<b>1.760</b>
C29	1.849	<b>0.754</b>	C73	4.346	<b>1.866</b>
C30	0.561	<b>0.262</b>	C74	4.277	<b>2.567</b>
C31	1.546	<b>0.885</b>	C75	6.511	<b>3.438</b>
C32	1.640	<b>0.885</b>	C76	1.879	<b>0.930</b>
C33	0.185	0.169	C77	2.859	<b>1.289</b>
C34	0.245	0.231	C78	3.178	<b>1.536</b>
C35	3.147	<b>1.648</b>	C79	3.543	<b>1.807</b>
C36	0.221	0.200	C80	4.295	<b>2.208</b>
C37	0.225	0.223	C81	4.000	<b>2.519</b>
C38	2.479	<b>1.578</b>	C82	2.323	<b>1.195</b>
C39	0.699	<b>0.385</b>	C83	3.152	<b>1.689</b>
C40	0.519	<b>0.246</b>	C84	4.149	<b>2.302</b>
C41	0.697	<b>0.416</b>	C85	3.697	<b>1.572</b>
C42	0.800	<b>0.377</b>	C86	5.174	<b>2.290</b>
C43	0.632	<b>0.362</b>	C87	5.566	<b>2.272</b>
C44	0.564	<b>0.270</b>			

evaluations allowed is only 5000. For this reason, there exists a large difference between HHEA and HACO A when solving large problem instances. Indeed, if the maximum number of fitness evaluations is increased to 30 000, the difference between HHEA and HACO A becomes increasingly less.

#### F. HACO A Versus MMAS

The experimental results in Table XIII demonstrate that the performance of HACO A is significantly better than that of MMAS. This is to be expected as HACO A incorporates the same mechanisms as MMAS but also employs heuristic information and dynamic parameter adjustment. For this reason, HACO A should be more powerful than MMAS, which was demonstrated by the final results.

## VI. CONCLUSION

This paper has proposed a novel HACO A to deal with an extended variant of the well-known CARP (abbreviated as ECARP). This approach is primarily characterized by the exploitation of heuristic information, the use of adaptive parameters (by means of a dynamic parameter adjustment technique), and the integration of local optimization techniques. The experimental results across a large number of problem

TABLE XIII  
AVERAGE COMPUTATIONAL ERROR (%) OF MMAS AND HACO A

SN	MMAS	HACO A	SN	MMAS	HACO A
C1	0.296	0.254	C45	0.371	0.354
C2	0.078	0.077	C46	1.008	<b>0.631</b>
C3	0.182	0.177	C47	0.690	<b>0.354</b>
C4	0.559	<b>0.308</b>	C48	0.341	0.323
C5	0.058	0.062	C49	1.237	<b>0.655</b>
C6	0.056	0.046	C50	0.467	0.424
C7	0.799	<b>0.469</b>	C51	0.484	0.454
C8	1.148	<b>0.754</b>	C52	1.187	<b>0.962</b>
C9	1.917	<b>0.992</b>	C53	0.849	<b>0.439</b>
C10	0.105	0.108	C54	0.589	0.554
C11	0.413	0.354	C55	1.382	<b>1.093</b>
C12	0.315	0.285	C56	0.213	0.139
C13	0.143	0.139	C57	0.517	0.447
C14	0.203	0.154	C58	0.388	0.393
C15	0.074	0.069	C59	0.415	0.331
C16	0.291	0.277	C60	0.580	<b>0.347</b>
C17	0.016	0.015	C61	0.287	0.246
C18	0.205	0.162	C62	0.198	0.169
C19	0.089	0.085	C63	0.795	<b>0.477</b>
C20	0.443	0.354	C64	1.698	<b>1.501</b>
C21	0.056	0.046	C65	1.754	<b>1.601</b>
C22	0.096	0.085	C66	1.099	1.083
C23	0.417	<b>0.215</b>	C67	1.972	<b>1.531</b>
C24	0.417	0.362	C68	2.053	<b>1.130</b>
C25	0.412	<b>0.223</b>	C69	4.640	<b>2.337</b>
C26	0.317	0.262	C70	1.659	1.631
C27	1.355	<b>0.931</b>	C71	2.249	<b>1.236</b>
C28	1.031	0.954	C72	2.853	<b>1.760</b>
C29	1.396	<b>0.754</b>	C73	2.911	<b>1.866</b>
C30	0.309	0.262	C74	3.193	<b>2.567</b>
C31	1.168	<b>0.885</b>	C75	6.264	<b>3.438</b>
C32	1.217	<b>0.885</b>	C76	1.175	<b>0.930</b>
C33	0.215	0.169	C77	2.260	<b>1.289</b>
C34	0.317	0.231	C78	2.549	<b>1.536</b>
C35	1.769	1.648	C79	2.194	<b>1.807</b>
C36	0.240	0.200	C80	3.538	<b>2.208</b>
C37	0.234	0.223	C81	4.043	<b>2.519</b>
C38	2.472	<b>1.578</b>	C82	1.983	<b>1.195</b>
C39	0.413	0.385	C83	1.999	<b>1.689</b>
C40	0.314	0.246	C84	3.767	<b>2.302</b>
C41	0.465	0.416	C85	1.840	<b>1.572</b>
C42	0.667	<b>0.377</b>	C86	3.526	<b>2.290</b>
C43	0.398	0.362	C87	3.688	<b>2.272</b>
C44	0.492	<b>0.270</b>			

instances demonstrate that HACO A significantly outperforms other problem-specific heuristics.

The main contributions of this paper may be summarized as follows: 1) Motivated by practical requirements, an ECARP is proposed that considers both the total service time and fixed investment costs. 2) A HACO A is proposed to deal with the increased complexity of the ECARP. Domain-specific knowledge, arc cluster information and arc priority information, is continuously extracted from the solutions obtained throughout the search and is used to guide the subsequent optimization process. The dynamic parameter adjustment noticeably improves the performance of the algorithm and decreases the sensitivity of initial parameter choices. Finally, a local optimization step based on two-opt significantly improves the performance of HACO A. 3) An extensive experimental study on 87 problem instances has been carried out to evaluate the performance of HACO A. The results indicate the superiority of HACO A over other problem-specific heuristics.

The novel aspects of HACO A can be listed as follows: the extraction of heuristic information, the application of heuristic information, the dynamic parameter adjustment, and the local optimization based on two-opt. Although our approach tackles the ECARP, the novel points summarized from HACO A are still available to deal with some complicated combinatorial optimization problems.

There are numerous aspects that we intend to address in the near future. We believe that it would be beneficial to learn additional heuristic information (see, e.g., [62]) as heuristic information has proven itself to be highly beneficial when it comes to solving complex ECARP instances. Furthermore, the way this information is obtained may be improved further. In particular, some classical approaches in the fields of data mining and machine learning may be employed to extract a higher and possibly more fine-grained degree of heuristic information from the ongoing optimization process.

Please note that, in order to make a tradeoff between optimality and computation time, the service architecture was optimized via ERPS while the service scheduling was optimized using HACO A. For this reason, the optimized architecture is not really optimal, and there exists a large gap between the obtained optimal solution and the real optimal solution. In the future research, both the service architecture and service scheduling should be optimized synchronously using metaheuristic.

#### ACKNOWLEDGMENT

The authors would like to thank all members of the Evolutionary Algorithms for Dynamic Optimisation Problems reading group (School of Computer Science, University of Birmingham) for their helpful suggestions and, in particular, T. T. Nguyen, Dr. H. Handa, and Dr. T. Ray for their constructive comments. Parts of the work in this paper were carried out while L.-N. Xing was a visiting Ph.D. student at the Centre of Excellence for Research in Computational Intelligence and Applications, University of Birmingham, U.K.

#### REFERENCES

- [1] G. A. Tobin and R. Brinkmann, "The effectiveness of street sweepers in removing pollutants from road surfaces in Florida," *J. Environ. Sci. Health (Part A)*, vol. 37, no. 9, pp. 1687–1700, Sep. 2002.
- [2] M. Dror, H. Stern, and P. Trudeau, "Postman tour on a graph with precedence relation on arcs," *Networks*, vol. 17, no. 3, pp. 283–294, 1987.
- [3] E. Dijkgraaf and R. H. J. M. Gradus, "Fair competition in the refuse collection market," *Appl. Econ. Lett.*, vol. 14, no. 10, pp. 701–704, Aug. 2007.
- [4] H. Handa, L. Chapman, and X. Yao, "Robust route optimisation for gritting/salting trucks: A CERCIA experience," *IEEE Comput. Intell. Mag.*, vol. 1, no. 1, pp. 6–9, Feb. 2006.
- [5] J. Araoz, E. Fernandez, and C. Zoltan, "Privatized rural postman problems," *Comput. Oper. Res.*, vol. 33, no. 12, pp. 3432–3449, Dec. 2006.
- [6] T. Bektas and S. Elmastas, "Solving school bus routing problems through integer programming," *J. Oper. Res. Soc.*, vol. 58, no. 12, pp. 1599–1604, Dec. 2007.
- [7] H. I. Stern and M. Dror, "Routing electric meter readers," *Comput. Oper. Res.*, vol. 6, no. 4, pp. 209–223, 1979.
- [8] H. S. Han, J. J. Yu, C. G. Park, and J. G. Lee, "Development of inspection gauge system for gas pipeline," *KSME Int. J.*, vol. 18, no. 3, pp. 370–378, 2004.
- [9] J. Tao, P. W. Que, and Z. S. Tao, "Magnetic flux leakage device for offshore oil pipeline defect inspection," *Mater. Perform.*, vol. 44, no. 10, pp. 48–51, 2005.
- [10] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Competitive memetic algorithms for arc routing problems," *Ann. Oper. Res.*, vol. 131, no. 1–4, pp. 159–185, Oct. 2004.
- [11] X. Fan and W. Liu, "Public service-oriented government and the construction of innovation-based city in China," in *Proc. 5th Int. Symp. Manage. Technol.*, vol. 2. Hangzhou, China: Zhejiang Univ. Press, 2007, pp. 1626–1630.

- [12] C. Blum and M. Dorigo, "Search bias in ant colony optimization: On the role of competition-balanced systems," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 159–174, Apr. 2005.
- [13] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.
- [14] C. Blum, "Beam-ACO—Hybridizing ant colony optimization with beam search: An application to open shop scheduling," *Comput. Oper. Res.*, vol. 32, no. 6, pp. 1565–1591, Jun. 2005.
- [15] K. K. Lim, Y. S. Ong, M. H. Lim, X. Chen, and A. Agarwal, "Hybrid ant colony algorithm for path planning in sparse graphs," *Soft Comput.*, vol. 12, no. 10, pp. 981–994, Aug. 2008.
- [16] R. Hirabayashi, Y. Saruwatari, and N. Nishida, "Tour construction algorithm for the capacitated arc routing problem," *Asia-Pacific J. Oper. Res.*, vol. 9, no. 2, pp. 155–175, 1992.
- [17] B. L. Golden and R. T. Wong, "Capacitated arc routing problems," *Networks*, vol. 11, no. 3, pp. 305–315, 1981.
- [18] 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.
- [19] W. L. Pearn, "Approximate solutions for the capacitated arc routing problem," *Comput. Oper. Res.*, vol. 16, no. 6, pp. 589–600, 1989.
- [20] G. Ulusoy, "The fleet size and mix problem for capacitated arc routing," *Eur. J. Oper. Res.*, vol. 22, no. 3, pp. 329–337, Dec. 1985.
- [21] W. L. Pearn, "Augment-insert algorithms for the capacitated arc routing problem," *Comput. Oper. Res.*, vol. 18, no. 2, pp. 189–198, 1991.
- [22] 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.
- [23] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1151–1166, Oct. 2009.
- [24] Y. S. Ong, M. H. Lim, and X. S. Chen, "Research frontier: Memetic computation—Past, present and future," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.
- [25] L. Feng, Y. S. Ong, and Q. H. Nguyen, "Towards probabilistic memetic algorithm: An initial study on capacitated arc routing problem," in *Proc. IEEE World Congr. Comput. Intell., Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 18–23.
- [26] P. Beullens, L. Muyltermans, D. Cattrysse, and D. Van Oudheusden, "A guided local search heuristic for the capacitated arc routing problem," *Eur. J. Oper. Res.*, vol. 147, no. 3, pp. 629–643, Jun. 2003.
- [27] K. F. Doerner, R. F. Hartl, V. Maniezzo, and M. Reimann, "Applying ant colony optimisation to the capacitated arc routing problem," in *Proc. Ant Colony Optimisation Swarm Intell.*, vol. 3172. Berlin, Germany: Springer-Verlag, 2004, pp. 420–421.
- [28] J. S. DeArmon, "A comparison of heuristics for the capacitated Chinese postman problem," M.S. thesis, Univ. Maryland, College Park, MD, 1981.
- [29] P. Greistorfer, "A tabu scatter search metaheuristic for the arc routing problem," *Comput. Ind. Eng.*, vol. 44, no. 2, pp. 249–266, Feb. 2003.
- [30] 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, Jan./Feb. 2000.
- [31] A. Hertz and M. Mittaz, "A variable neighborhood descent algorithm for the undirected capacitated arc routing problem," *Transp. Sci.*, vol. 35, no. 4, pp. 425–434, Nov. 2001.
- [32] E. Benavent, V. Campos, and A. Corberan, "The capacitated arc routing problem: Lower bounds," *Networks*, vol. 22, no. 7, pp. 669–690, Dec. 1992.
- [33] M. Polacek, K. F. Doerner, R. F. Hartl, and V. Maniezzo, "A variable neighborhood search for the capacitated arc routing problem with intermediate facilities," *J. Heuristics*, vol. 14, no. 5, pp. 405–423, Oct. 2008.
- [34] J. Brandão and R. Eglese, "A deterministic tabu search algorithm for the capacitated arc routing problem," *Comput. Oper. Res.*, vol. 35, no. 4, pp. 1112–1126, Apr. 2008.
- [35] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "A genetic algorithm for the capacitated arc routing problem and its extensions," in *Applications of Evolutionary Computing*, vol. 2037, *Lecture Notes in Computer Science*. Berlin, Germany: Springer-Verlag, 2001, pp. 473–483.
- [36] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Evolutionary algorithms for periodic arc routing problems," *Eur. J. Oper. Res.*, vol. 165, no. 2, pp. 535–553, Sep. 2005.
- [37] P. Lacomme, C. Prins, and M. Sevaux, "A genetic algorithm for a bi-objective capacitated arc routing problem," *Comput. Oper. Res.*, vol. 33, no. 12, pp. 3473–3493, Dec. 2006.
- [38] A. Amberg, W. Domschke, and S. Vo, "Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees," *Eur. J. Oper. Res.*, vol. 124, no. 2, pp. 360–376, Jul. 2000.
- [39] X. H. Li, Z. Y. Zhu, and M. S. Xia, "Resolve multiple capacitated arc routing problem based on improved genetic algorithm," *Comput. Eng. Appl.*, vol. 45, no. 11, pp. 230–234, 2009.
- [40] L. N. Xing, P. Rohlfshagen, Y. W. Chen, and X. Yao, "An evolutionary approach to the multidepot capacitated arc routing problem," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 356–374, Jun. 2010.
- [41] A. Kansou and A. Yassine, "New upper bounds for the multi-depot capacitated arc routing problem," *Int. J. Metaheuristics*, vol. 1, no. 1, pp. 81–95, 2010.
- [42] A. Kansou and A. Yassine, "A two ant colony approaches for the multi-depot capacitated arc routing problem," in *Proc. Int. Conf. Comput. Ind. Eng.*, Troyes, France, 2009, pp. 1040–1045.
- [43] F. Chu, N. Labadi, and C. Prins, "A scatter search for the periodic capacitated arc routing problem," *Eur. J. Oper. Res.*, vol. 169, no. 2, pp. 586–605, Mar. 2006.
- [44] F. Chu, N. Labadi, and C. Prins, "Heuristics for the periodic capacitated arc routing problem," *J. Intell. Manuf.*, vol. 16, no. 2, pp. 243–251, Apr. 2005.
- [45] F. Chu, N. Labadi, and C. Prins, "The periodic capacitated arc routing problem linear programming model, metaheuristic and lower bounds," *J. Syst. Sci. Syst. Eng.*, vol. 13, no. 4, pp. 423–435, Dec. 2004.
- [46] J. M. Belenguer, E. Benavent, N. Labadi, C. Prins, and M. Reghioui, "Split-delivery capacitated arc-routing problem: Lower bound and metaheuristic," *Transp. Sci.*, vol. 44, no. 2, pp. 206–220, May 2010.
- [47] M. Reghioui, C. Prins, and N. Labadi, "GRASP with path relinking for the capacitated arc routing problem with time windows," in *Proc. EvoWorkshops*, vol. 4448, *Lecture Notes in Computer Science*, 2007, pp. 722–731.
- [48] A. Amaya, A. Langevin, and M. Trepanier, "The capacitated arc routing problem with refill points," *Oper. Res. Lett.*, vol. 35, no. 1, pp. 45–53, Jan. 2007.
- [49] G. Laporte, R. Musmanno, and F. Vocaturto, "An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands," *Transp. Sci.*, vol. 44, no. 1, pp. 125–135, Feb. 2010.
- [50] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
- [51] M. Dorigo, V. Maniezzo, and A. Colomi, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.
- [52] C. Blum, "Ant colony optimization: Introduction and recent trends," *Phys. Life Rev.*, vol. 2, no. 4, pp. 353–373, Dec. 2005.
- [53] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2/3, pp. 243–278, Nov. 2005.
- [54] B. Bullnheimer, R. Hartl, and C. Strauss, "A new rank-based version of the Ant System: A computational study," *Central Eur. J. Opera. Res. Econ.*, vol. 7, no. 1, pp. 25–38, 1999.
- [55] T. Stutzle and H. H. Hoos, "Max-min ant system," *Future Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.
- [56] C. Blum and M. Dorigo, "The hyper-cube framework for ant colony optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 1161–1172, Apr. 2004.
- [57] Y. W. Leung and F. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimisation," *IEEE Trans. Evol. Comput.*, vol. 5, no. 1, pp. 41–53, Feb. 2001.
- [58] S. Lin and B. W. Kemighan, "An effective heuristic algorithm for the traveling salesman problem," *Oper. Res.*, vol. 21, no. 2, pp. 498–516, Mar./Apr. 1973.
- [59] M. Kiuchi, Y. Shinano, R. Hirabayashi, and Y. Saruwatari, "An exact algorithm for the capacitated arc routing problem using parallel branch and bound method," in *Proc. Abstr. Spring Nat. Conf. Oper. Res. Soc. Jpn.*, 1995, pp. 28–29.
- [60] 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, Apr. 2003.
- [61] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
- [62] J. He, X. Yao, and J. Li, "A comparative study of three evolutionary algorithms incorporating different amount of domain knowledge for node covering problems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 35, no. 2, pp. 266–271, May 2005.



**Li-Ning Xing** received the B.S. degree in economics and the B.S. degree in science from Xi'an Jiaotong University, Xi'an, China, in 2002, and the Ph.D. degree in management science from the National University of Defense Technology, Changsha, China, in 2009.

He is currently a Lecturer with the College of Information Systems and Management, National University of Defense Technology. His research interests include management decision making, simulation optimization, and artificial intelligence.



**Philipp Rohlfshagen** received the B.S. degree in computer science and artificial intelligence from the University of Sussex, Brighton, U.K., in 2003, and the M.S. degree in natural computation and the Ph.D. degree in evolutionary computation from the University of Birmingham, Birmingham, U.K., in 2004 and 2007, respectively.

He is currently a Research Fellow at the Center of Excellence for Research in Computational Intelligence and Applications, School of Computer Science, University of Birmingham, where he is

studying the theoretical aspects of evolutionary algorithms for dynamic optimization problems. His research interests include combinatorial optimization, computational complexity, evolutionary computation, global optimization, algorithm design, and molecular genetics.



**Ying-Wu Chen** received the B.S. degree in automatic control, the M.S. degree in system engineering, and the Ph.D. degree in engineering from the National University of Defense Technology (NUDT), Changsha, China, in 1984, 1987, and 1994, respectively.

Since 1999, he has been a Professor at NUDT, where he was a Lecturer from 1989 to 1994 and an Associate Professor from 1994 to 1999. He is a Distinguished Professor and the Director of the Department of Management Science and Engineering,

College of Information System and Management, NUDT, where he focuses on management theory and application. He has authored more than 60 research publications. His current research interests include assistant decision-making systems for planning, decision-making systems of project evaluation, management decision, and artificial intelligence. He is the Editor of *Principles of System Engineering* and *Technology of Quantificational Analysis*.



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

He is currently a Chair (Full Professor) of Computer Science and the Director of the Center of Excellence for Research in Computational Intelligence and Applications, University of Birmingham,

Birmingham, U.K. He is a Distinguished Lecturer of the IEEE Computational Intelligence Society, a Distinguished Visiting Professor at USTC, and a Visiting Professor at Nanjing University, Xidian University, and Northeast Normal University. He has been an invited keynote speaker at more than 50 international conferences and has more than 300 refereed research publications in evolutionary computation and neural network ensembles. His research interests include evolutionary computation, neural network ensembles, global optimization, data mining, computational time complexity of evolutionary algorithms, and real-world applications. In addition to basic research, he works closely with many industrial partners on various real-world problems.

Dr. Yao was an Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008 and is an Associate Editor or an Editorial Board Member of ten other international journals. He is the Editor of the *World Scientific Book Series on Advances in Natural Computation* and a Guest Editor of several journal special issues.