Applied Soft Computing 49 (2016) 748–758

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Immune clonal algorithm based on directed evolution for multi-objective capacitated arc routing problem

Ronghua Shang[a,*], Bingqi Du[a], Hongna Ma[a], Licheng Jiao[a], Yu Xue[b], Rustam Stolkin[c]

[a] Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education of China, Xidian University, Xi'an, China
[b] School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, China
[c] School of Computer and Software Extreme Robotics Lab, University of Birmingham, UK

ARTICLE INFO

Article history:
Received 15 November 2015
Received in revised form 2 September 2016
Accepted 3 September 2016
Available online 7 September 2016

Keywords:
Immune clone
Decomposition algorithm
Comparison operator
MO-CARP

ABSTRACT

The capacitated arc routing problem is playing an increasingly important role in our society, engendering increasing attention from the research community. Among the various models, multi-objective capacitated arc routing problem comes much closer to real-world problems. Therefore, this paper proposes an immune clonal algorithm based on directed evolution to solve this problem. Firstly, the proposed algorithm adopts the framework of the immune clonal algorithm and expands the scale of the initial antibody population in the initialization process to increase the diversity of the antibodies. Secondly, the proposed algorithm is combined with a decomposition strategy in the operations of the immune gene. Antibodies are classified to perform the immune genetic operations, which helps the antibody populations to share the neighborhood information in a timely manner. Thirdly, the proposed algorithm applies a novel kind of comparison operator to build the total population, which helps it to evolve in the direction of a better population and improves the quality of the antibodies. Experimental results suggest that the proposed algorithm can generate better non-dominant solutions than several compared state-of-the-art algorithms, especially for large-scale sets.

1. Introduction

THE Arc Routing Problem (ARP) has wide applications to practical problems in daily life, such as snow removal in winter, urban rubbish collection, and sprinkler path planning, etc. [1,2]. One of the most important models is Capacitated ARP (CARP) which is much closer to many real-world problems. CARP aims to find the optimal routes which can serve all predetermined tasks under the condition of meeting the capacity of vehicles [3,4]. When CARP has only one objective (total cost), we refer to it as single objective CARP. For single objective CARP, researchers have proposed many effective algorithms, including heuristic algorithms and metaheuristic algorithms. Heuristic algorithms mainly include Path Scanning algorithm [5], Augment Assignment algorithm [6] and Ulusoy-Split algorithm [7]. Heuristic algorithms can converge to the local optimal solutions in a relatively short time, so the algorithms are effective for relatively small-scale instances. In contrast, for the large-scale instances, it is hard for the algorithms to escape from local optima, causing algorithms to fail to reach ideal solutions. Scholars have proposed advanced metaheuristic algorithms in order to improve this problem, which has been applied to solving ARP and Vehicle Routing Problems [8]. Typical metaheuristic algorithms are Simulated Annealing Algorithm for the salt in the wintertime problem [9], Tabu Search algorithm [10], Guided Local Search algorithm [11], Memetic Algorithm [12], Memetic Algorithm based on extended neighborhood search [13] and Cooperative Coevolution with route distance grouping for large-scale CARPs [14]. These metaheuristic algorithms offer advantages in efficiency, optimal solutions and stability when solving the basic CARP.

However, in practical applications, the relevant end-users not only want to minimize the total cost but also consider other factors. For example, in the rubbish collection example in Troyes, France mentioned in literature, end-users not only hope to minimize the total cost but also wish to complete the rubbish clean-up as rapidly as possible. Considering this case, Lacomme et al. set up a corresponding model which minimizes both the total cost and the makespan (the cost of the longest circuit) [15]. We consider the CARP with two objectives as Multi-Objective CARP (MO-CARP). It is clear that the two objectives are conflicting and they cannot both achieve optimal values at the same time. Therefore, there is no unique global optimal solution when solving MO-CARP and optimization methods usually attempt to retain solutions which provide a good balance between the two objectives.

* Corresponding author.
E-mail address: rhshang@mail.xidian.edu.cn (R. Shang).

http://dx.doi.org/10.1016/j.asoc.2016.09.005
1568-4946/

To solve MO-CARP, some algorithms have been proposed. In 2006, Lacomme [15] proposed an effective genetic algorithm (LMOGA) to overcome these problems for the first time. LMOGA combines fast non-dominant sorting with a selection strategy based on crowding distance. In 2010, Grandinetti et al. proposed a constraint method, in which three objectives are considered and an optimization-based heuristic procedure is proposed to find a set of solutions belonging to the optimal Pareto front and a good performance is obtained [4,16]. In 2011, Mei et al. put forward a more effective algorithm, namely Decomposition- Based Memetic Algorithm (D-MAENS) [17]. It adopts the framework of the multi-objective evolutionary algorithm based on the problem decomposition and embeds MAENS algorithm for the single objective CARP. Firstly, it decomposes the whole MO-CARP into several subproblems and then it maintains a population during the searching process, in which each individual corresponds to a unique subproblem. Next, a crossover operator and local search, similar to this used in MAENS, are applied to solve each subproblem. Experimental results show that D-MAENS is better than LMOGA, but there is still room to improve the speed of convergence and the assignment of computing source. To overcome this issue, in 2014, an Improved Decomposition-Based Memetic Algorithm (ID-MAENS) was proposed which adds an elite strategy contributing to the reservation of good solutions [18]. The experimental results show that ID-MAENS can obtain better non-dominant solutions than other existing algorithms when solving MO-CARP. Furthermore, a multi-population cooperative co-evolutionary algorithm was proposed for MO-CARP. The algorithm applies a variety of elite storage mechanisms and adopts an evolutionary strategy and a local search strategy based on extended neighborhood. Experimental results show that the algorithm yields better performance and faster convergence speed [19]. In 2015, a multi-objective evolutionary algorithm, coined as Memetic NSGA-II [20], has been designed. It is a hybrid of non-dominated sorting genetic algorithm-II, which shows the energetic effects [21].

Although these algorithms have their own advantages in solving MO-CARP, there are also some limits in the capacity of searching a more optimal solution because of the limited diversity, especially for the large sale MO-CARP. To overcome this issue and improve the capacity of searching a more optimal solution effectively, based on the advantages of some effective algorithms in the numerical multi-objective optimization problems [22–25], this paper proposes a novel algorithm for MO-CARP, which we refer to as an immune clonal algorithm based on directed evolution (DE-ICA). Firstly, DE-ICA uses the framework of the immune clonal algorithm and enlarges the size of the initial population, increasing the diversity of the population and improving the quality of solutions. Secondly, DE-ICA combines the immune gene manipulation with the decomposition algorithm. The algorithm performs reorganization and mutation operations under the framework of the decomposition algorithm. The decomposition operation helps to share information between adjacent populations and rapidly converge to ideal solutions [26]. Finally, DE-ICA applies a novel comparison operation to select the best individuals to join the next evolutionary iteration, which helps it to evolve in the direction of a better population and improves the quality of the antibodies. DE-ICA can constantly optimize in the specified direction and easily escapes from local optima.

2. The description of MO-CARP

Basic CARP can be described as follows: given an undirected connected graph $G(V, E, A)$, $V = \{v_0, v_1, \ldots, v_n\}$ is vertex set, where v_0 is the depot. $E = \{(v_i, v_j) \in V, i \neq j\}$ means two-way edge set connected by two vertices. Each edge has three non-negative attributes: service cost S_c, travel cost T_c which means the cost of shortest path [27] between two vertex and demand d. If $d > 0$, then the edge is called a task edge. Also, each arc has three non-negative attributes: service cost S_c, travel cost T_c and demand d. If $d > 0$, then the arc is called a task arc [13]. Some vehicles start from the depot and serve the task edges and the task arcs, under the condition of not exceeding the capacity of the vehicles, and finally return to the depot. All the task edges and task arcs should be served and be served only once and the goal is to minimize the total cost of the whole route [14]. Therefore, the definition of CARP is as follows:

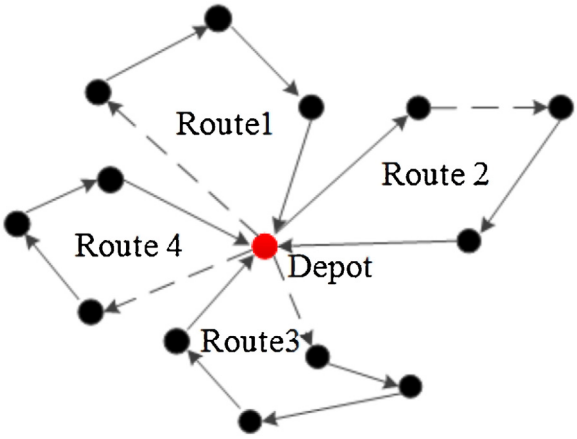

Fig. 1. A simple example of basic CARP.

$$\begin{aligned}
&\min f(\boldsymbol{x}) = \sum_{g=1}^{n} cost(Tg)\\
&cost(Tg) = \sum_{i=1}^{|Tg-1|} Sc(vgi, vg(i+1)) \times ygi + Tc(vgi, vg(i+1)) \times \quad (1 - ygi)\\
&s.t.\ (vgi, vg(i+1)) \in ER \cup AR, \quad \forall ygi = 1,\ 1 \le g \le n\\
&\quad (vgi, vg(i+1)) \neq (vkj, vk(j+1)), \quad \forall ygi = 1,\ ykj = 1,\ g \neq k\\
&\sum_{i=1}^{|Tg|-1} d(vgi, vg(i+1)) \le Q, \quad 1 \le g \le n
\end{aligned} \tag{1}$$

where T_g represents the g-th service circuit and $|T_g|$ means the length of the circuit. T_g can be expressed by the vertex sequence as $T_g = (v_{g1}, v_{g2}, \ldots, v_{g|Tg|})$. Here v_{gi} denotes the i-th task in the g-th service circuit. If $y_{gi} = 1$, then the edge $(v_{gi}, v_{g(i+1)})$ should be served. If $y_{gi} = 0$, then the edge $(v_{gi}, v_{g(i+1)})$ is not a task edge and it has no service cost but does incur travel cost. n is the total number of the circuits in the solution and $1 \le g \le n$. E_R and A_R represent the sets of the task edges and the task arcs respectively. Q is the capacity of the vehicles.

In Fig. 1, the red node represents the depot and two black nodes form an edge. There are four circuits in Fig. 1 and in each circuit, the solid lines denote task edges and dotted lines denote non-task edges. The arrows represent the driving direction of the vehicles.

Based on the mathematical model of single objective CARP and the same constraints in Formula (1), the MO-CARP can be described as follows [6]:

$$\begin{cases}
\min f_1(\boldsymbol{x}) = \sum_{g=1}^{n} cost(Tg)\\
\min f_2(\boldsymbol{x}) = \max(cost(Tg)), \quad 1 \le g \le n
\end{cases} \tag{2}$$

In the mathematical model, the goals of established MO-CARP models are to minimize both the total cost of the route and the cost of the longest circuit. For a better understanding of the MO-CARP, we provide three useful definitions [28,29].

Definition 1 *(Pareto-Dominance)*. Based on the MO-CARP described in Formula (2), given two feasible solutions $\boldsymbol{x}$, $\boldsymbol{x}^* \in \boldsymbol{\Omega}$, $\boldsymbol{x}^*$ known as the non-dominant solution dominates $\boldsymbol{x}$ (denoted $\boldsymbol{x}^* \succ \boldsymbol{x}$) only if the following conditions are met:

$$\begin{cases} f_1(\boldsymbol{x}^*) < f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}^*) \le f_2(\boldsymbol{x}) \end{cases} \text{ or } \begin{cases} f_1(\boldsymbol{x}^*) \le f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}^*) < f_2(\boldsymbol{x}) \end{cases} \tag{3}$$

Definition 2 *(Pareto-Optimal and Pareto-Optimal Set)*. Feasible solution $\boldsymbol{x}^* \in \boldsymbol{\Omega}$ is called Pareto-optimal solution only when the following formula is satisfied:

$$\forall \boldsymbol{x} \in \Omega, \ \neg \exists \boldsymbol{x} \succ \boldsymbol{x}^* \tag{4}$$

The Pareto-optimal set P_S is the set of all the Pareto-optimal solutions.

Definition 3 *(Pareto-Optimal Front [25])*. The set P_F constituted by the function values of the two goals corresponding to the Pareto-optimal solutions in Pareto-optimal set P_S is called Pareto-optimal front:

$$PF = \{\boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))^T | \boldsymbol{x} \in Ps\} \tag{5}$$

3. Description of DE-ICA

The Immune Clonal Algorithm Based on Directed Evolution (DE-ICA) adopts the process of immune clonal algorithm as a framework and draws on this effective decomposition algorithm. Meanwhile, DE-ICA analyzes and improves some defects of the current algorithms for MO-CARP. Compared with other algorithms, the immune clonal algorithm has advantages on quick convergence and global optimization [25,30]. The immune clonal algorithm uses a heuristic algorithm to generate an initial antibody population, and then evaluates the fitness of the initial antibodies and determines the clonal ratio of the antibodies by calculating the affinities between antibodies and antigens [31]. Next, the immune clonal algorithm performs immune gene operations including gene recombination and gene mutation. Finally, the immune clonal algorithm selects offspring for the next iteration according to certain principles through the clonal selection operation [20]. Here, the antigen represents the objective function and constraint condition and the antibody is the solution that satisfies the objective function and constraint condition. Specific to MO-CARP, DE-ICA firstly initializes the antibody population. Compared with the existing algorithms for MO-CARP, DE-ICA enlarges the size of the initial population to increase the diversity of antibodies. Secondly, DE-ICA directly expands the population size according to the characteristics of MO-CARP, which is helpful for improving the quality of solutions and converging to Pareto-optimal solutions. Thirdly, DE-ICA decomposes the whole problem into several subproblems combining with the decomposition strategy which is conducive for sharing information between adjacent populations and convergence to a better solution. Then, antibodies in the population are divided into antibody sub-populations. Those sub-populations represent the solution of the corresponding sub-problem to perform the immune gene operations. As a result, the decomposition strategy facilitates the algorithm to converge fast on the two objectives. At the same time, this paper proposes a novel directed comparison operator to filtrate the antibodies produced in the previous process and the selected antibodies are added into the total population as candidates for the clonal selection. Finally, DE-ICA applies a fast non- dominant sorting and crowding distance method to evaluate the candidate antibodies and select offspring, which can ensure both the quality of solutions and the diversity of the antibody population. This constitutes a very effective strategy for choosing offspring. We will introduce each part of the DE-ICA algorithm below. In the following sections, the total population represents the enlarged candidate population for the next iteration.

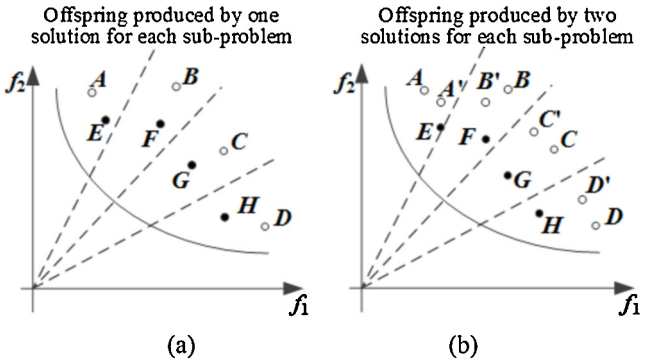

Fig. 2. The influence of population size on solutions' diversity and quality. (a) Offspring *E*, *F*, *G*, and *H* produced by solutions *A*, *B*, *C* and *D*. (b) Offspring *E*, *F*, *G*, and *H* produced by solutions *A*, *A*′, *B*, *B*′, *C*, *C*′ and *D*, *D*′.

3.1. Antibody initialization

The antibody initialization operation in DE-ICA adopts the path-scanning algorithm which is a classic heuristic algorithm proposed by Golden in 1983 [5]. The basic principle of the path-scanning algorithm for solving CARP can be described as follows. First we establish an empty route, and then insert tasks into the route according to certain principles. If the total demands of the route exceed the capacity *Q* after inserting some task, then we give up on that task and the vehicle returns to the depot directly from the end of the last inserted task.

The size of the population has a great influence on the solutions [25], which also results in different performance of algorithms. The influence of the population size on the effectiveness of solutions is described below in detail.

In Fig. 2(a), f_1 is the total cost and f_2 stands for the longest route cost, and each sub-problem is assigned to one solution (*A*, *B*, *C* or *D*). According to the theory of the decomposition algorithm, *A* should produce an offspring solution (denoted *E*) with an adjacent solution. In a similar fashion, *B*, *C* and *D* produce offspring solutions (sequentially denoted *F*, *G* and *H*) with their adjacent solutions. By contrast, in Fig. 2(b), there are two representative solutions in each sub-problem (*A*, *A*′, *B*, *B*′, *C*, *C*′ and *D*, *D*′). We can use the roulette method to select individuals from the representative solutions as parents and the selected individuals produce offspring with their adjacent solutions. Clearly it is beneficial for the algorithm to find better solutions quickly because of the selectivity of solutions in each sub-problem. On the other hand, enlarging the size of the population can increase the diversity of the population to some extent, which is beneficial for improving the quality of the solutions.

3.2. Immune clonal operation

In the theory of artificial immune systems, the clonal operation is carried out by reproducing the antibodies in the population according to a certain proportion. The clonal operation makes various gene operations possible and facilitates antibodies to share information [32,33]. The immune clonal algorithm usually regards the total cost of the whole solution as a key evaluation index of the affinity between antibody and antigen, when solving single objective CARP. For instance, we usually define the affinity between the antibody S_i and the antigen as:

$$Aff(si) = \left(\frac{lower\_bound}{total\_\cos t(si)}\right)^3 \tag{6}$$

where, *lower_bound* represents the lower bound of the test instance which can be obtained from the reference literature. The bigger the affinity's value is, the smaller the total cost of the solution is. Clonal

proportion is not just about the affinity between antigen and antibody, but about the affinity between antibodies. The greater the affinity between antibodies is, the higher the similarity between antibodies and the easier it will be for antibodies to restrain each other. The immune clonal algorithm usually sets the clonal proportion based on the two values of affinities previously mentioned.

In the process of solving MO-CARP, the goals are to minimize the total cost and the cost of the longest circuit at the same time. Because the calculation of affinities is very complex, the calculation of antibody clonal ratio is also complex, especially when solving large-scale CARP. From the literature [31], we can see that when the immune clonal algorithm is used for solving the CARP, the ratio of clone is set as 3 which helps increase the antibody's diversity and promotes good performance under the premise of the lower computational cost. Therefore, we directly clone the non-dominant solutions in the initial population at the ratio 3 in order to guarantee the speed and the simple calculation of DE-ICA, and then the clonal individuals are added into the initial population. This approach increases the proportion of the good solutions in the initial population, which is beneficial for improving the quality of the solutions.

3.3. Immune gene operations

Immune gene operations usually include genetic recombination and mutation. Immune gene operations can increase the diversity of the population, decrease the affinity between antibodies and improve the quality of the solutions. Decomposition algorithm is an effective method for MO-CARP [14], so DE-ICA uses this framework in the immune gene operations.

3.3.1. Population decomposition operation

The cooperative co-evolution algorithm was originally proposed by Potter et al. [34]. The main idea is to divide a problem into many sub-problems, and then solve the sub-problems independently. The application of the decomposition strategy in DE-ICA is inspired by the work of [17] and [35]. Uniformly distributed weight vectors $w_1,\cdots,w_R$ decompose the MO-CARP with two goals into R single objective sub-problems. The function expression can be described by the i-th weight vector as follows:

$$Fi(x) = \lambda i1 \times f1(x) + \lambda i2 \times f2(x), \quad 1 \le i \le R \tag{7}$$

where, R is set to 60 and both $f_1(x)$ and $f_2(x)$ are normalized [17]. $\boldsymbol{\lambda}_i$ is a two-dimensional vector, which represents the weight vector of the i-th sub-problem and can be expressed as: $\lambda i = (\frac{i-1}{R-1}, 1 - \frac{i-1}{R-1})$. When solving each of the single objective sub-problems, DE-ICA firstly assigns the new population composed by the cloned non-dominant solutions and the initial solutions into the corresponding sub-problems. Considering the expression of Formula (7), the principle by which we assign the individuals is to sort them according to ascending order of the second objective function. We assign the $(2i\text{-}1)$-th sorted individual and the $2i$-th sorted individual into the i-th sub-problem for the first iteration. Because we have cloned non-dominant solutions, the same individuals are also in the new population. We assign the same individuals into the same sub-problem to increase the probability of their election as parents, which also help to improve the quality of solutions. Algorithm 1 presents the pseudo code of the decomposition algorithm in DE-ICA.

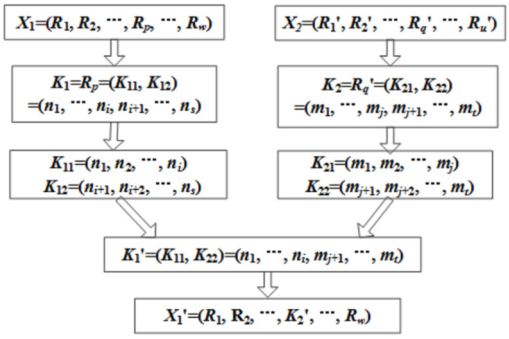

Fig. 3. The operation process of SBX gene recombination operator.

Algorithm 1: Decomposition operator in DE-ICA

```
1: procedure Decomposition(popsize, R);
2:Sort the antibodies in the initial population in ascending order
  according to the second goal;
3: for i = 1 → popsize do
4:     for j = 0→ R do
5:         Assign the i-th antibody into the j-th sub-problem;
6:     end for
7: end for
8: Match the cloned antibodies into the corresponding sub-problems;
9:         for j = 0→ R do
10:        Select antibodies in the j-th sub-problem by roulette method
to perform the immune gene operations and get a new antibody;
11:        end for
12:        Update the antibodies in the population; The new antibody
           population contains the candidates for the clonal selection;
13: end procedure
```

Where R means that the MO-CARP is decomposed into R single objective sub-problems, and *popsize* is the size of the initial antibody population. In this paper, R=60 and *popsize*=120.

3.3.2. Gene recombination operator

DE-ICA algorithm selects the effective Sequence Based Crossover (SBX) of the current recombination operators [13]. Appropriately selected parents can obtain new antibodies by the gene recombination operation. The process of SBX operator is shown in Fig. 3.

In Fig. 3, first, we randomly select a route K_1 from the parent X_1 and a route K_2 from the parent X_2. Next, we randomly divided K_1 and K_2 into two sub-routes expressed as $K_1 = (K_{11}, K_{12}), K_2 = (K_{21}, K_{22})$. Finally, we replace the route K_{12} with the route K_{22} and the new antibody $X_{1'}$ is formed. It may be the case that there are repetitive tasks or missing tasks in the new antibody X_1'. If X_1' has repetitive task, then we only retain the positions of the tasks in K_{22}. If $X_{1'}$ has missing tasks, then we insert the missing tasks into the routes under certain conditions. In order to guarantee the quality of solutions, the locations of the insertions should meet the condition that at least one of the produced extra cost and the violations is smaller than inserting other location.

3.3.3. Gene mutation operator

Gene mutation provides the possibility of obtaining various kinds of antibodies. Gene mutation improves the quality of antibodies and helps the algorithm to escape from local optima. Because the search range of a single gene recombination operator is small, gene mutation is very helpful to search the related area comprehensively [13].

DE-ICA performs the gene mutation operation with a probability of 0.2 on the basis of gene recombination. The algorithm adopts four traditional gene mutation operators, namely Single-Insertion, Double-Insertion, Swap and 2-opt. Here, we introduce the four operators briefly.

(1) Single-Insertion: where the operator randomly selects a route and then randomly selects a task from the route when performing the gene mutation operation. Next, the task is reinserted into another location or directly connected with the depot to build a new route. If the task is an edge task, then the situation where the task is inserted in the opposite direction should be considered and the corresponding antibody with least total cost will be reserved.

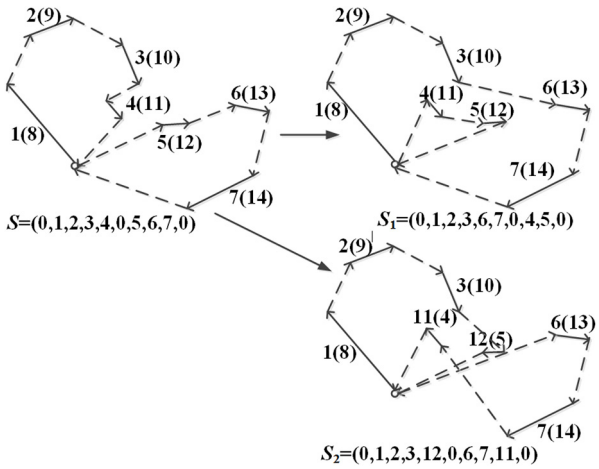

Fig. 4. A simple example of 2-opt working on double routes.

(2) Double-Insertion: where the principle of the operator is similar to that of Single-Insertion. The difference between Single-Insertion and Double-Insertion is that Double- Insertion randomly selects a route and randomly selects two continuous tasks in the route. Then the two continuous tasks are reinserted into other locations. Similarly, if the selected tasks are edge tasks, then the situation that the tasks are inserted in the opposite direction should also be considered.

(3) Swap: where the operator randomly selects two different tasks in the sequences of an antibody, and swaps the locations of the two tasks.

(4) 2-opt: consists of two different kinds of operators. One of them is for single route, and another is for double routes [25]. DE-ICA applies the 2-opt operator for double routes. The operator first randomly selects two routes (denoted K_1 and K_2) in an antibody. Next, K_1 and K_2 are randomly decomposed into two parts (respectively denoted K_{11}, K_{12} and K_{21}, K_{22}). Finally, there will be two candidate antibodies formed by reconnecting the four routes through different ways of connection. One of the two antibodies is made up of K_{11}, K_{22} and K_{12}, K_{21}. Another antibody is made up of K_{11} and the opposite direction of K_{21}, K_{22} and the opposite direction of K_{12}. Fig. 4 describes a simple example of 2-opt working on double routes.

In Fig. 4, the solid lines show the task lines and the dotted lines mean travel lines. Arrows indicate the directions of vehicles to serve tasks. Each task has two serial numbers. The serial numbers outside the parentheses represent the current driving direction, and the serial numbers inside the parentheses represent the opposite of the current driving directions. Select the two circuits before the action of 2-opt operator, expressed as $S=(0,1,2,3,4,0,5,6,7,0)$, where 0 represents the depot and two circuits are $K_1=(0,1,2,3,4,0)$ and $K_2=(0,5,6,7,0)$. The previous route is divided into four sub-routes after 2-opt operator has acted on them, namely $K_{11}=(0,1,2,3,0), K_{12}=(0,4,0), K_{21}=(0,5,0)$ and $K_{22}=(0,6,7,0)$. There will be two different antibodies according to the way of connecting above. The two antibodies are respectively expressed as $S_1=(0,1,2,3,6,7,0,4,5,0)$ and $S_2=(0,1,2,3,12,0,6,7,11,0)$. The antibody with less total cost will be output as the result of the 2-opt operator.

These gene mutation operators are simple and effective. In DE-ICA, we select one of the four operators to generate a new antibody with a probability of P_m and apply the four operators at the same time to get four new antibodies with a probability of $(1\text{-}P_m)$. Then we select the antibody with least total cost as the output antibody. Choosing one of the four algorithms randomly is helpful for increasing the diversity of antibodies, while the application of four algorithms to generate antibodies can improve the affinity of antibodies. P_m is set to 0.6 based on the above consideration. On the one hand, it is beneficial to keep antibodies with high affinity. On the other hand, it helps the algorithm to maintain the diversity of antibodies to help avoid local optima. The pseudo code of the gene mutation operator is shown in Algorithm 2.

Algorithm 2: gene mutation operator
1: **procedure** gene mutation (P_m)
2: Randomly generate a number v between 0 and 1;
3: **if** $v<P_m$ **then**
4: Randomly select one of the four operators to obtain a new antibody, denoted B_{out};
5: **else**
6: Achieve a new antibody by Single-Insertion, denoted A;
7: Achieve a new antibody by Double-Insertion, denoted B;
8: Achieve a new antibody by Swap, denoted C;
9: Achieve a new antibody by 2-opt, denoted D;
10: Compare the four antibodies (A, B, C, D) and select one according to certain rules, denoted B_{out};
11: **end if**
12: B_{out} is the final antibody produced in this process;
13: **end procedure**
Where P_m represents the probability of obtaining new antibodies by a randomly selected operator. In this paper, P_m=0.6.

3.3.4. Directed comparison operator

The current algorithms for MO-CARP usually add the individuals obtained by gene recombination or the individuals obtained by the gene mutation. However, we know that evolutionary algorithms are highly random. Suppose that we have obtained a good individual after the gene recombination of parents, at this point, the individual undergoes gene mutation. After mutation, we may get an individual which is worse than the one before the gene mutation, so such operations might not necessarily be beneficial to the reservation of good solutions and the fast convergence of the algorithm. In order to overcome the deficiencies of this method, we compare the individual before gene mutation with the individual after gene mutation and incorporate the better solution into the total population.

For MO-CARP, the process of individual selection is usually more complicated than for single objective CARP. In this paper, MO-CARP needs to optimize both the total cost and the cost of the longest circuit. Considering the objective function of the single objective CARP, we usually pay more attention to the optimization of the total cost when solving practical problems. And for the cost of the longest circuit subjective to the capacity of vehicles, the potential of the optimization of total cost is greater. Therefore, for simplicity, DE-ICA compares the total cost of the individual before gene mutation with that of the individual after gene mutation and selects the one with smaller total cost first. If the total cost of the two individuals is equal, then we compare the cost of the longest circuit of the two individuals and the individual with smaller cost of the longest circuit will be the first choice. This approach forms a state of directed evolution and the whole population evolves fast in the direction of the reduction of the total cost. This method adopts four kinds of gene mutation operation which enhance the diversity of solutions so as to help DE-ICA to find a better solution. The use of the directed comparison operator guides the whole evolution to the global optimal solution improving the searching efficiency. The pseudo code of the directed comparison operator in DE-ICA is shown in Algorithm 3.

Algorithm 3: Directed comparison operator in DE-ICA
1: **procedure** Directed comparison operator (R, P_n)
2: **for** $i = 0 \rightarrow R$ **do**
3: Mark the antibody gotten by the gene recombination as A_{out};
4: Randomly generate a number w between 0 and 1;
5: **if** $w<P_n$ **then**
6: Randomly select one of the four gene mutation operators to obtain an antibody, denoted B_{out};
7: **else**
8: Achieve a new antibody by Single-Insertion, denoted A;
9: Achieve a new antibody by Double-Insertion, denoted B;
10: Achieve a new antibody by Swap, denoted C;
11: Achieve a new antibody by 2-opt, denoted D;
12: Compare the four antibodies (A, B, C, D) and select one according to certain rules, denoted B_{out};
13: **end if**
14: Compare the total cost of A_{out} and B_{out} and add the one with smaller total cost into the total population. If the total cost of the two antibodies is equal, then we add the individual with smaller cost of the longest circuit into the total population;
15: **end for**
16: **end procedure**
Where P_n =0.2means the probability of gene mutation and in this paper.

3.3.5. Clonal selection operator

The clonal selection is a reverse process of the clonal proliferation. Its purpose is to select the antibodies with higher affinity from the offspring obtained by cloning and proliferation. For single objective CARP, random sorting method is known to be an effective method [36]. It adopts the basic frame of bubble-sort, the difference is that there is a probability p_f. When comparing the quality of two solutions, a number is generated randomly. If this number is smaller than p_f or both two compared solutions are feasible, the criterion of sorting is the objective function value. Otherwise, the criterion is the violation. It balances objective and penalty functions directly and explicitly in optimization. For MO-CARP, considering we always use the total cost as the primary criteria to select new individuals in directed comparison operator, we apply the fast non-dominant sorting and crowded distance method, mentioned in [17], to avoid convergence on local optima. Fast non-dominant sorting and crowded distance method can ensure not only the quality of solutions but also the diversity of the antibody population. This forms a very effective strategy for choosing offspring and plays an important role in the DE-ICA algorithm.

3.3.6. The processing flow of DE-ICA

This paper proposes a novel immune clonal algorithm based on directed evolution to solve MO-CARP. It applies the main idea of the immune clonal algorithm. It first initializes an antibody population and then carries out the immune clonal operation. Next, the algorithm performs immune gene operations and draws lessons from the framework of the effective decomposition algorithm during the process of immune gene operations. At the same time, a directed comparison operator is added into the algorithm. Finally, it performs the clonal selection operation. The main steps of DE-ICA can be described in Algorithm 4.

Algorithm 4: DE-ICA for MO-CARP

1: **procedure** DE-ICA (R, P_n)
2: Set the termination condition and initialize the iteration ite=0, then set the size of the initial antibody population to P_{size};
3: Initialize the initial population $\boldsymbol{P}$ by path-scanning algorithm;
4: **while** ite<200 **do**
5: Perform fast non-dominant sorting on the population $\boldsymbol{P}$ and clone the non-dominant solutions to obtain a new antibody population, namely the total population, denoted $\boldsymbol{P}_t$;
6: Allocate the antibodies in the population $\boldsymbol{P}_t$to the corresponding sub-problems according to certain rules.
7: **for** $i = 0 \rightarrow R$ **do**
8: Perform gene recombination operation on antibody $\boldsymbol{A}_{nti1}$ in the i-th sub-problem according to the theory of decomposition algorithm to achieve a new antibody, denoted $\boldsymbol{A}_{nti2}$;
9: Randomly generate a number u between 0 and 1;
10: **if** u<P_n **then**
11: Perform gene mutation operation on antibody $\boldsymbol{A}_{nti2}$ to achieve a new antibody, denoted $\boldsymbol{A}_{nti3}$;
12: Compare antibody $\boldsymbol{A}_{nti2}$ and antibody $\boldsymbol{A}_{nti3}$ then select one according to certain rules to add into the population $\boldsymbol{P}_t$;
13: **end if**
14: Put antibody $\boldsymbol{A}_{nti2}$ into the population $\boldsymbol{P}_t$;
15: **end for**
16: Sort the antibodies in the population $\boldsymbol{P}_t$ by fast non-dominant sorting and crowded distance method then put the top P_{size} different antibodies into the population for the next iteration, denoted $\boldsymbol{P}_t$;
17: ite=ite+1;
18: Perform fast non-dominant sorting on the population $\boldsymbol{P}_t$and output the non-dominant solutions into the population $\boldsymbol{P}_2$;
19: **end procedure**

3.4. Time complexity analysis of the algorithms

With the purpose of testing the proposed algorithm more objectively, we analyze the computational complexity of DE-ICA and make a comparison with D-MAENS and ID-MAENS. In each iteration of evolution of DE-ICA, we assume that the scale of the population is N, the number of objectives is r, the scale of antibody corresponding to the non-dominated solution set is N_n, the number of non-dominated solutions to maintain in each iteration is M, the colon ratio is q, and the probability of gene mutation is P_m. Therefore, the computational complexity of initializing the antibody population and calculating each objective is $O(Nr)$ and the colony operation's computational complexity is $O(N_nq)$, the computational complexity of gene recombination is $O({N_n}^2q^2)$, the computational complexity of gene mutation is $O(N_nqP_m)$, the computational complexity of selecting M non-dominated solution from N_n non-dominated solutions is $O(rMN_n\log N_n)$. Thus, the total computational complexity in each iteration of DE-ICA is $O(Nr+N_nq+{N_n}^2q^2+N_nqP_m+rMN_n\log N_n)$ and the time complexity of DE-ICA becomes $O((q^*N_n)^2)$ after simplified.

As for D-MAENS and ID-MAENS, they have the same time complexity as $O(Nr+N^2+N^2P_m+r(2N\text{-}1)^2/2)$ and it is simplified as $O(N^2)$. Especially, in all tested instances we found that the value of N_n ranges from $N/5$ to $N/2$. At the time, the ratio of colon q in DE-ICA is 3, so the computational complexity of DE-ICA ranges from $O((3^*N/5)^2)=O((3/5^*N)^2)=O(9/25^*N^2)$ to $O((3^*N/2)^2)=O((3/2^*N)^2)=O(9/4^*N^2)$. Therefore, compared with the computational complexity of D-MAENS and ID-MAENS, these three algorithms are similar.

4. Experimental studies

4.1. Test problems and the compared algorithms

In order to evaluate the effectiveness of DE-ICA, a small-scale test set *Beullens* [11], a medium-scale test set *egl* [37–39] and a large-scale test set *EGL-G* [40] are included to compare DE-ICA and D-MAENS [17] and ID-MAENS [18]. For strengthening the generalization ability of the comparison among three algorithms, three algorithms are tested on artificial sets Kshs [41]. Besides, one artificial data set named Art1 with 80 vertexes and 90 tasks is produced. Each algorithm runs 30 times independently.

4.2. Statistical test

The Kruskal-Wallis test (KW-test) [42] is a non-parametric statistical test, which has been used to compare the difference among three algorithms. Here, the KW-test returns a p-value with a significance of 95%. If p is smaller than 5%, we can surely reject the null hypothesis at the 5% significance level, and it means that there is a significant among three compared algorithms.

4.3. Performance metrics

We select indicators to measure the performance of the algorithms for MO-CARP. Three measures are commonly used in the literature [21]: (1) measuring the convergence of the algorithms; (2) measuring the diversity of the non-dominant solutions obtained by algorithms; (3) measuring the convergence and the diversity of the non-dominant solutions. The convergence refers to the proximity between the non-dominant set obtained by the algorithm and the Pareto-optimal set. The diversity means the distribution of the non-dominant solutions. Because the solution space of CARP is discrete, the Pareto-front of the test instances is not evenly distributed. Therefore measuring the diversity of the non-dominant solutions has no practical significance [43]. We adopt three criteria to measure the performance of the algorithms for MO-CARP.

4.3.1. The distance to the reference set (I_D)

The measure standard (I_D) was first proposed in literature [44]. I_D is defined by Eq. (8):

$$ID(X) = \frac{\sum_{j=1}^{N}(\min(d(xi, yj)))}{|S|}, \qquad 1 \le i \le M \tag{8}$$

where, $x_1,\ldots,x_M$ are points in the test set X. $y_1,\ldots,y_N$ are points in the reference set S. $d(x_i, y_j)$ means the Euclidean distance. $I_D(X)$ is the average distance between the points in the reference set S and the closest points in X. The smaller the distance, the closer the test set X to the reference set S. It is difficult for us to obtain the accurate

Pareto-optimal set when solving practical MO-CARP, so we select a new non-dominant set as the reference set *S*. We first merge the non-dominant sets obtained by the three algorithms and then get a new non-dominant set, namely the reference set *S*.

4.3.2. Purity

Purity was first proposed in literature [45] and is defined by Eq. (9).

$$Purity(X) = \frac{|X \cap S|}{|X|} \tag{9}$$

where, X is the non-dominant set obtained by the test algorithm. $|X|$ means the total number of the solutions in X. S is the reference set. $|X \cap S|$ denotes the number of the same solutions in X and S. Purity is a ratio and the greater the value, the better the convergence of the algorithm.

4.3.3. Hypervolume (HV)

HV describes the area in the object space formed by the non-dominant solutions of the test algorithms [46] and is defined by Eq. (10).

$$HV(X) = vol(\cup_{i=1}^{N} xi) \tag{10}$$

There is a reference point during the calculation. We select the point formed by the two maximal objective function values of all the non-dominant solutions obtained by the three algorithms as the reference point. *HV* can also reflect the proximity between the Pareto-front and the obtained non-dominant. The greater the *HV*, the closer the non-dominant set obtained by the test algorithm is to the Pareto-front. If the value of *HV* is zero, it implies that there is only one non-dominant solution.

4.4. Parameter settings

The main parameters in DE-ICA are set as follows: the maximum iteration number G_{max} is set as 200, the probability of local search P_{ls} is 0.1, the clone ratio q is 3. In practical applications, MO-CARP is usually a medium-scale or large-scale problem and algorithms can obtain a large number of solutions. According to the principle in Fig. 2, one of the important influence factors on how to find the solutions which are closer to the Pareto-front is the population size. The influence of initial population size is analyzed in Fig. 5.

In Fig. 5, the test problems *egl-s1-C* is used to test the performance with *popsize* changing from 30 to 180 with an interval of 30. In Fig. 5(a), the purity increases rapidly when *popsize* is less than 120. Then, the rate of increasing becomes slowly. The same situation is also shown in Fig. 5(b), when *popsize* is greater than 120, the rate of decreasing is also slower than before. The expanded scale of the initial population will increase the diversity during the process of evolution. However, if the population size is too large, it will cost more computing resources. Therefore, the initial *popsize* is set at 120 in DE-ICA to achieve better results.

4.5. Simulation results and analysis

Because of the characteristics of MO-CARP, the algorithms will generate a non-dominant set rather than a solution. So it is difficult for us to evaluate the solutions. In practical applications, we usually prefer to greatly optimize the total cost than the cost of the longest circuit. So in the experiments, we compare the non-dominant solutions obtained by the three algorithms. In the following tables, we select one non-dominant solution with the least total cost obtained from the non-dominant set by each algorithm, denoted by optimum. Vertices mean the total number of vertices. Edges mean the total number of edges. Tasks mean the total number of tasks.

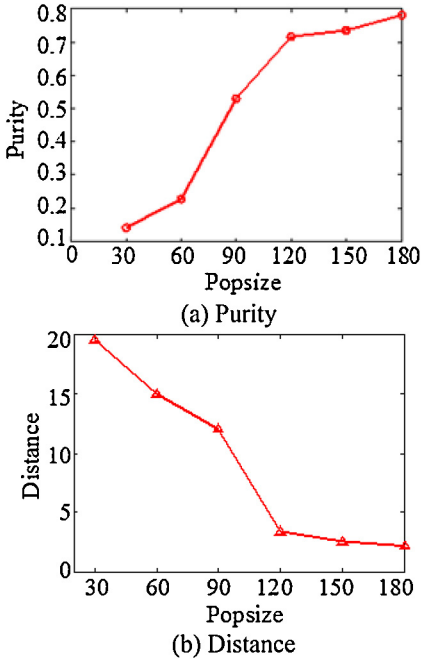

Fig. 5. The purity and distance to the reference set with different *popsize*.

Winner represents the superior algorithm among the three compared algorithms, D is short for D-MAENS, ID is short for ID-MAENS, and DE is short for DE-ICA. (a, b, c) represents the results of non-dominant solutions, in which a is the total cost and b denotes the cost of the longest circuit and c means the violation of capacity. Here, an ideal value (f_a, f_b) is given as a reference. Because all the results in following tables are presented without any violation, we simply list the values of f_a and f_b. Where f_a is the optimal value of total cost so far, which can be obtained from the literatures [11,40,47–50] and f_b denotes the optimal cost of the longest route cost. We use bold fonts to denote the best non-dominant solution.

4.5.1. The comparison of the optima

Table 1 shows the non-dominant solutions obtained by the three algorithms in 30 runs on *Beullens*, *egl*, *EGL* and artificial sets. Because of limited space, we only list some of instances according to a certain interval.

Table 1 shows that DE-ICA performs best on 27 instances of the 58 instances. On these instances, the optima obtained by DE-ICA can completely dominate the optima yielded by D-MAENS and ID-MAENS. In addition, if two solutions of two algorithms cannot dominate each other, then we consider both algorithms to be winners. There are 10 instances, on which DE-ICA and ID-MAENS are both winners. And we can see that there are 21 of 58 instances on which all three algorithms can obtain the same optima. Overall, DE-ICA shows a better performance than D-MAENS and ID-MAENS on these test instances. *Beullens* and Kshs are small-scale test set and the number of the non-dominant solutions for small-scale MO-CARP is limited, so it's comparatively simple to solve. As a result, the solutions obtained by D-MAENS and ID-MAENS are good enough and DE-ICA does not show a significant advantage. Compared with D-MAENS and ID-MAENS on *egl*, Art 1 and EGL-G, DE-ICA can obtain better non-dominant solutions. This is because the search space grows as the scale of the tests increases gradually. The immune clonal algorithm can focus more searching resource on the effective space, and DE-ICA shows an obvious advantage on large scale instances.

Table 1
The comparison of the optima generated by three algorithms.

Name	Vertices	Edges	Tasks	D-MAENS	ID-MAENS	DE-ICA	Ideal value	winner
C01	69	98	79	(4195,655,0)	**(4170,610,0)**	**(4170,610,0)**	(3215,/)	ID¡, DE
C09	76	117	97	(5265,525,0)	**(5260,525,0)**	**(5260,525,0)**	(4120,/)	ID¡, DE
C17	43	56	42	**(3595,640,0)**	**(3585,650,0)**	**(3575,665,0)**	(2620,/)	All
C25	37	50	38	(2310,560,0)	(2310,560,0)	(2310,560,0)	(1815,/)	All
D01	69	98	79	(3235,680,0)	(3235,680,0)	(3235,680,0)	(3215,/)	All
D09	76	117	97	(4120,700,0)	**(4120,695,0)**	**(4120,695,0)**	(4120,/)	ID¡, DE
D17	43	56	42	(2620,710,0)	(2620,710,0)	(2620,710,0)	(2620,/)	All
D25	37	50	38	(1815,760,0)	(1815,760,0)	(1815,760,0)	(1815,/)	All
E01	73	105	85	(4930,620,0)	(4940,605,0)	**(4910,600,0)**	(4045,/)	DE
E09	91	141	103	(5970,590,0)	**(5935,590,0)**	**(5905,600,0)**	(4730,/)	ID, DE
E17	38	50	36	**(2755,670,0)**	**(2755,670,0)**	**(2740,700,0)**	(2055,/)	All
E25	26	35	28	(1615,565,0)	(1615,565,0)	(1615,565,0)	(1615,/)	All
F01	73	105	85	(4045,850,0)	(4045,850,0)	(4045,850,0)	(4040,/)	All
F09	91	141	103	(4810,775,0)	(4810,775,0)	(4810,775,0)	(4730,/)	All
F17	38	50	36	(2055,825,0)	(2055,825,0)	(2055,825,0)	(2055,/)	All
F25	26	35	28	(1390,695,0)	(1390,695,0)	(1390,695,0)	(1390,/)	All
e1-A	77	98	51	(3548,943,0)	(3548,943,0)	(3548,943,0)	(3548,/)	All
e1-B	77	98	51	(4525,839,0)	(4525,839,0)	(4525,839,0)	(4498,/)	All
e1-C	77	98	51	(5621,836,0)	**(5595,836,0)**	**(5595,836,0)**	(5595,/)	ID, DE
e2-A	77	98	72	(5018,953,0)	(5018,953,0)	(5018,953,0)	(5018,/)	All
e2-B	77	98	72	(6347,871,0)	(6347,871,0)	**(6321,870,0)**	(6317,/)	DE
e2-C	77	98	72	(8339,854,0)	(8354,854,0)	**(8335,854,0)**	(8335,/)	DE
e3-A	77	98	87	(5916,942,0)	(5910,999,0)	**(5898,929,0)**	(5898,/)	DE
e3-B	77	98	87	(7801,872,0)	**(7787,872,0)**	**(7787,872,0)**	(7777,/)	ID, DE
e3-C	77	98	87	(10365,827,0)	**(10309,827,0)**	(10311,827,0)	(10292,/)	ID
e4-A	77	98	98	(6491,968,0)	**(6476,929,0)**	**(6473,941,0)**	(6461,/)	ID, DE
e4-B	77	98	98	(9060,853,0)	(9057,926,0)	**(9031,853,0)**	(8975,/)	DE
e4-C	77	98	98	(11764,820,0)	(11699,822,0)	**(11634,820,0)**	(11594,/)	DE
s1-A	140	190	75	(5113,1027,0)	(5073,1023,0)	**(5018,1023,0)**	(5018,/)	DE
s1-B	140	190	75	(6435,984,0)	(6435,984,0)	**(6435,984,0)**	(6388,/)	All
s1-C	140	190	75	(8519,1018,0)	**(8518,1018,0)**	**(8518,1018,0)**	(8518,/)	ID, DE
s2-A	140	190	147	(10134,1061,0)	(10089,1063,0)	**(10040,1058,0)**	(9909,/)	DE
s2-B	140	190	147	(13397,1040,0)	(13365,1040,0)	**(13283,1040,0)**	(13124,/)	DE
s2-C	140	190	147	(16836,1040,0)	(16755,1020,0)	**(16691,1016,0)**	(16425,/)	DE
s3-A	140	190	159	(10516,1077,0)	(10453,1099,0)	**(10402,1040,0)**	(10242,/)	DE
s3-B	140	190	159	(14009,1060,0)	(13956,1040,0)	**(13841,1040,0)**	(13715,/)	DE
s3-C	140	190	159	(17575,1040,0)	(17416,1040,0)	**(17324,1040,0)**	(17216,/)	DE
s4-A	140	190	190	(12616,1080,0)	**(12446,1058,0)**	**(12422,1060,0)**	(12293,/)	ID, DE
s4-B	140	190	190	(16683,1047,0)	(16540,1027,0)	**(16430,1027,0)**	(16262,/)	DE
s4-C	140	190	190	(21363,1027,0)	(21072,1027,0)	**(20964,1027,0)**	(20530,/)	DE
G1-A	255	375	347	(1039145,73909,0)	(1022714,72182,0)	**(1012078,70605,0)**	(1001210,/)	DE
G1-B	255	375	347	(1158572,65050,0)	(1142797,65050,0)	**(1138229,65050,0)**	(1118596,/)	DE
G1-C	255	375	347	(1296432,67327,0)	(1275182,67327,0)	**(1258065,65050,0)**	(1245398,/)	DE
G1-D	255	375	347	(1438323,65050,0)	(1434618,65050,0)	**(1426809,65050,0)**	(1380711,/)	DE
G1-E	255	375	347	(1603107,65050,0)	(1588101,65050,0)	**(1584395,65050,0)**	(1521171,/)	DE
G2-A	255	375	375	(1148660,71074,0)	(1132914,68328,0)	**(1125827,67955,0)**	(1101797,/)	DE
G2-B	255	375	375	(1272569,69858,0)	(1243212,65050,0)	**(1240357,65050,0)**	(1213093,/)	DE
G2-C	255	375	375	(1429768,65050,0)	(1431715,65050,0)	**(1424978,65050,0)**	(1342537,/)	DE
G2-D	255	375	375	(1569517,65050,0)	(1549876,65050,0)	**(1535056,65050,0)**	(1486584,/)	DE
G2-E	255	375	375	(1719771,65050,0)	(1690044,65050,0)	**(1679487,65050,0)**	(1624438,/)	DE
Kshs1	8	15	15	(14661, 4171)	(14661, 4171)	(14661, 4171)	/	All
Kshs2	10	15	15	(9863, 2646)	(9863, 2646)	(9863, 2646)	/	All
Kshs3	6	15	15	(9320, 2670)	(9320, 2670)	(9320, 2670)	/	All
Kshs4	8	15	15	(11498, 3349)	(11498, 3349)	(11498, 3349)	/	All
Kshs5	8	15	15	(10957, 4195)	(10957, 4195)	(10957, 4195)	/	All
Kshs6	8	15	15	(10197, 4032)	(10197, 4032)	(10197, 4032)	/	All
Art1	80	90	90	**(6127, 911)**	(7956, 973)	(7925, 941)	/	DE
Art2	140	70	70	**(4546, 1018)**	(5167, 1050)	(5167, 1050)	/	DE

The values in bold mean "the best results".

4.5.2. The comparison of the non-dominant solutions

Next, we will present the non-dominant solutions of three algorithms. Solutions of each algorithm presented below are selected according to the following rule. At first, one group of non-dominant solutions can be acquired from an independent run. Then after 30 runs, there will be 30 groups of non-dominant solutions. Finally, we put these 30 groups together, from which we select the non-dominant solutions to present. Especially, all the optima in Table 1 will appear in the following figure. Due to limited space, we select some small scale instances and some large instances in Fig. 6 to show the non-dominant solutions.

In Fig. 6, the former two lines of figures are small-scale instances, in which DE-ICA shows the best convergence on 2 instances compared with other two algorithms. Besides, on other 4 instances of small-scale instances, DE-ICA is not worse than D-MAENS and ID-MAENS. In general, DE-ICA shows a slightly advantage on small scale instances. The reason is that small scale instances have a smaller solution space and they are easier to solve. Consequently, other two compared algorithms have enough capacity to deal with it and get a similar performance with DE-ICA.

In the latter two lines of figures, DE-ICA gets the best convergence on 5 instances (e4a, s1a, s4a, G1-A, G2-E). On these instances, DE-ICA can reach areas with both low total cost and the low

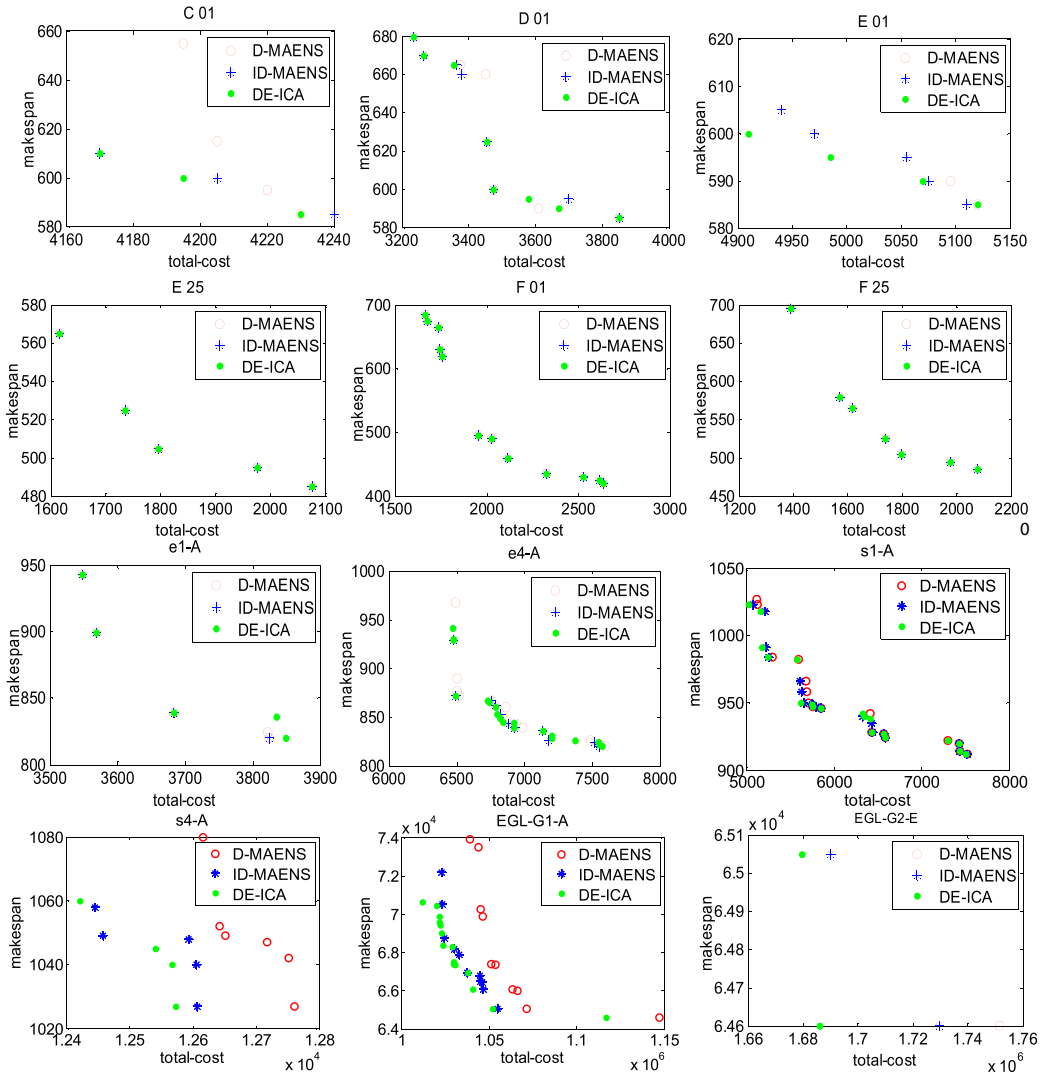

Fig. 6. The non-dominant solutions obtained by three algorithms on some small and large-scale instances.

makespan. As a result, the non-dominant solutions obtained by DE-ICA almost completely dominate the non-dominant solutions obtained by D-MAENS and ID-MAENS. On the remaining instances, DE-ICA is also not worse than the other two algorithms in convergence. In conclusion, DE-ICA demonstrates an obvious advantage in convergence on these large scale instances. The results above show that the DE-ICA is more suitable for the large-scale problems.

4.5.3. *The statistical analysis of performance metrics*

To compare the performance of the three algorithms more objectively, we adopt the three criteria mentioned above to measure the characteristics of three algorithms for MO-CARP. As a result of the limited space, we only list some of instances according to a certain interval. Besides, we also adopt K-W test mentioned above to analysis the algorithms' performance. Table 2–4 show the simulation results on some instances.

From Table 2, as for average value, we can see that DE-ICA performs better than the other two algorithms on 19 of 22 instances. It means that DE-ICA is much closer to the Pareto-front than other two algorithms on these 19 instances. As for the K-W test, it returns a p-value smaller than 0.05 on 9 instances, which indicates there is a significant difference among three algorithms on these instances. When it considers both average value and K-W test, three are 8 instances (C25, D01, D25, F01, F25, E1a, S4a, G2a) on which DE-ICA obtains a significantly better performance than the compared algorithms. Meanwhile, on 1 (G1a) instance DE-ICA gets a significantly worse performance than other algorithms.

From Table 3, as for average value, we can see that DE-ICA performs better than the other two algorithms on 13 of 22 instances. It means that DE-ICA have a better diversity than other two algorithms on these 15 instances. When it refers to the K-W test, it returns a p-value smaller than 0.05 on 7 instances, in which there are 4 instances (D01, F25, E1a, G2e) that DE-ICA is significantly better than other two algorithms. Meanwhile, DE-ICA gets a significantly worse performance than the compared algorithms on 3 instances (D25, S2a, F01). The statistical analysis results show that the difference on most instances is not significant.

In Table 4, DE-ICA performs better on average value than D-MAENS and ID-MAENS on 31 of 32 instances. On these 31 instances, DE-ICA can produce more non-dominate solutions than other two algorithms. On 4 instances (D01, D25, E01, F01) of *Bullens'* set, 13 instances (E1a, E1c, E2a, E3c, E4c, S1a, S1c, S2a, S2c, S3a, S4a) of *egl* set, 5 instances (G1a, G1b, G1e, G2a, G2b) of *EGL-G* set, 2 instances (Kshs5, Art1) of artificial set, the KW-test returns a p-value smaller than 0.05 on these 24 instances, which means that there is a significantly difference among three algorithms. In addition, in these

Table 2
The performance of D-MAENS, ID-MAENS and DE-ICA on I_D.

			C01	C25	D01	D25	E01	E25	F01	F25	E1a	E2a	E3a
Algorithm	DE	Mean	31.70	**4.4**	**45.8**	**5.8**	**92.5**	**0**	**52.1**	**0.5**	11.5	**34.4**	**85.4**
	D	Mean	**21.90**	10.1	53.2	7.9	118.4	1.1	59.4	1	**11.2**	45.5	91.8
	ID	Mean	63.00	6.4	529.7	256.2	224.6	1.4	1190.4	47.2	12.8	42.5	96.4
K-W test			0.365	0.04	0.031	0.001	0.302	0.168	0.001	0.002	0.028	0.526	0.742

			E4a	S1a	S2a	S3a	S4a	G1a	G1e	G2a	G2e	Kshs5	Art1
Algorithm	DE	Mean	**93.9**	**0**	**154.6**	**89.8**	**68.9**	29502	**17841**	**37175**	**24972**	**37175**	**24972**
	D	Mean	113.5	**0**	171	92.3	205.9	62895	32347	48062	38180	48062	38180
	ID	Mean	110.2	**0**	218.1	111.6	108.0	**25346**	21873	59393	30163	59393	30163
K-W test			0.437	1	1	0.38	0.003	0.008	0.185	0.014	0.185	0.119	0.633

The values in bold mean "the best results".

Table 3
The performance of D-MAENS, ID-MAENS and DE-ICA on *HV*.

			C01	C25	D01	D25	E01	E25	F01	F25	E1a	E2a	E3a	E4a
Algorithm	DE	Mean	**133**	7643	**60250**	126803	**4497**	22311	171733	**99778**	**85951**	129957	129066	**85951**
	D	Mean	17	6938	60665	**127975**	2910	34194	**181778**	99525	78862	**142545**	**135890**	78862
	ID	Mean	78	**8106**	1589	23515	1668	**37007**	19604	57115	85813	133738	151214	0
K-W test			0.44	0.215	0.001	0.001	0.178	0.304	0.001	0.003	0.028	0.599	0.433	0.895

			S1a	S2a	S3a	S4a	G1a	G1e	G2a	G2e	Kshs5	Art1
Algorithm	DE	Mean	**355597**	170610	189932	**16399**	**938296822**	**16000902**	**520514180**	**17846311**	**5474203**	94393
	D	Mean	**355597**	149874	**194042**	12150	929365659	11636352	492830781	5165689	5538078	**97346**
	ID	Mean	351881	**176560**	180188	11751	930326759	14233229	446000709	9559093	5706647	74103
K-W test			0.898	0	0.599	0.237	0.23	0.281	0.403	0.033	0.633	0.491

The values in bold mean "the best results".

Table 4
The performance of D-MAENS, ID-MAENS and DE-ICA on *purity*.

			C01	C25	D01	D25	E01	E25	F01	F25	E1a	E1c	E2a	E2c	E3a	E3c	E4a	E4c
Algorithm	DE	Mean	**0.67**	**0.84**	**0.88**	**0.95**	**0.81**	**1**	**0.75**	**0.92**	**0.92**	**0.89**	**0.62**	**0.57**	**0.62**	**0.67**	**0.73**	**0.63**
	D	Mean	0.56	0.59	0.43	0.93	0.41	0.95	0.46	0.91	0.91	0.28	0.29	0.31	0.42	0.06	0.16	0
	ID	Mean	**0.67**	0.71	0.71	0.34	0.3	0.85	0	0.88	0.88	0.06	0.45	0.47	0.34	0.35	0.30	0.39
K-W test			0.887	0.059	0	0.001	0.044	0.356	0	0.967	0.023	0.001	0.027	0.268	0.002	0.003	0.817	0.014

			S1a	S1c	S2a	S2c	S3a	S3c	S4a	S4c	G1a	G1b	G1e	G2a	G2b	G2e	Kshs5	Art1
Algorithm	DE	Mean	**0.62**	**0.59**	**0.73**	**0.63**	**0.78**	**0.87**	**0.73**	**0.44**	0.50	**0.79**	**0.45**	**0.73**	**0.58**	**0.78**	**0.92**	**0.97**
	D	Mean	0.42	0.17	0.16	0.02	0.11	0.11	0.16	0.22	0.03	0.06	0.15	0.02	0	0.11	0.91	0.71
	ID	Mean	0.34	0.55	0.30	0.50	0.28	0.24	0.30	0.33	**0.73**	0.24	0.40	0.33	0.49	0.17	0.88	0.73
K-W test			0.005	0.005	0	0.003	0	0.001	0	0.618	0	0.005	0.007	0.015	0.004	0.061	0.03	0

The values in bold mean "the best results".

24 instances, DE-ICA is significantly better than other compared algorithms on 23 (except the G1a) instances and only one result obtained by DE-ICA on G1a is significantly worse than other algorithms.

In conclusion, from the performance about 3 metrics and the statistical results, the DE-ICA demonstrates a satisfactory performance compared with other two algorithms. On the metric of *HV*, it presents a slightly advantage and shows a certain advantage on I_D. Especially, it demonstrates an absolutely advantage on purity.

4.5.4. Graphic summary for tables of simulation results

To improve the clarity of the comparison among three algorithms, we make a graphic summary for the tables of the simulation results about the times of winners on optimal solution of each algorithm. The results are shown in Fig. 7.

From Fig. 7, we can see that DE-ICA shows a significant advantage on optima when compared with other algorithms on 58 instances. All the experimental results show DE-ICA has an effective performance in solving MO-CARP especially on large scale problems.

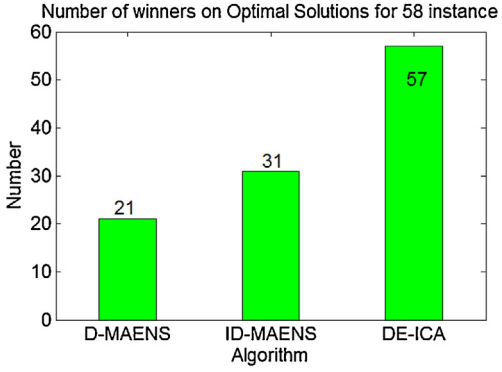

Fig. 7. The statistic of experimental results of three algorithms.

5. Conclusions

This paper has proposed a new algorithm for the popular MO-CARP, comprising an immune clonal algorithm based on directed evolution (DE-ICA). DE-ICA increases the scale of the initial popu-

lation and then performs the clonal operation of the non-dominant solutions. DE-ICA also draws lessons from the effective decomposition operation. This paper also proposes a novel directed comparison operator which can constantly improve the quality of the non-dominant solutions on the basis of increasing the diversity of the population. Simulation results show that DE-ICA is competitive with respect to improving the quality of the non-dominant solutions. Additionally, DE-ICA demonstrates better performance than D-MAENS and ID-MAENS on most of 58 instances, especially for the large-scale problems. Meanwhile, we have demonstrated that DE-ICA has good convergence properties, by observing the distribution of the non-dominant solutions of the three algorithms. In future work, we will study versions of the CARP that closely resemble real-world problems, for example: CARP with time windows and CARP with a variety of vehicle capacities.

Acknowledgements

We would like to express our sincere appreciation to the editors and the anonymous reviewers for their insightful comments, which have greatly helped us in improving the quality of the paper. This work was partially supported by the National Natural Science Foundation of China, under Grant 61371201, the Program for Cheung Kong Scholars and Innovative Research Team in University under Grant IRT1170.

References

[1] L. Feng, Y.S. Ong, M.H. Lim, I.W. Tsang, Memetic search with inter-domain learning: a realization between CVRP and CARP, IEEE Trans. Evol. Comput. 19 (5) (2015) 644–658.
[2] X.Z. Wen, L. Shao, Y. Xue, W. Fang, A rapid learning algorithm for vehicle classification, Inf. Sci. 295 (1) (2015) 395–406.
[3] J. Shen, H.W. Tan, J. Wang, J.W. Wang, S. Lee, A novel routing protocol providing good transmission reliability in underwater sensor networks, J. Internet Technol. 16 (1) (2015) 171–178.
[4] L. Grandinetti, F. Guerriero, D. Lagan'a, O. Pisacane, An approximate ε-constraint method for the multi-objective undirected capacitated arc routing problem, in: P. Festa (Ed.), Proceedings of the 9th International Symposium on Experimental Algorithms, Lecture Notes in Computer Science, SEA'10, vol. 6049, Springer Verlag, 2010, pp. 214–225.
[5] B.L. Golden, J.S. DeArmon, E.K. Baker, Computational experiments with algorithms for a class of routing problems, Comput. Oper. Res. 10 (1) (1983) 47–59.
[6] B.L. Golden, R.T. Wong, Capacitated arc routing problems, Networks 11 (3) (1981) 305–315.
[7] G. Ulusoy, The fleet size and mix problem for capacitated arc routing, Eur. J. Oper. Res. 22 (3) (1985) 329–337.
[8] G. Ghiania, F. Guerrierob, G. Laportec, Real-time vehicle routing: solution concepts, algorithms and parallel computing strategies, Eur. J. Oper. Res. 151 (1) (2003) 1–11.
[9] R.W. Eglese, Routeing winter gritting vehicles, Discrete Appl. Math. 48 (3) (1994) 231–244.
[10] A. Hertz, G. Laporte, M. Mittaz, A tabu search heuristic for the capacitated arc routing problem, Oper. Res. 48 (1) (2000) 129–135.
[11] P. Beullens, L. Muyldermans, D. Cattrysse, A guided local search heuristic for the capacitated arc routing problem, Eur. J. Oper. Res. 147 (3) (2003) 629–643.
[12] P. Lacomme, C. Prins, W. Ramdane_Cherif, Competitive memetic algorithms for arc routing Problems, Ann. Oper. Res. 131 (1) (2004) 159–185.
[13] K. Tang, Y. Mei, X. Yao, Memetic algorithm with extended neighborhood search for capacitated arc routing problems, IEEE Trans. Evol. Comput. 13 (5) (2009) 1151–1166.
[14] Y. Mei, X.D. Li, X. Yao, Cooperative co-evolution with route distance grouping for large-scale capacitated arc routing problems, IEEE Trans. Evol. Comput. 18 (3) (2014) 435–449.
[15] P. Lacomme, C. Prins, M. Sevaux, A genetical algorithm for a bi-objective capacitated arc routing problem, Comput. Oper. Res. 33 (12) (2006) 3473–3493.
[16] L. Grandinetti, F. Guerriero, D. Lagana, O. Pisacane, An optimization-based heuristic for the multi-objective undirected capacitated arc routing problem, Comput. Oper. Res. 39 (2012) 2300–2309.
[17] Y. Mei, K. Tang, X. Yao, Decomposition-based memetic algorithm for multiobjective capacitated arc routing problems, IEEE Trans. Evol. Comput. 15 (2) (2011) 151–165.
[18] R.H. Shang, J. Wang, L.C. Jiao, An improved decomposition- based memetic algorithm for multi-objective capacitated arc routing problem, Appl. Soft Comput. 19 (2014) 343–361.
[19] R.H. Shang, Y.Y. Wang, J. Wang, A multi-population cooperative coevolutionary algorithm for multi-objective capacitated arc routing problem, Inf. Sci. 277 (2014) 609–642.
[20] K. Deb, S. Agrawal, A. Pratap, A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[21] S.K. Mandal, D. Pacciarelli, A. Løkketangen, G. Hasle, A memetic NSGA-II for the bi-objective mixed capacitated general routing problem, J. Heuristics 21 (3) (2015) 359–390.
[22] K.C. Tan, E.F. Khor, T.H. Lee, Multiobjective Evolutionary Algorithms and Applications, Springer-Verlag, United Kingdom, 2005.
[23] X.N. Shen, X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Inf. Sci. 298 (3) (2015) 198–224.
[24] K.C. Tan, Y.J. Yang, C.K. Goh, A distributed cooperative coevolutionary algorithm for multiobjective optimization, IEEE Trans. Evol. Comput. 10 (5) (2006) 527–549.
[25] R.H. Shang, L.C. Jiao, F. Liu, A novel immune clonal algorithm for MO problems, IEEE Trans. Evol. Comput. 16 (1) (2012) 35–49.
[26] S.K. Goh, H.A. Abbass, K.C. Tan, Decompositional independent component analysis using multi-objective optimization, Soft Comput. 20 (4) (2016) 1289–1304.
[27] F. Guerriero, R. Musmanno, Label correcting methods to solve multicriteria shortest path problems, J. Optim. Theory Appl. 111 (3) (2001) 589–613.
[28] Y.Y. Li, H.Y. He, Y. Wang, X. Xu, L.C. Jiao, An improved multiobjective estimation of distribution algorithm for environmental economic dispatch of hydrothermal power systems, Appl. Soft Comput. 28 (2015) 559–568.
[29] S. Jiang, Y.S. Ong, J. Zhang, L. Feng, Consistencies or contradictions of performance metrics in multiobjective optimization, IEEE Trans. Cybern. 44 (12) (2014) 239–2404.
[30] J. Zhang, Q. Zhou, Study on the optimization of logistics distribution VRP based on immune clone algorithm, J. Hunan Univ.: Nat. Sci. 31 (5) (2004) 54–58.
[31] R.H. Shang, H.N. Ma, J. Wang, Immune clonal selection algorithm for capacitated arc routing problem, Soft Comput. (2015) 1–28.
[32] L.C. Jiao, Y.Y. Li, M.G. Gong, X.R. Zhang, Quantum-inspired immune clonal algorithm for global optimization, IEEE Trans. Syst. Man Cybern.: Part B 38 (5) (2008) 1234–1253.
[33] L.C. Jiao, L. Wang, A novel genetic algorithm based on immunity, IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 30 (5) (2000) 552–561.
[34] M. Potter, K.D. Jong, A cooperative coevolutionary approach to function optimization, Parallel Prob. Solving Nat. Lect. Notes Comput. Sci. (1994) 249–257.
[35] Q. Zhang, H. Li, MOEA/D: a multi-objective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[36] T.P. Runarsson, X. Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Trans. Evol. Comput. 4 (3) (2000) 284–294.
[37] R. Eglese, L. Li, A tabu search based heuristic for arc routing with a capacity constraint and time deadline, Meta-Heuristics: Theory Appl. (1996) 633–650.
[38] L. Li, R. Eglese, An interactive algorithm for vehicle routing for winter-gritting, J. Oper. Res. Soc. 47 (2) (1996) 217–228.
[39] R.W. Eglese, Routeing winter gritting vehicles, Discrete Appl. Math. 48 (3) (1994) 231–244.
[40] J. Brand˜ao, R. Eglese, A deterministic tabu search algorithm for the capacitated arc routing problem, Comput. Oper. Res. 35 (4) (2008) 1112–1126.
[41] M. Kiuchi, Y. Shinano, R. Hirabayashi, Y. Saruwatari, An exact algorithm for the capacitated arc routing problem using parallel branch and bound method, Abstracts of the 1995 Spring National Conference of the Operational Research Society of Japan (1995) 28–29.
[42] W. Kruskal, W. Wallis, Use of ranks in one-criterion variance analysis, J. Am. Stat. Assoc. 47 (260) (1952) 583–621.
[43] G.B. Danizig, J.H. Ramser, The truck dispatching problem, Manag. Sci. 6 (1) (1959) 80–91.
[44] P. Czyzak, A. Jaszkiewicz, Pareto simulated annealing a metaheuristic technique for multiple-objective combinatorial optimization, J. Multicrit. Decis. Anal. 7 (1998) 34–47.
[45] S. Bandyopadhyay, S. Pal, B. Aruna, Multiobjective GAs, quantitative indices, and pattern classification, IEEE Trans. Syst. Man Cybern.: Part B 4 (5) (2004) 2088–2099.
[46] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: improving the strength Pareto evolutionary algorithm, EUROGEN 9 (2001) 5–100.
[47] J.M. Belenguer, E. Benavent, A cutting plane algorithm for the capacitated arc routing problem, Comput. Oper. Res. 30 (5) (2003) 705–728.
[48] D. Ahr, Contributions to Multiple Postmen Problems. Ph.D. Thesis, Rupercht −Karls-Universitat, Heidelberg, Germany, 2004.
[49] R. Baldacci, V. Maniezzo, Exact methods based on node routing formulations for arc routing problems, Networks 47 (2006) 52–60.
[50] H. Longo, M.P. De Aragao, E. Uchoa, Solving capacitated arc routing problems using a transformation to the CARP, Comput. Oper. Res. 33 (6) (2006) 1823–1837.