Expert Systems With Applications 71 (2017) 57–68

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems

Peng Guo[a], Wenming Cheng[a], Yi Wang[b,*]

[a] School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China
[b] Department of Mathematics and Computer Science, Auburn University at Montgomery, P.O. Box 244023, Montgomery, AL 36124-4023, USA

ARTICLE INFO

Article history:
Received 11 March 2016
Revised 1 November 2016
Accepted 19 November 2016
Available online 26 November 2016

Keywords:
Facility location
Evolutionary algorithm
Fitness approximation
Local search
Extreme machine learning

ABSTRACT

This paper considers the two-stage capacitated facility location problem (TSCFLP) in which products manufactured in plants are delivered to customers via storage depots. Customer demands are satisfied subject to limited plant production and limited depot storage capacity. The objective is to determine the locations of plants and depots in order to minimize the total cost including the fixed cost and transportation cost. However, the problem is known to be NP-hard. A practicable exact algorithm is impossible to be developed. In order to solve large-sized problems encountered in the practical decision process, an efficient alternative approximate method becomes more valuable. This paper aims to propose a hybrid evolutionary algorithm framework with machine learning fitness approximation for delivering better solutions in a reasonable amount of computational time. In our study, genetic operators are adopted to perform the search process and a local search strategy is used to refine the best solution found in the population. To avoid the expensive consumption of computational time during the fitness evaluating process, the framework uses extreme machine learning to approximate the fitness of most individuals. Moreover, two heuristics based on the characteristics of the problem is incorporated to generate a good initial population. Computational experiments are performed on two sets of test instances from the recent literature. The performance of the proposed algorithm is evaluated and analyzed. Compared with other algorithms in the literature, the proposed algorithm can find the optimal or near-optimal solutions in a reasonable amount of computational time. By employing the proposed algorithm, facilities can be positioned more efficiently, which means the fixed cost and the transportation cost can be decreased significantly, and organizations can enhance competitiveness by using the optimized facility location scheme.

1. Introduction

As a strategic issue in the design of supply chain networks, logistics facility location problems have been studied extensively in the past few decades, for example, see Mesa and Boffey (1996), Owen and Daskin (1998), Snyder (2006), Sahin and Sural (2007) and Melo, Nickel, and da Gama (2009). In a facility location problem, management decision makers have to decide which sites should be chosen to establish new facilities from a set of available candidate sites, while constraints are met in order to minimize the total cost. The constraints are such that the demands of all customers have to be met, the capacity limits of the suppliers and facilities must not be violated, etc. The cost includes fixed costs to open plants and depots, and variable costs associated with transportation. The decision about the facility location leads to long term commitments due to the megacephalic fixed costs of these facilities. The selection of facility location will profoundly influence the management planning of organizations, especially in the relation between the sectors and theirs customers.

Owing to the importance of the facility location problem, it has been widely considered in the literature. In particular, single-stage capacitated facility location problems (CFLP) have been successfully analyzed (Klose & Drexl, 2005; Zhang, Chen, & Ye, 2005). Subsequently, many effective search algorithms have been designed for their near-optimal and/or optimal solutions of large-sized instances, including heuristics based on Lagrangian relaxation (Avella, Boccia, Sforza, & Vasil'ev, 2008), kernel search (Guastaroba & Speranza, 2014; 2012) and cut-and-solve algorithm (Yang, Chu, & Chen, 2012). Moreover, meta-heuristics, such as tabu search (TS) (Sadigh, Mozafari, & Kashan, 2010; Sun, 2011), simulated annealing,

* Corresponding author.
E-mail addresses: pengguo318@gmail.com (P. Guo), wmcheng@swjtu.edu.cn (W. Cheng), ywang2@aum.edu (Y. Wang).

http://dx.doi.org/10.1016/j.eswa.2016.11.025
0957-4174/© 2016 Elsevier Ltd. All rights reserved.

genetic algorithm (Arostegui, Kadipasaoglu, & Khumawala, 2006) and hybrid firefly-genetic algorithm (HFGA) (Rahmani & MirHassani, 2014) were developed to solve the CFLP.

Recently, the two-stage capacitated facility location problem (TSCFLP) has attracted researchers' attention, as in many situations more than one type of facilities are considered simultaneously. Due to the intractability of the TSCFLP, it is hard to obtain optimal solutions for a large-sized instance. Therefore, some researchers focused on heuristics rather than mathematical modeling in solving the TSCFLP. Klose (1999) proposed a linear programming based heuristic for the TSCFLP with single source constraints, and evaluated the proposed algorithm on a large set of test problems with up to 10 plants, 50 potential depots and 500 customers. They presented a Lagrangian relax-and-cut procedure for the problem, and found the proposed algorithm could yield very good solutions in the expense of consuming a lot of effort to optimize or re-optimize the Lagrangian dual (Klose, 2000).

In order to solve large sized instances of the TSCFLP, Fernandes et al. (2014) suggested a simple genetic algorithm (GA) to determine which plants and depots should be opened, and obtained the flow values between plants and depots and between depots and customers by solving a minimum cost flow problem. In addition, the two-stage uncapacitated facility location problem was considered by Marín (2007), and its lower bounds were delivered by a relaxation formulation. Wang and Watada (2012) studied a TSCFLP with fuzzy costs and demands, and developed a particle swarm optimization to solve the problem under consideration.

Although several heuristics have been developed to solve the TSCFLP, most methods are not fit for solving large-sized problems in terms of solution quality and computational time. For examples, the computational time of the GA varies from 236 s to 2784 s for the problem instances with 50 plants and 100 plants in Fernandes et al. (2014), and each function evaluation involves solving a minimum cost flow problem which is computationally expensive.

For most evolutionary algorithms (EAs) like the GA, a very large number of fitness function evaluations are required before a well acceptable solution can be delivered. This characteristic of the EAs seriously limits their applications in solving intractable high-dimensional and multimodal optimization problems. One promising way to significantly ease the computational burden of the EAs is to adopt computationally cheap surrogate models instead of computationally expensive fitness evaluations (Jin, 2005). Various techniques for the construction of surrogate models (also called approximation models or meta-models) have been employed to obtain the efficient and effective hybrid EAs. Among these techniques, artificial neural network (ANN), support vector machine (SVM) and kriging models are among some of the most prominent and commonly used techniques (Dias, Rocha, Ferreira, & do Carmo Lopes, 2014; Hacioglu, 2007; Zhang, Liu, Tsang, & Virginas, 2010; Zheng, Chen, Liu, & Huang, 2016). By elaborating surrogate models, the computational burden can be greatly reduced. This is due to the efforts in constructing the surrogate models and then using it to predict fitness values are much lower than the efforts of directly calculating the fitness functions by the standard approach. Inspired by the application of surrogate models in solving complex continuous optimization problems, *it is intended in this paper to adapt a meta-heuristic algorithm with fitness approximation to solve the TSCFLP, in order to reduce the computational time and guarantee correct convergence.*

The extreme learning machine (ELM) developed by Huang, Zhu, and Siew (2006) strikes a balance between speed and generalization performance, and attracts more and more attention from various respects. Compared with the ANN, the SVM and other traditional forecasting models, the ELM model retains the advantages of fast learning, good ability to generalize and convenience in terms of modeling. *In this paper, a hybrid evolutionary algorithm with fitness approximation (HEA/FA) is proposed to obtain the optimal or neat-optimal solutions for the TSCFLP.* In the framework of the algorithm, the extreme machine learning is used as surrogate model to approximate fitness values of most individuals in the population. *The contribution of this paper is to introduce the basic structure of the evolutionary algorithm with the integration of the ELM fitness approximation when solving a discrete optimization problem.* The proposed algorithm uses the genetic operations (selection, crossover and mutation) as well as restarting strategy and a special local search operation to update the current population. Computational results and comparison to other meta-heuristic algorithms proposed in the literature demonstrate the effectiveness and efficiency of the proposed HEA/FA approach.

The remainder of this paper is organized as follows. In Section 2, the TSCFLP is described in details and formulated as a mixed integer programming model. In Section 3, the ELM algorithm is briefly introduced for the convenience of readers. Section 4 presents the hybrid evolutionary framework of optimizing the TSCFLP using fitness function approximation via the ELM. Subsequently, the numerical tests and comparisons are carried out in Section 5. Section 6 discusses the strength and the weakness of the proposed algorithm. Finally Section 7 summarizes some conclusions and points out future research.

2. Problem formulation

The problem under study is defined as follows: products are produced at plants and then transported to depots, while both plants and depots have limited capacities. From depots product are delivered to customers to satisfy their demands. The use of plants/depots is accompanied by a fixed cost, while transportation from plants to customers via depots results in a variable cost. The two-stage facility location problem aims to identify what plants and depots to use, as well as the product flows from the plants to the depots and then to the customers, such that the demands are met and the total cost is minimized. The problem under consideration in this paper is the same as that in the Ref. (Litvinchev & Ozuna Espinosa, 2012).

The parameters of this problem under study are defined as follows. Let I, J, K be the sets of plants, depots and customers, respectively. Let f_i be the fixed cost for plant $i \in I$; b_i be the capacity of plant $i \in I$; g_j be the fixed cost associated with depot $j \in J$; p_j be the capacity of depot $j \in J$; c_{ij} be the cost of transporting one unit of products from plant $i \in I$ to depot $j \in J$; d_{jk} be the cost of transporting one unit of the product from depot $j \in J$ to customer $k \in K$; q_k be the demand of customer $k \in K$.

Let binary variables y_i, $i \in I$, be equal to 1 if and only if plant i is chosen to be opened. Similarly, let z_j be a binary variable which is equal to 1 if and only if depot $j \in J$ is opened, otherwise it is equal to 0. And let real variables x_{ij} and s_{jk} define the flow of products from plants to customers via depots. Using the above notations, the TSLFLP can be formulated as the following integer programming model:

$$\text{Minimize} \quad Z = \sum_{i\in I} f_i y_i + \sum_{j\in J} g_j z_j + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{j\in J}\sum_{k\in K} d_{jk}s_{jk} \tag{2.1}$$

subject to

$$\sum_{j\in J} s_{jk} \geq q_k \qquad \forall\ k \in K \tag{2.2}$$

$$\sum_{i\in I} x_{ij} \geq \sum_{k\in K} s_{jk} \qquad \forall\ j \in J \tag{2.3}$$

$$\sum_{j\in J} x_{ij} \le b_i y_i \qquad \forall\ i \in I \tag{2.4}$$

$$\sum_{k\in K} s_{jk} \le p_j z_j \qquad \forall\ j \in J \tag{2.5}$$

$$x_{ij} \le b_i z_j \qquad \forall i \in I,\ j \in J \tag{2.6}$$

$$x_{ij},\ s_{jk} \ge 0, y_i, z_j \in \{0,1\}, \qquad \forall\ i \in I,\ j \in J,\ k \in K \tag{2.7}$$

The objective function (2.1) includes the opening costs of plants and depots and the transportation costs in the two-stages. Constraints (2.2) ensure the demand from each customer must be met. Constraints (2.3) guarantee that the total amount of products transported from a depot must be at most the total transported to it from the plants. Constraints (2.4) and (2.5) gives the capacity limits for plants and depots and assure the flow only from plants and depots opened. Constraints (2.6) determine the upper bounds of the decision variables x_{ij}. At last, constraints (2.7) represent the boundary values of the decision variables.

The model considered in this paper is a mixed integer linear programming (MILP) model and is characterized by binary variables y_i, z_j , and continuous variables x_{ij}, s_{jk}. Let $y = (y_i : i \in I)$, $z = (z_j : j \in J)$, $x = (x_{ij} : i \in I, j \in J)$ and $s = (s_{jk} : j \in J, k \in K)$. A solution to the TSCFLP problem shall be denoted by $\mathcal{X} = \{y, z, x, s\}$.

The capacitated facility location problem (CFLP) is equivalent to the second stage of the TSCFLP, so it can be looked as a special case of the TSCFLP. Since the CFLP was proved to be NP-hard by Cornuéjols, Nemhauser, and Wolsey (1990), the TSCFLP under consideration is also NP-hard. Since polynomial time algorithms are both theoretically and practically impossible for the large-sized TSCFLP, it is necessary to develop heuristic or meta-heuristic approaches. In order to accelerate the solution speed, a hybrid evolutionary algorithm with fitness approximation (HEA/FA) based on the extreme learning machine is developed to obtain near-optimal solutions in this paper.

3. Extreme learning machine

The extreme learning machine (ELM), which was proposed by Huang et al. (2006), is a relatively new learning algorithm of single hidden-layer feed-forward neural networks (SLFNs). It selects hidden nodes at random and analytically determines the output weights of the SLFNs. Unlike conventional gradient-based learning methods, the ELM does not need to tune the parameters between the input layer and the hidden layer iteratively. All hidden node parameters are assigned randomly, which are independent of the training data. Different from the traditional learning algorithms for neural networks the ELM not only attempts to reach the smallest training error but also the smallest norm of output weights.

Let n, h and m denote the node numbers of the input layer, the hidden layer and the output layer, respectively. Let R denote the set of real numbers and R^n denote the n-dimensional real space. Let A^T be the transpose of a matrix A. Suppose there are N arbitrary distinct training samples $(\mathbf{x}_j, \mathbf{t}_j) \in R^n \times R^m$, where $\mathbf{x}_j$ is an input vector in R^n and $\mathbf{t}_j$ is a target vector in R^m, and $1 \le j \le N$. The SLFNs with h hidden nodes and activation function $g(\cdot)$ are mathematically modeled as:

$$\sum_{i=1}^{h} \beta_i g(\mathbf{w}_i^T \mathbf{x}_j + b_i) + \beta_0 = \mathbf{o}_j, \quad j = 1, \ldots, N \tag{3.1}$$

where $\mathbf{w}_i = [\omega_{i1}, \omega_{i2}, \ldots, \omega_{in}]^T$, $1 \le i \le h$, is the weight vector connecting the ith hidden node and the input node $\mathbf{x}_j$; $\beta_i = [\beta_{i1}, \beta_{i2}, \ldots, \beta_{im}]^T$ is the weight vector connecting the ith hidden node and the output nodes, $b_i \in R$ is the threshold of the ith hidden node, and β_0 is a bias parameter vector in R^m. Moreover, $\mathbf{o}_j = [o_{j1}, o_{j2}, \ldots, o_{jm}]^T$, $1 \le j \le N$, is the output vector of the SLFNs.

That the SLFNs with h hidden nodes with activation function $g(\cdot)$ can approximate the N samples with zero error means that $\sum_{j=1}^{N} \| \mathbf{o}_j - \mathbf{t}_j \| = 0$ for a chosen norm, i.e., there exist $\beta_i, \beta_0 \in R^m, \mathbf{w}_i \in R^n$, $b_i \in R$, such that

$$\sum_{i=1}^{h} \beta_i g(\mathbf{w}_i^T \mathbf{x}_j + b_i) + \beta_0 = \mathbf{t}_j, \quad j = 1, \ldots, N. \tag{3.2}$$

The above N equations can be rewritten compactly as

$$\mathbf{H}\beta = \mathbf{T} \tag{3.3}$$

where $\mathbf{H} \in R^{N\times(h+1)}$ is the hidden-layer output matrix of the ELM

$$\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_h, b_1, \ldots, b_h, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} 1 & g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_h \cdot \mathbf{x}_1 + b_h) \\ \vdots & \cdots & \vdots & \\ 1 & g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_h \cdot \mathbf{x}_N + b_h) \end{bmatrix}_{N\times(h+1)} \tag{3.4}$$

$\beta = [\beta_0, \beta_1, \beta_2, \ldots, \beta_h]^T \in R^{(h+1)\times m}$ denotes the *matrix* of output weights, and $\mathbf{T} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N]^T \in R^{N\times m}$ denote the target matrix of training data. As described in Huang et al. (2006), the ith column of $\mathbf{H}$ is the ith hidden node output with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. When the randomly selected input weights $\mathbf{w}_i$ and the hidden layer thresholds b_i, $1 \le i \le h$, are given, the network training is to find a least squares solution β of the linear system. The least norm least-squares solution of the above linear system is

$$\beta^* = \mathbf{H}^\dagger \mathbf{T} \tag{3.5}$$

where $\mathbf{H}^\dagger$is the Moore–Penrose pseudo inverse of the hidden layer output matrix $\mathbf{H}$. Thus, the procedure of ELM can be described in three main steps as follows:

Algorithm ELM: Given a training set $S = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \ldots, N\}$, activation function $g(\cdot)$, and number of hidden nodes h,

1. Generate the weight vector $\mathbf{w}_i$ and bias b_i, $i = 1, \ldots, h$ at random.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Estimate the output weight $\beta^* = \mathbf{H}^\dagger \mathbf{T}$.

Since the ELM was developed, a number of researchers have already applied the ELM to many application fields, such as Medical/Biomedical, computer vision, image/video processing, chemical process, fault detection and diagnosis, and so on (Huang, Huang, Song, & You, 2015). Compared with other state-of-the-art learning algorithms, including support vector machine (SVM) and deep learning, the ELM algorithm is fast and stable in training, easy in implementation, and accurate in modeling and prediction (Huang, Wang, & Lan, 2011).

4. Hybrid evolutionary algorithm with fitness approximation for TSCFLP

In this section, we propose a hybrid evolutionary algorithm (HEA) specialized for the TSCFLP in detail. Besides incorporating successful elements of the previously constructed effective heuristic algorithms (Fernandes et al., 2014), the proposed HEA gains pretty good balance between quality of solutions and computational time in two respects: (1) incorporating s local search strategy to enhance the quality of the elitist individuals, and (2) introducing approximating fitness evaluation approach based on the

ELM to alleviate the computational burden of the HEA. The idea of introducing fitness approximation has been proven to be effective and efficient in solving complex continuous optimization problems (Hacioglu, 2007; Jin, Olhofer, & Sendhoff, 2002; Zheng et al., 2016).

4.1. Framework of the proposed algorithm

In this paper, the generic algorithm (GA) is employed to work as the search framework of the evolutionary computation algorithm owing to its success application to solving the CFLP (Arostegui et al., 2006; Correa, Steiner, Freitas, & Carnieri, 2004; Jaramillo, Bhadury, & Batta, 2002). The GA was inspired by biological mechanisms such as the genetic inheritance laws of Mendel and the natural selection concept of Charles Darwin, where the best adapted species survive (Holland, 1973).

In order to use a GA to solve a specific problem, a suitable representation (or encoding) is firstly considered. The candidate solution of the problem are encoded by a set of parameters and referred to as chromosomes or individuals. The word ‘individual’ and the word ‘solution’ are used interchangeably herein. A fitness value is directly calculated by the objective function of the problem under consideration for each individual. At each iteration, the GA evolves the current population of individuals into a new population through selection, crossover and mutation processes. Only the fittest individuals are selected from the current population and retained in next iteration (Gen & Cheng, 2000). Regarding the stopping criterion, two commonly used termination conditions are used in the proposed algorithm herein. If the number of iterations t exceeds the maximum number of iterations $t_{\max}$ or the best solution is not improved upon in t_{nip} successive iterations, the algorithm is terminated; otherwise, a new iteration is started and the iteration timer t is increased by 1.

The proposed algorithm can be separated into six main parts: (1) initialization operation, (2) selection operation, (3) crossover operation, (4) mutation operation, (5) fitness approximation operation, and (6) local search operation. The procedure of the HEA with fitness approximation (HEA/FA) is shown in Algorithm 1. The details of the proposed algorithm presented for the TSCFLP are elaborated as follows.

4.2. Encoding and population initialization

The encoding of individuals used in our implementation is the same as that of many studies (Fernandes et al., 2014; Rahmani & MirHassani, 2014). Once the binary decision variables $y = (y_i : i \in I)$ and $z = (z_j : j \in J)$ of the MILP model proposed in Section 2 are determined, the MILP model is simplified into a linear programming (LP) model with the flow variables $x = (x_{ij} : i \in I, j \in J)$ and $s = (s_{jk} : j \in J, k \in K)$. The simplified LP model can be easily handled by various commercial linear programming softwares. In this paper, Gurobi v6.5 is adopted to obtain the solution of the LP model optimally. The real variable x_{ij} is limited in the interval $[0, b_i], i = 1, \ldots, I$ for each $j \in J$, and the values of the variable s_{jk} lie in the interval $[0, p_j], j = 1, \ldots, J$ for each $k \in K$. The LP model is thus given by (2.1)–(2.5). In the population of individuals of the HEA/FA algorithm, each individual contains $|I| + |J|$ binary elements, which refer to all plants and depots in the TSCFLP. The binary variables for one individual shall be collectively denoted by $X = (y, z)$. The values of these elements indicate whether corresponding facility would be selected. The value of 1 shows the corresponding plant (or depot) is selected to open, otherwise it is closed.

Generally, the initial population of individuals is filled with N_p randomly generated binary vectors. In order to guarantee an initial population with certain quality and diversity, two heuristics are used to produce two individuals with special property. The remaining individuals are initialized with random binary vectors.

For the first heuristic, a cost–benefit index constructed by Fernandes et al. (2014) is used to decide which facility to open. Both fixed and transportation costs are integrated to form the priority index for each plant or depot. The index is defined as the ratio of the sum of a facility’s fixed cost and its related transportation costs to its capacity. For plant $i (i \in I)$, its cost–benefit index is stated by $(f_i + \sum_{j\in J} c_{ij})/b_i$. Once the plants to open are determined, the index of a depot $j (j \in J)$ is delivered by $(\sum_{i\in I} c_{ij} + g_j + \sum_{k\in K} d_{jk})/p_j$.

In this paper, facilities with smaller index are selected sequentially. Based on the above description, the heuristic is called cost–benefit ranking (CBR) algorithm, and its procedure is described in Algorithm 2.

In the second heuristic, a rounding method for generating an initial individual of the population is presented. Firstly, the optimal solution of the linear programming model (2.1)–(2.7) with continuous variables y and z in the interval [0, 1] can be obtained by using Gurobi easily. Then the rounding operation is performed on the binary variables y_i, $i \in I$ and z_j, $j \in J$ in the optimal solution. If y_i (or z_j) is less than 0.5, it is set to 0; otherwise, it is set to 1. Once the rounding operation is done for each element in the solution, one individual with binary values can be obtained. The individual is regarded as the output of the heuristic.

However, the individuals delivered by the above heuristics may be infeasible owing to the capacity constraints of plants and depots, i.e. $\Sigma_{i \in I} b_i y_i \geq \Sigma_{k \in K} q_k$ and $\Sigma_{j \in J} p_j z_j \geq \Sigma_{k \in K} q_k$, may not be met. In our heuristic algorithm, an approach with the cost–benefit index is designed to correct the infeasible individuals obtained from the rounding heuristic. The approach is based on the observation in Theorem 4.1.

Theorem 4.1. *In any optimal solution for the TSCFLP, the following equality:*

$$\sum_{i\in I}\sum_{j\in J} x_{ij} = \sum_{j\in J}\sum_{k\in K} s_{jk} = \sum_{k\in K} q_k \tag{4.1}$$

must be met.

Proof. The second equality in (4.1) is considered first. Recalling the constraints (2.2), an optimal solution must meet the following equality

$$\sum_{j\in J} s_{jk} = q_k + \mu_k, \qquad k \in K,$$

where $\mu_k \geq 0$, $k \in K$ are slack variables. From the above equation and noting q_k, $k \in K$ are constants, it is clear that only if each $\mu_k = 0$, $k \in K$, the cost term $\Sigma_{j \in J}\Sigma_{k \in K} d_{jk} s_{jk}$ appearing in the objective function (2.1) is minimal. Thus by setting $\mu_k = 0$, $k \in K$, the second equality $\sum_{j\in J}\sum_{k\in K} s_{jk} = \sum_{k\in K} q_k$ is obtained.

In a very similar spirit, the first equality is proved. By constraints (2.3) and the above argument, an optimal solution must meet the following:

$$\sum_{i\in I}\sum_{j\in J} x_{ij} \geq \sum_{j\in J}\sum_{k\in K} s_{jk} = \sum_{k\in K} q_k.$$

Constraints (2.3) also implies that

$$\sum_{i\in I} x_{ij} = \sum_{k\in K} s_{jk} + \tilde{\mu}_j, \qquad j \in J,$$

where $\tilde{\mu}_j \geq 0$, $j \in J$ are slack variables. This makes it clear that only when each $\tilde{\mu}_j = 0$, $j \in J$, can the cost term $\Sigma_{i \in I}\Sigma_{j \in J} c_{ij} x_{ij}$ be minimal. Of course, in turn, this implies the first equality in Eq. (4.1). The theorem has been proved. □

Algorithm 1 The hybrid evolutionary algorithm with fitness approximation.

```
1: Set the value of the population size $N_p$, the maximum number of iterations $t_{\max}$, the maximum number of consecutive non-improvement iterations $t^*_{\text{nip}}$;
2: Generate a population that consists of 2 individuals and $2N_p - 2$ randomly generated individuals. The first two individuals are obtained from the heuristics CBR (see Algorithm 2) and a rounding heuristics. Further, the algorithm MIH (see Algorithm 3) is applied to correct and improve each individual to guarantee its feasibility.
3: Evaluate all $2N_p$ individuals using the exact fitness function.Use these $2N_p$ individuals and their fitness values to form a training set $S$. Train the ELM model using $S$.Select the first $N_p$ individuals that have smaller fitness values from the $2N_p$ individuals to form the initial population $P$;
4: Set $t = 1$ and $t_{\text{nip}} = 0$;
5: while the stopping criterion is not met do
6:   for $i := 1$ to $N_p$ do
7:     Select two individuals $X_\imath$, $X_\jmath$ from population $P$ randomly; /% Selection Operation I %/
8:     Apply Crossover operator to the selected solutions $X_\imath$ and $X_\jmath$ to obtain one offspring solution based on the adaptive crossover probability; /% Crossover Operation %/
9:     Add the offspring individuals to a candidate population $P'$ excluding any identical candidate;
10:  end for
11:  for $i := 1$ to $|P'|$ do
12:    Apply mutation operator to the candidate population $P'$ based on the adaptive mutation probability, excluding any identical candidate; /% Mutation Operation %/
13:  end for
14:  Estimate all individuals of the candidate population $P'$ with the ELM-based approximation model and find the best individual $X'_{\text{best}}$ among these candidate individuals;
15:  Generate a set $Q^*$ of all possible neighboring individuals relative to the best individual $X'_{\text{best}}$ using Local Search strategy;
16:  Estimate the individuals of the set $Q^*$ with the ELM fitness approximation model, and select the best individual from the set $Q^*$ to update the best individual $X'_{\text{best}}$ based on their approximated fitness values;
17:  Calculate the best $N_e$ individuals of the population $P'$ with exact fitness function, where, $N_e \ll N_p$, for example, $N_e = 10\%N_p$. Moreover, these $N_e$ individuals with exact fitness values are added into the training set to update the ELM model.
18:  Choose the best individual $X^*$ from these individuals to update the best individual $X_{\text{best}}$ found so far if $Z(X^*) < Z(X_{\text{best}})$, and set the parameter $t_{\text{nip}} = 0$; otherwise, do not update $X_{\text{best}}$ and increase $t_{\text{nip}}$ by 1.
19:  Combine the populations $P$ and $P'$ to form a population set $\tilde{P}$, and select the best $N_p$ individuals from the set $\tilde{P}$ based on their fitness values to update population $P$; /% Selection Operation II % /
20:  Perform the restarting strategy on the population $P$;
21:  $t = t + 1$;
22: end while
23: Output the best individual $X_{\text{best}}$ found by the algorithm and its objective function value.
```

Algorithm 2 The CBR algorithm.

```
1: Initialize: $y_i := 0, \forall i \in I$, $z_j := 0, \forall j \in J$;
2: Calculate the cost–benefit index for each plant using the expression $(f_i + \sum_{j\in J} c_{ij})/b_i$;
3: while $\sum_{i\in I} b_i y_i \le \sum_{k\in K} q_k$ do
4:   Select one plant $\imath$ with the smallest index from the set of unconsidered plants;
5:   $y_\imath = 1$;
6:   Delete plant $\imath$ from the set of the unconsidered plants;
7: end while
8: Calculate the cost–benefit index for each depot using the expression $(\sum_{i\in I} c_{ij} + g_j + \sum_{k\in K} d_{jk})/p_j$;
9: while $\sum_{j\in J} p_j z_j \le \sum_{k\in K} q_k$ do
10:  Select one depot $\jmath$ with the smallest index from the set of unconsidered depots;
11:  $z_\jmath = 1$;
12:  Delete depot $\jmath$ from the set of the unconsidered depots;
13: end while
14: Combine $y = (y_i : i \in I)$ and $z = (z_j : j \in J)$ into one complete individual.
```

In our proposed algorithm, infeasible individuals are corrected based on the cost–benefit indices of facilities. Afterward, the corrected individuals are improved based on Theorem 4.1. The entire correction procedure is named *modified and improved heuristic* (MIH). Its detailed steps are shown in Algorithm 3.

Once the initial population of the HEA/FA algorithm is produced, their fitness values are calculated exactly. In this paper, the objective function value is directly regarded as the fitness value. That is to say, the smaller the fitness value of one individual, the better its quality. In order to produce a sufficient number of solutions for the training set of the ELM, the size of the training set is initially chosen to be $2N_p$. Then the first N_p solutions with better fitness values are selected to constitute the initial population fed to the GA. Subsequently, this population is updated through genetic search operations described below.

4.3. Selection operation

Selection is invoked two times in the optimization process of the HEA/FA. *Selection for reproduction* (step 7 in Algorithm 1) is carried out before genetic operations are performed. This is performed on a purely random strategy without bias to filter two individuals from the population. Moreover, *selection for the individuals of next generation* (step 19 in Algorithm 1) is performed after the off-

Algorithm 3 The MIH algorithm.

```
1: Input: A given individual $X$, the parameters of the TSCFLP, $y_i := X_i$, $\forall i \in I$, $z_j := X_{|I|+j}$, $\forall j \in J$,the corresponding improved feasible individual $X^c \leftarrow [\ ]$, and its two components $y_i^c \leftarrow y_i$, $\forall i \in I$, $z_j^c \leftarrow z_j$, $\forall j \in J$;
2: for $i \in I$ do
3:   $w_i^p \leftarrow (f_i + \sum_{j\in J} c_{ij})/b_i$;
4: end for
5: Rank all plants according to their priority indexes $w_i^p, \forall i \in I$;
6: $i = 1$;
7: while $\sum_{i\in I} b_i y_i^c < \sum_{k\in K} q_k$ do
8:   Select next plant $i'$ with the minimum priority index $w_{i'}^p$ from the set of remaining plants;
9:   $y_{i'}^c \leftarrow 1$, $i \leftarrow i+1$;
10: end while
11: $i \leftarrow |I|$; %Improvethe corrected component $y^c = (y_{i'}^c), i' \in I$
12: while $\sum_{i\in I} b_i y_i^c > \sum_{k\in K} q_k$ do
13:   Select a plant $i'$ with the maximum priority index $w_{i'}^p$;
14:   $y_{i'}^c \leftarrow 0$;
15:   if $\sum_{i\in I} b_i y_i^c < \sum_{k\in K} q_k$ then
16:     $y_{i'}^c \leftarrow 1$;
17:     break
18:   end if
19:   $i \leftarrow i-1$;
20: end while
21: for $j \in J$ do
22:   $w_j^d \leftarrow (\sum_{i\in I} c_{ij} y_j + g_j + \sum_{k\in K} d_{jk})/p_j$;
23: end for
24: Rank all depots according to their priority indexes $w_j^d, \forall j \in J$;
25: $j = 1$;
26: while $\sum_{j\in J} p_j z_j^c < \sum_{k\in K} q_k$ do
27:   Select a depot $j'$ with the minimum priority index $w_{j'}^d$;
28:   $z_{j'}^c \leftarrow 1$, $j \leftarrow j+1$;
29: end while
30: $j \leftarrow |J|$; %Improve the corrected component $z_{j'}^c$, $\forall j' \in J$
31: while $\sum_{j\in J} p_j z_j^c > \sum_{k\in K} q_k$ do
32:   Select a depot $j'$ with the maximum priority index $w_{j'}^d$;
33:   $z_{j'}^c \leftarrow 0$;
34:   if $\sum_{j\in J} p_j z_j^c < \sum_{k\in K} q_k$ then
35:     $z_{j'}^c \leftarrow 1$;
36:     break
37:   end if
38:   $j \leftarrow j-1$;
39: end while
40: Output: The improved feasible individual $X^c = [y^c \quad z^c]$.
```

spring individuals of new generation have been produced. In order to maintain the diversity of the population, the best N_p solutions are chosen from the combination of parents and offspring.

4.4. Crossover operation

In our algorithm, the CX reproduction operator is adopted to finish the crossover operation. The CX reproduction operator proposed for the quadratic assignment problem (Merz & Freisleben, 1999; 2000), preserves the information contained in two parents

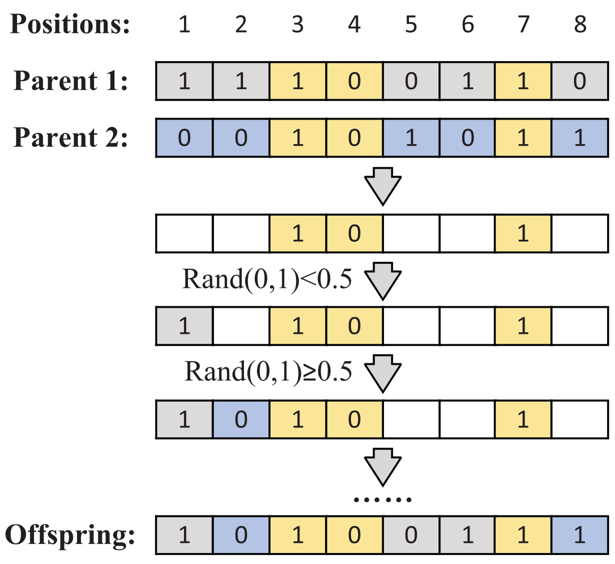

Fig. 4.1. The CX reproduction operator.

in the sense that all alleles of the offspring are taken either from the first or from the second parent. The operator does not carry out any implicit mutation operation since an element that is assigned to position ı in the offspring solution is also assigned to position ı in one or both parent solutions.

However, the CX operator was designed for the solution with permutation sequence, and is difficult to tackle the binary solutions in our problem. The CX operator is modified to achieve the crossover operation for the binary encoding scheme. The procedure of the CX operator can be delineated as follows.

Firstly, all elements found at the same positions in the two parents are assigned to the corresponding positions in the offspring. Then, a uniformly distributed random number between zero and one is generated. The random number is then compared with 0.5. If the random number is less than 0.5, the element in the selected position of the offspring is set to equal to the element of the same position in the first parent; otherwise, its corresponding position is occupied by the element from the second parent. Subsequently, the next no-assigned position is processed in the same way until all positions have been occupied. Fig. 4.1 illustrates the manner in which the CX reproduction operator performs.

In addition, the crossover probability ρ_c is selected adaptively as in (Srinivas & Patnaik, 1994). Based on their strategy, the crossover probability and the mutation probability are increased when the population tends to get stuck at a local optimum and are decreased when the population is scattered in the search space of the algorithm. Meanwhile, if the fitness value of the selected individual is better than average fitness value of the population, the corresponding probability should be smaller to preserve the individual in next population; otherwise, the corresponding probability should be larger to generate new individuals in next population. Let f_{best} be the best fitness of the current population, $\bar{f}$ be the average fitness value of the population and f' be the smaller of the fitness of the two individuals to be crossed. Then the crossover probability ρ_c is calculated as follows:

$$\rho_c = \begin{cases} \rho_c^{min} + \left(\dfrac{f_{best} - f'}{f_{best} - \bar{f}}\right) \times (\rho_c^{max} - \rho_c^{min}), & f' < \bar{f} \\ \rho_c^{max}, & f' \geq \bar{f} \end{cases} \tag{4.2}$$

where ρ_c^{max} is the maximum value of the crossover probability and ρ_c^{min} is the minimum value of the crossover probability. When

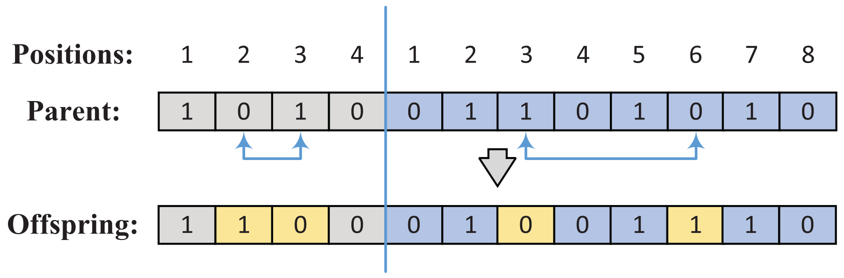
Positions: 1 2 3 4 1 2 3 4 5 6 7 8
Parent: 1 0 1 0 0 1 1 0 1 0 1 0
Offspring: 1 1 0 0 0 1 0 0 1 1 1 0

Fig. 4.2. The mutation operator.

$\rho_c^{max} = 0.9$ and $\rho_c^{min} = 0.5$, the HEA/FA algorithm could provide the best performance based on preliminary experiments.

4.5. Mutation operation

Generally, the inversion of the mutant gene is commonly used in the GA for the binary optimization problems. Nevertheless, the offspring individual delivered by the inversion operation is probably infeasible due to the violation of the capacity constraints for the problem under study. For instance, assume an incumbent individual just meets the requirement of the customers. When one plant is closed under the inversion operation, the production capacity of the remaining plants may be less than the demand of all customers. In this case, the candidate individual obtained by mutation becomes infeasible due to the violation of the capacity constraint. To avoid the above mentioned problem of common mutation strategy, alternatively, the mutation operator used in the HEA/FA exchanges two randomly selected elements in a chosen individual.

The two components of the individual are handled by the exchange operator separately. For the first $|I|$ binary values, two elements $\imath$ and $\jmath$ are randomly selected from the current individual and are swapped directly. The remaining $|J|$ binary values are operated in the same way. To describe the operation of the mutation operator, a simple example is shown in Fig. 4.2.

The mutation probability adopts the same adaptively updating strategy like Eq. (4.2). The expression for mutation probability ρ_m is given as

$$\rho_m = \begin{cases} \rho_m^{min} + \left(\dfrac{f_{best} - f'}{f_{best} - \bar{f}}\right) \times (\rho_m^{max} - \rho_m^{min}), & f < \bar{f} \\ \rho_m^{max}, & f \geq \bar{f} \end{cases} \tag{4.3}$$

where ρ_m^{min} and ρ_m^{max} are the minimum and the maximum of the mutation probability, f_{max} and $\bar{f}$ are the same as defined above, and f is the fitness of the individual under mutation. Based on the preliminary experiments, when ρ_m^{max} and ρ_m^{min} are set to 0.2 and 0.01 respectively, the algorithm could provide the best performance for the problem under study. From the expression of ρ_c and ρ_m, it is seen that ρ_c and ρ_m will get lower values for better individuals and get higher values for worse individuals.

4.6. Local search strategy

Generally, integrating a local search (LS) technique within a meta-heuristic will typically generate more competitive results. The main benefit of hybridizing evolutionary algorithms with a LS algorithm is to accelerate the speed of convergence. In our algorithm, the best individual of the population in each iteration is improved by a local search. The crucial idea of our LS strategy is generating all possible alternatives of the incumbent solution by inverting each element of a solution, while each of these alternatives is corrected by the MIH algorithm in order to meet the feasibility. If a candidate is better than the incumbent one in terms of the fitness value, the candidate individual will be accepted as a better solution. The LS procedure is outlined in Algorithm 4.

Algorithm 4 Thelocal search procedure.

```
1: Input: One individual $X$ and its fitness value;
2: The candidate individuals set $Q^* \leftarrow [\quad]$;
3: $\kappa$=0;
4: for each element $\imath$ in the individual $X$ do
5:   Perform inversion on the element $\imath$ to produce a new
     individual $X'$, while keeping other elements fixed;
6:   Use the MIH algorithm to correct and refine the new
     individual $X'$;
7:   if the individual $X'$ is different from the incumbent
     one $X$ then
8:     Add the individual $X'$ to the set $Q^*$;
9:     $\kappa = \kappa + 1$;
10:  end if
11: end for
12: Estimate the individuals of the set $Q^*$ with the ELM fit-
    ness approximation;
13: Choose the best one from the set $Q^*$. If the best one is
    better than $X$, then update the individual $X$. Otherwise,
    $X$ keeps unchanged;
14: Output: the individual $X$ and its fitness value.
```

4.7. Restarting strategy

In order to maintain the diversity of the population, it is common to use a restarting strategy in evolutionary algorithms (Ruiz, Maroto, & Alcaraz, 2006). In the proposed HEA/FA algorithm, a certain percentage of the population is substituted by randomly generated individuals with correction by the MIH algorithm. Whenever the number of identical binary elements in the best individual and in the worst individual is more than or equal to $0.9(|I| + |J|)$, the restarting operation is performed on the population. The population is sorted in non-decreasing order of objective function values. Then the first 90% of individuals from the sorted list are retained in the population. The remaining 10% of individuals are replaced with newly generated individuals at random with correction by the MIH algorithm.

4.8. Fitness evaluation

To avoid prematurely converging to a false optimum, the approximation model needs to be used together with the exact fitness function. For the individuals delivered by the genetic operation, their fitness values are approximated by the ELM-model. Since the best individual (X_{best}) found so far plays a central role in ensuring the whole population to converge to a true optimum, the

Table 5.1
Value intervals of the five classes of instances introduced by Fernandes et al. (2014).

Parameter	Class 1	Class 2	Class 3	Class 4	Class 5
b_i	[2B 5B]	[5B 10B]	[15B 25B]	[5B 10B]	[5B 10B]
f_i	$[2 \times 10^4\ 3 \times 10^4]$	$[2 \times 10^4\ 3 \times 10^4]$	$[2 \times 10^4\ 3 \times 10^4]$	$[2 \times 10^4\ 3 \times 10^4]$	$[2 \times 10^4\ 3 \times 10^4]$
c_{ij}	[35 45]	[35 45]	[35 45]	[50 100]	[35 45]
p_j	[2P 5P]	[5P 10P]	[15P 25P]	[5P 10P]	[5P 10P]
g_j	$[8 \times 10^3\ 1.2 \times 10^4]$	$[8 \times 10^3\ 1.2 \times 10^4]$	$[8 \times 10^3\ 1.2 \times 10^4]$	$[8 \times 10^3\ 1.2 \times 10^4]$	$[8 \times 10^3\ 1.2 \times 10^4]$
d_{jk}	[55 65]	[55 65]	[800 1000]	[50 100]	[800 1000]
q_k	[10 20]	[10 20]	[10 20]	[10 20]	[10 20]

Note: $B = \sum_{k \in K} q_k/|I|$, $P = \sum_{k \in K} q_k/|J|$.

HEA/FA always calculates the fitness of the best individual using the exact fitness function to ensure a correct convergence. Moreover, the set of the elite individuals with better fitness values are selected from the current population. In this paper, the proportion of the elites is set to 10%. That is to say, the number of elite individuals $N_e = 10\% \times N_p$.

In each generation of the HEA/FA, the true fitness values are calculated for the best individual and the elites in the population. These individuals in turn are added to the training set S to retrain and refine the ELM. As the evolution of the training set, the deviation between the exact fitness value and the approximated fitness value is expected to decrease, so that the evolutionary algorithm evolves better estimations produced by the ELM model.

5. Computational experiments

In this section, the numerical experiments are performed for evaluating the performance of the proposed algorithm. All the reported computational experiments presented hereunder were conducted on a personal computer with an Intel i7 3.6 GHz processor and 8GB of RAM. The performance of each algorithm is measured by the relative percentage deviation (RPD) defined by the equation

$$RPD(\%) = (Z(\text{alg}) - Z(\text{LB})) \times 100/Z(\text{LB}), \tag{5.1}$$

where Z(alg) is the solution value delivered by a specific algorithm and Z(LB) is the lower bound of the corresponding instance. Z(LB) can be obtained by solving the LP model with the continuous decision variables y_i and z_j, $i \in I$, $j \in J$.

5.1. Experimental instances

For assessing the performance of the HEA/FA algorithm, a well-defined set of instances developed by Fernandes et al. (2014) were used in this paper. The benchmark instances consist of five classes of instances by varying seven parameters of the considered problem: b_i, f_i, c_{ij}, p_j, g_j, d_{jk}, q_k. The value intervals of these parameters are outlined in Table 5.1. For each instance, the values of the seven parameters are randomly generated from an uniform random distribution. In Fernandes et al. (2014), the number of plants was set to 50 and 100, respectively. In all instances, the number of depots is twice as many as the number of plants, and the number of customers is twice as many as that of depots, so that $|K| = 2|J| = 4|I|$. For simplicity, the number of plants is used to indicate the size of the considered instances as in Fernandes et al. (2014).

5.2. Parameters tuning

The parameters selection can significantly affect the quality of an algorithm, in terms of solution performance and computational time. In our algorithm, the following three parameters are necessary to be considered: N_p, $t_{\max}$ and t_{nip}. Based on preliminary tests, when $t_{\max} = 200$ and $t_{\text{nip}} = 50$, the HEA/FA could provide a good

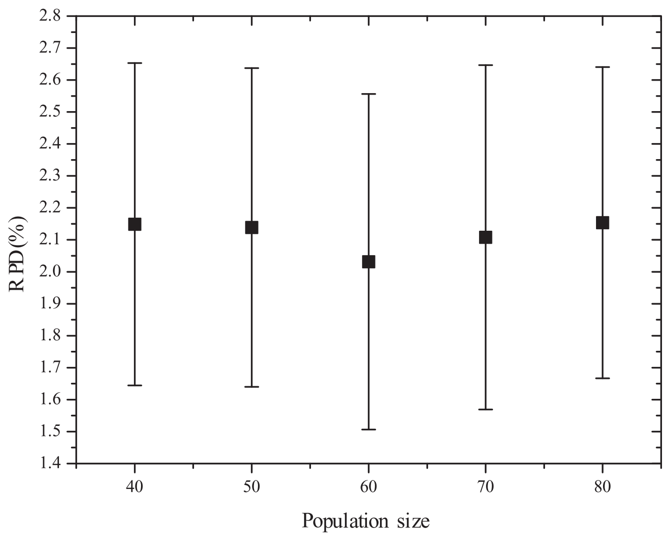

Fig. 5.1. Means plot and 99% confidence level Tukey's HSD intervals for population size N_p.

performance within a reasonable computational time. For the population size, the following candidate values are tuned for our algorithm: N_p=40, 50, 60, 70 and 80. A set of test instances with 50 plants were randomly generated in the same way as in Fernandes et al. (2014). These test instances were solved by the HEA/FA with different population sizes. The computational results were examined by means of the one-way analysis of variance (ANOVA) test. The means plot and 99% confidence level Tukey's Honestly Significant Difference (HSD) intervals for these 5 candidate values of N_p are described in Fig. 5.1.

As can be seen from the figure, the Tukey's HSD intervals for different N_p values have overlapping, which indicates that there is no significant difference among the five N_p values. However, when the population size is 60, the mean results delivered by the HEA/FA are better compared with other values. Therefore, the parameter N_p is set to 60 in the following computations.

5.3. Algorithms comparison

Several existing algorithms are deployed to perform a comparison for evaluating the results delivered by the HEA/FA. Only the GA proposed by Fernandes et al. (2014) was directly applied to solve the TSCFLP under study in the literature. In this paper, two additional meta-heuristic algorithms that were developed to solve the CFLP, including the tabu search (Sadigh et al., 2010), the hybrid firefly genetic algorithm (Rahmani & MirHassani, 2014) are modified to solve our problem. The computational time of the two algorithms are unacceptable if they seek exact function evaluations during the search process. For fair comparison, the function approximation model based on the extreme learning machine is

Table 5.2
Computational results for instances with 50 plants.

Class	Instance	LB	GA	TS		HFGA		HEA/FA		
				RPD_a (%)	Time (s)	RPD_a (%)	Time (s)	RPD_m (%)	RPD_a (%)	Time (s)
1	1	721209.6	0.13	0.94	103.46	0.97	108.25	**0.13**	0.24	416.17
	2	730451.6	0.40	1.16	132.78	4.48	100.18	**0.31**	**0.36**	492.78
	3	731885.3	0.24	0.38	98.43	0.38	104.29	**0.22**	**0.24**	316.08
	4	721,515	0.81	2.74	95.93	3.17	105.69	**0.54**	**0.59**	455.55
	5	713633.8	0.82	1.66	100.98	1.90	108.70	0.86	0.97	463.69
2	1	479860.2	2.69	3.40	37.77	3.40	55.21	2.74	2.82	230.84
	2	483072.2	2.30	3.78	45.90	3.85	59.11	2.34	2.41	199.11
	3	486018.5	2.14	3.13	37.80	3.13	51.98	**1.87**	**1.89**	167.28
	4	482374.6	2.04	3.13	38.23	3.35	54.53	2.07	2.12	162.13
	5	474803.3	3.14	4.12	38.00	4.27	57.74	**3.12**	**3.12**	170.57
3	1	2,608,800	3.07	4.03	21.04	4.17	41.99	3.14	3.30	117.09
	2	2,616,252	3.12	3.97	20.24	4.03	41.08	3.30	3.36	116.67
	3	2,598,277	3.11	4.09	22.53	4.30	39.55	3.30	3.39	161.80
	4	2,612,534	3.07	3.91	22.01	4.10	40.88	3.18	3.22	148.86
	5	2,568,856	3.01	3.91	22.18	4.18	41.53	3.17	3.19	158.15
4	1	525294.1	3.14	6.09	39.29	6.52	56.62	3.36	3.54	216.81
	2	526911.7	2.33	4.00	37.40	4.00	54.10	2.74	2.92	199.39
	3	532592.3	2.66	4.30	40.27	4.40	55.66	**2.59**	**2.62**	247.29
	4	529,372	2.53	4.20	37.45	4.20	51.02	2.63	2.81	261.75
	5	521470.1	3.13	5.41	52.79	6.09	52.31	3.18	3.18	182.24
5	1	2,743,547	1.20	1.62	37.53	1.70	57.73	**1.16**	**1.20**	199.23
	2	2,752,021	1.07	1.64	42.06	1.81	58.78	1.11	1.16	259.72
	3	2,737,769	1.10	1.81	39.98	2.02	54.52	1.22	1.28	202.71
	4	2,748,216	1.07	1.58	36.70	1.61	52.04	1.10	1.13	203.32
	5	2,702,350	1.25	1.73	45.21	1.97	55.62	1.31	1.34	182.20
	Average		1.98	3.07	49.84	3.36	62.36	2.03	2.10	237.26

adopted in the above two algorithms. Best solutions from the initial population of the HEA/FA are fed as initial solutions for both the TS and the HFGA. Meanwhile the best neighborhood solution of the TS is evaluated by using the exact fitness function at each iteration.

Besides the maximum iteration stopping criterion, the TS and the HFGA will stop the search process if the best solution is not improved upon in t_{nip} successive iterations. Here, the parameter t_{nip} is set to 50 just like in the HEA/FA. Other parameters of the TS and the HFGA are identical to those given in Sadigh et al. (2010) and Rahmani and MirHassani (2014), respectively.

The above algorithms, including the HEA/FA, the TS and the HFGA, were coded in Matlab 7.14 within a Windows 7 environment. Additionally, the computational results of the GA designed by Fernandes et al. (2014) were listed directly in our statistical tables for the purpose of comparing to the best results in literature. Due to the stochastic nature of these approaches, each was run five times for a given instance to reach reliable results. The best objective function value among 5 runs and the average objective value are recorded for each algorithm. Let RPD_m denote the least RPD value of the best objective function value and RPD_a be the average RPD value of five runs in the following statistical tables.

5.4. Experimental results

Table 5.2 reports the computational results of the instances with 50 plants. In the table, when the results of the HEA/FA are better than those of other three algorithms, the corresponding RPD values are in boldface font. It can be observed that the HEA/FA gives good solutions in a reasonable computational time. The average RPD values delivered by the HEA/FA are slightly larger than that of the GA. *Remarkably, the HEA/FA gives better solutions for six instances compared with the GA.* The RPD values of the GA and the HEA/FA are significantly smaller than that of the TS and the HFGA. But the run time of the TS and the HFGA is shorter compared with that of the HEA/FA. This is because that the TS is not a population-based algorithm. Its computational time is the shortest among the three algorithms for instances with 50 plants.

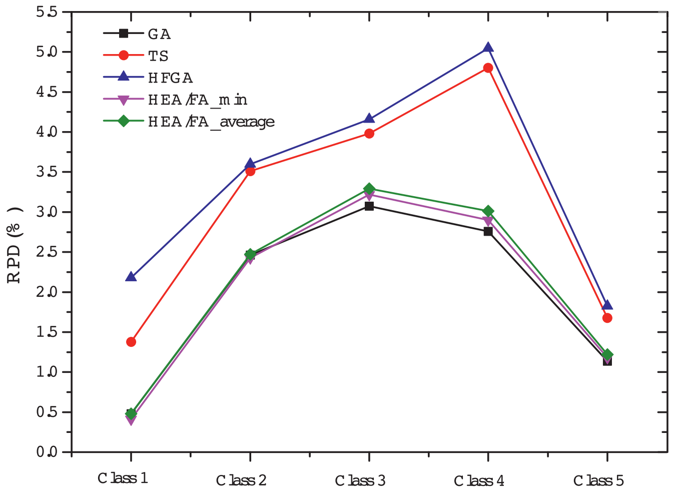

Fig. 5.2. Average RPD values delivered by four algorithms for instances with 50 plants.

Fig. 5.2 shows the performances of the four algorithms under different problem classes with 50 plants. In the figure, the RPD value is the mean of five problem instances for each class. The mean RPD values of Class 1 and Class 5 given by the four algorithms are smaller than that of Class 2, Class 3 and Class 4. It can be concluded that instances of Class 1 and 5 can be easily solved by using the meta-heuristics. It can be also observed that the performance of the HEA/FA is consistent with that of the GA in solving instances of Class 1, Class 2 and Class 5. The average RPD values obtained by the HEA/FA are slightly larger than that of the GA in the instances of Class 3 and Class 4. *Although the TS and the HFGA are good at solving the CFLP, their results are apparently worse compared to the GA and the HEA/FA.* The TS and the HFGA are more likely to be trapped in local optima when solving the TSCFLP. Moreover, the RPD values delivered by the TS with less computational time are better than that of the HFGA.

Table 5.3
Computational results for instances with 100 plants.

Class	Instance	LB	GA	TS		HFGA		HEA/FA		
				RPD$_a$ (%)	Time (s)	RPD$_a$ (%)	Time (s)	RPD$_m$ (%)	RPD$_a$ (%)	Time (s)
1	1	1,475,952	0.55	0.61	1309.98	0.81	1028.52	**0.29**	**0.33**	1341.67
	2	1,462,736	1.01	0.95	1304.37	0.97	1037.95	**0.27**	**0.43**	1514.37
	3	1,492,163	0.34	1.22	1503.02	1.66	1079.60	**0.23**	0.48	1481.83
	4	1,459,076	0.49	0.47	1306.33	0.51	1061.16	**0.25**	**0.28**	1122.33
	5	1,490,742	0.67	0.77	1304.20	0.77	1079.56	**0.17**	**0.24**	1501.11
2	1	970908.5	0.89	0.93	529.72	2.70	429.30	**0.63**	**0.70**	944.83
	2	965908.5	0.74	1.06	503.31	1.06	454.48	**0.47**	**0.56**	908.30
	3	975499.7	1.42	0.61	518.24	0.61	479.33	**0.20**	**0.25**	922.00
	4	973019.1	0.56	0.80	579.12	0.95	455.10	**0.56**	0.59	1083.61
	5	941,567	1.12	1.87	508.43	1.87	454.35	**0.86**	**0.96**	1159.33
3	1	5,213,566	1.63	2.35	251.33	2.50	253.56	1.91	2.03	855.68
	2	5,191,321	1.67	2.36	244.18	2.51	229.73	1.96	1.99	1220.48
	3	5,145,991	1.58	2.14	253.41	2.27	226.82	1.78	1.84	1193.12
	4	5,225,601	1.74	2.37	259.86	2.48	234.88	1.98	2.01	968.64
	5	5,163,182	1.72	2.26	290.81	2.46	237.05	1.99	2.08	942.29
4	1	1,052,172	0.82	1.27	498.54	1.27	425.89	0.91	1.04	878.64
	2	1,043,553	0.93	1.12	511.60	1.12	453.81	**0.82**	**0.89**	672.60
	3	1,050,683	1.88	1.98	548.44	2.17	457.95	**1.00**	**1.29**	877.26
	4	1,044,571	0.96	2.64	558.05	2.69	443.97	1.36	1.62	797.26
	5	1,053,869	0.64	1.74	511.31	1.74	470.10	0.94	1.06	843.02
5	1	5,486,098	0.48	0.97	504.81	1.08	431.94	0.61	0.62	967.57
	2	5,461,680	0.47	0.61	501.41	0.61	465.11	**0.41**	**0.44**	1017.10
	3	5,425,391	0.62	0.82	544.59	0.97	439.15	0.64	0.73	1371.96
	4	5,494,811	0.52	0.78	538.09	0.78	429.18	0.58	0.58	1084.43
	5	5,442,621	0.47	0.73	516.53	0.75	438.65	**0.43**	**0.47**	1162.63
Average			0.96	1.34	635.99	1.49	527.89	0.85	0.94	1073.28

The results of problem instances with 100 plants obtained by the four algorithms are listed in Table 5.3. In the table, the average RPD value of the best solution among five runs given by the HEA/FA is 0.85%, which is less than the average RPD value of 0.96% given by the GA. Moreover, the average RPD of the mean objective function value delivered by the HEA/FA is 0.94%, which is slightly less than the one 0.96% of the GA. There are 12 instances in which the quality of the average solutions obtained by the HEA/FA is better than that of the GA. And the best solutions yielded by the HEA/FA are better than that of the GA in 14 out of 25 instances. Note that these RPD values are marked by the bold-faced font.

As in the instances with 50 plants, the RPD values obtained by the TS and the HFGA are still worse (larger) than that of the GA and the HEA/FA. Interestingly, the computational results show that the TS outperforms the HFGA, but it consumes more run time compared with the HFGA. In the same way, the average RPD values given by the four algorithms under different problem classes are plotted in Fig. 5.3. The results given by the GA and the HEA/FA are clearly better than that of the TS and the HFGA. The HEA/FA significantly outperforms the GA in the problem instances of Class 1, Class 2 and Class 5. But the proposed HEA/FA is not as good at handling the problem instances with Class 3 and Class 4.

Furthermore, the computational time of the HEA/FA is listed in Tables 5.2 and 5.3. The longest time of HEA/FA is 492.78 s and 1514.37 s in solving problem instances with 50 plants and 100 plants, respectively. Most computational time is consumed by exact fitness evaluation in the search process. Overall, the computational time of the HEA/FA is acceptable for obtaining a good quality solution to the underlying problem. Based on the above analysis, the HEA/FA could be regarded as an alternative algorithm in solving the TSCFLP, especially for large-sized problems.

6. Discussion

The proposed hybrid algorithm can deliver better solutions in solving the two-stage facility location problem compared with other existing algorithms, such as the genetic algorithm (GA), the tabu search (TS) and the hybrid firefly genetic algorithm (HFGA).

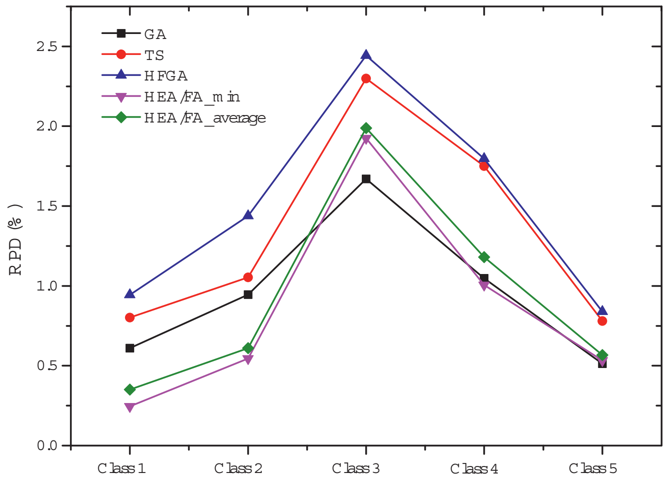

Fig. 5.3. Average RPD values delivered by four algorithms for instances with 100 plants.

Certainly, the framework of the HEA/FA can solve other facility location problems, including the single-stage capacitated facility location problem (Melo et al., 2009), the single source capacitated facility location problem (Guastaroba & Speranza, 2014), etc. Moreover, the proposed algorithm framework can be extended to solve many other binary optimization problems. Therefore, it is highly flexible which is important for applications in solving these practical engineering problems.

In addition, this algorithm is simple to implement since its optimization process adopts the generic GA framework and the code of the extreme learning machine can be obtained from the following web-page: http://www.ntu.edu.sg/home/egbhuang/. Meanwhile, the framework of the HEA/FA consists of two nested algorithms for maintaining the diversity of the population and speeding up the convergence of the algorithm. According to the computational results, the proposed algorithm is very successful in delivering optimal/near-optimal solutions for the TSCFLP benchmark in-

Table 6.4
Strengths and weaknesses of the proposed HEA/FA algorithm.

Strengths	Weaknesses
Flexible to extend	Require some parameters
Simple to implement	Consume more computational time
Easy to understand	
Robust parameter design	
Parallelizable	

stances. The success of the proposed algorithm is especially prominent for large-sized problems.

Generally, infeasible solutions are discarded by the GA during the searching process (Fernandes et al., 2014). If that happens, infeasible solutions with potential excellent genes (elements) are excluded, albeit saving the computational time. However, for our algorithm, infeasible individuals are corrected by the approach with the cost–benefit index based on the capacity constraints of plants and depots, and subsequently preserved in the population. In this new scheme, the diversity of population is maintained very well. As shown by our experiments in Section 5, this scheme can give better solutions for large-sized instances.

Ideally, a good meta-heuristic algorithm has relatively few control parameters and is robust to the presence of parameters variation. For the proposed algorithm, the extreme machine learning algorithm and the local search only require very few number of parameters, and the number of parameters of genetic algorithm are not so many. So, the proposed algorithm can be considered to use only a few parameters. Furthermore, the proposed algorithm is less sensitive to different parameters as shown in Fig. 5.1.

Although this paper has adopted the ELM-based approximation model to alleviate the computational burden of the HEA, the number of exact fitness function evaluations is still large for selected individuals. This situation encountered by the proposed algorithm can be looked as the weakness of the HEA/FA. But it is worth to obtain better quality of solutions in the expense of slightly longer computing time. Table 6.4 summarizes the strengths and weaknesses of the proposed HEA/FA algorithm mentioned above.

Theoretically, the hybrid algorithm that combines the GA and a local search has a fascinating ability compared with either the single GA or the single local search algorithm. In the framework of the HEA/FA, the GA works as a general optimization procedure and the local search is introduced to further refine the best solution found in the population. In order to maintain the population diversity, infeasible individuals are modified and re-casted into the optimization process. Thanks to the introduction of the fitness approximation by the EML algorithm, the computational time of solving the transportation problem can be decreased significantly.

The computational results reveal a few more insightful implications. Firstly, the proposed HEA/FA although are designed for a two-stage capacitated facility location problem, it can be easily modified to solve other facility location problem variants as well as other 0–1 binary optimization problems. Moreover, the HEA/FA provides new best solutions as a reference for comparison purpose of further researches in the TSCFLP literature. Finally, the introduction of extreme machine learning provides an alternative for the construction of approximation models in the evolutionary algorithms, in order to reduce the computational time. A better framework of evolutionary algorithm yielding better results in a reasonable amount of time is crucial for expert and intelligent systems in the optimization field, in addition to robustness, simplicity et al.

7. Conclusions

In this paper, a hybrid evolutionary algorithm framework with fitness approximation was proposed to solve the TSCFLP. To improve the performance of the algorithm, the following tricks are adopted: two heuristics are introduced during the population initialization, and a local search based on inversion operation is integrated to refine the best individual at each iteration. In addition, probabilistically adaptive crossover and mutation are used to prevent the algorithm from getting stuck at a local optimal solution. In order to reduce the time consumption in fitness evaluation operation, the fitness values of most individuals are calculated by the surrogate model based on the extreme learning machine. However, the elite solutions are assessed by the exact fitness evaluation for getting a proper balance between accuracy and time efficiency of fitness evaluation. The performance of the proposed algorithm was tested and evaluated on two sets of benchmark instances. The computational results show that the HEA/FA in general delivers good results for the TSCFLP in a reasonable time, especially in large-sized instances. Compared with other existing algorithms, the HEA/FA provides better solutions for large-sized problem.

Further research includes the consideration of additional characteristics in the problem setting to make it even more realistic. For instance, some loading/discharging operations such as handling costs studied by Li, Chu, Prins, and Zhu (2014) in facilities might prove useful. Additionally, traffic situations such as asymmetrical transport distance matrices could be considered in more complex scenarios. For the HEA/FA, the probability of applying it to solve other combinatorial optimization problems could be developed, such as production scheduling problems and facility layout problems.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 51405403) and China Postdoctoral Science Foundation funded project (No. 2015M582566).

References

Arostegui, M. A., Kadipasaoglu, S. N., & Khumawala, B. M. (2006). An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems. *International Journal of Production Economics, 103*, 742–754.
Avella, P., Boccia, M., Sforza, A., & Vasil'ev, I. (2008). An effective heuristic for large-scale capacitated facility location problems. *Journal of Heuristics, 15*, 597–615.
Cornuéjols, G., Nemhauser, G. L., & Wolsey, L. A. (1990) (pp. 119–171)). New York, NY, USA: John Wiley & Sons.
Correa, E. S., Steiner, M. T. A., Freitas, A. A., & Carnieri, C. (2004). A genetic algorithm for solving a capacitated p-median problem. *Numerical Algorithms, 35*, 373–388.
Dias, J., Rocha, H., Ferreira, B., & do Carmo Lopes, M. (2014). A genetic algorithm with neural network fitness function evaluation for IMRT beam angle optimization. *Central European Journal of Operations Research, 22*, 431–455.
Fernandes, D. R., Rocha, C., Aloise, D., Ribeiro, G. M., Santos, E. M., & Silva, A. (2014). A simple and effective genetic algorithm for the two-stage capacitated facility location problem. *Computers & Industrial Engineering, 75*, 200–208.
Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization*: Vol. 7. John Wiley & Sons.
Guastaroba, G., & Speranza, M. (2014). A heuristic for {BILP} problems: The single source capacitated facility location problem. *European Journal of Operational Research, 238*, 438–450.
Guastaroba, G., & Speranza, M. G. (2012). Kernel search for the capacitated facility location problem. *Journal of Heuristics, 18*, 877–917.
Hacioglu, A. (2007). Fast evolutionary algorithm for airfoil design via neural network. *AIAA Journal, 45*, 2196–2203.
Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing, 2*, 88–105.
Huang, G., Huang, G.-B., Song, S., & You, K. (2015). Trends in extreme learning machines: A review. *Neural Networks, 61*, 32–48.
Huang, G.-B., Wang, D., & Lan, Y. (2011). Extreme learning machines: A survey. *International Journal of Machine Learning and Cybernetics, 2*, 107–122.
Huang, G.-B., Zhu, Q.-Y., & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing, 70*, 489–501.
Jaramillo, J. H., Bhadury, J., & Batta, R. (2002). On the use of genetic algorithms to solve location problems. *Computers & Operations Research, 29*, 761–779.
Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing, 9*, 3–12.
Jin, Y., Olhofer, M., & Sendhoff, B. (2002). A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation, 6*, 481–494.

Klose, A. (1999). An LP-based heuristic for two-stage capacitated facility location problems. *The Journal of the Operational Research Society, 50*, 157–166.

Klose, A. (2000). A Lagrangean relax-and-cut approach for the two-stage capacitated facility location problem. *European Journal of Operational Research, 126*, 408–421.

Klose, A., & Drexl, A. (2005). Lower bounds for the capacitated facility location problem based on column generation. *Management Science, 51*, 1689–1705.

Li, J., Chu, F., Prins, C., & Zhu, Z. (2014). Lower and upper bounds for a two-stage capacitated facility location problem with handling costs. *European Journal of Operational Research, 236*, 957–967.

Litvinchev, I., & Ozuna Espinosa, E. (2012). Solving the two-stage capacitated facility location problem by the Lagrangian heuristic. In *Computational logistics*. In *Lecture notes in computer science: Vol. 7555* (pp. 92–103). Berlin, Heidelberg.

Marín, A. (2007). Lower bounds for the two-stage uncapacitated facility location problem. *European Journal of Operational Research, 179*, 1126–1142.

Melo, M., Nickel, S., & da Gama, F. S. (2009). Facility location and supply chain management – A review. *European Journal of Operational Research, 196*, 401–412.

Merz, P., & Freisleben, B. (1999). A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In *Proceedings of the 1999 congress on evolutionary computation: Vol. 3* (pp. 2063–2070).

Merz, P., & Freisleben, B. (2000). Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Transactions on Evolutionary Computation, 4*, 337–352.

Mesa, J., & Boffey, T. (1996). A review of extensive facility location in networks. *European Journal of Operational Research, 95*, 592–603.

Owen, S., & Daskin, M. (1998). Strategic facility location: A review. *European Journal of Operational Research, 111*, 423–447.

Rahmani, A., & MirHassani, S. (2014). A hybrid firefly-genetic algorithm for the capacitated facility location problem. *Information Sciences, 283*, 70–78.

Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega, 34*, 461–476.

Sadigh, A. N., Mozafari, M., & Kashan, A. H. (2010). A mixed integer linear program and tabu search approach for the complementary edge covering problem. *Advances in Engineering Software, 41*, 762–768.

Sahin, G., & Sural, H. (2007). A review of hierarchical facility location models. *Computers & Operations Research, 34*, 2310–2331.

Snyder, L. (2006). Facility location under uncertainty: A review. *IIE Transactions, 38*, 537–554.

Srinivas, M., & Patnaik, L. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics, 24*, 656–667.

Sun, M. (2011). A tabu search heuristic procedure for the capacitated facility location problem. *Journal of Heuristics, 18*, 91–118.

Wang, S., & Watada, J. (2012). Capacitated two-stage facility location problem with fuzzy costs and demands. *International Journal of Machine Learning and Cybernetics, 4*, 65–74.

Yang, Z., Chu, F., & Chen, H. (2012). A cut-and-solve based algorithm for the single-source capacitated facility location problem. *European Journal of Operational Research, 221*, 521–532.

Zhang, J., Chen, B., & Ye, Y. (2005). A multiexchange local search algorithm for the capacitated facility location problem. *Mathematics of Operations Research, 30*, 389–403.

Zhang, Q., Liu, W., Tsang, E., & Virginas, B. (2010). Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Transactions on Evolutionary Computation, 14*, 456–474.

Zheng, Z., Chen, X., Liu, C., & Huang, K. (2016). Using support vector machine and dynamic parameter encoding to enhance global optimization. *Engineering Optimization, 48*, 851–867.