9-2015

Ensemble Multi-Objective Biogeography-Based Optimization with Application to Automated Warehouse Scheduling

Haiping Ma
Shaoxing University

Shufei Su
Shanghai University

Daniel J. Simon
Cleveland State University, d.j.simon@csuohio.edu

Minrui Fei
Shanghai University

Original Citation

H. Ma, S. Su, D. Simon and M. Fei, "Ensemble multi-objective biogeography-based optimization with application to automated warehouse scheduling," Eng Appl Artif Intell, vol. 44, no. 9, pp. 79-90, 2015.

Ensemble multi-objective biogeography-based optimization with application to automated warehouse scheduling

Haiping Ma, Shufei Su, Dan Simon, Minrui Fei

1. Introduction

Warehousing is an important part of production supply chain management, and serves as the backbone in many manufacturing enterprises. Warehousing keeps stocks of products until they are ready to be delivered to the market place. A delay in product delivery may lead to the failure of production supply chains. Efficient warehouse management contributes to the timely delivery of the product (Choi et al., 2013; Yeung et al., 2010, 2011). Modern warehouses are equipped with storage and retrieval (S/R) machines to pick up products from an input/output (I/O) location and store them at specific locations, and then to retrieve outgoing products from other storage locations and deliver them to the I/O location. Although S/R machines enhance warehouse management, scheduling is a challenging and vital task (Lerher et al., 2015b; Wang and Fang, 2011). The time and cost of product allocation and delivery are important variables to consider during warehouse scheduling.

In the past few decades, warehouse scheduling has been the subject of much research, including energy efficient design (Lerher et al., 2013), travel time models for shuttle-based systems (Lerher et al., 2015a), travel time models for aisle transfer systems (Lerher et al., 2010a), travel time models for double-deep systems (Lerher et al., 2010b), and models for mini-load multi-shuttle systems (Lerher et al., 2011). Other automated warehouse schedule research is reported in Berg (1999), Chan and Kumar (2009), Roodbergen and Vis (2009), and Yang et al. (2013).

Warehouse scheduling is a typical NP-hard problem, which is one of the most challenging types of combinatorial optimization problems (Gagliardi et al., 2012). Since this problem is so important for production supply chain success, more research needs to be carried out to make automated warehouse scheduling more robust and efficient. Motivated by these considerations, this paper highlights the benefits associated with automated warehouse scheduling and introduces a new evolutionary algorithm to find efficient schedules for warehouse management.

In recent years, ensemble learning has been introduced to enhance the performance of various systems; for example, feature selection, optimization, clustering analysis, etc. Ensemble learning is a hybrid method that uses multiple learning models instead of a single model. This approach is intuitively attractive because a single model may not always be the best to solve a complex problem, but multiple models are likely to yield results that are better than each of the constituent models. Ensemble learning has been successfully applied to time series prediction and classification, and to

evolutionary algorithms. It has been used with evolution strategies (ES) (Mallipeddi and Suganthan, 2010b), evolutionary programming (EP) (Mallipeddi and Suganthan, 2010a), harmony search (HS) (Pan et al., 2009), and differential evolution (DE) (Mallipeddi et al., 2011). It has been used to solve constrained optimization (Tasgetiren et al., 2010a), multi-objective optimization (Zhao and Suganthan, 2010), dynamic optimization (Yu and Suganthan, 2009), and the traveling salesman problem (Tasgetiren et al., 2010b).

Biogeography-based optimization (BBO) (Simon, 2008) is an evolutionary global optimization algorithm that was introduced in 2008. It is modeled after the immigration and emigration of species between habitats. The application of these processes to optimization allows information sharing between candidate solutions. BBO uses the fitness of each candidate solution to determine its immigration and emigration rate. The emigration rate is proportional to fitness and the immigration rate is inversely proportional to fitness. BBO has demonstrated good performance on various benchmark functions and real-world optimization problems (Ma and Simon, 2011). BBO has also been modified to solve multi-objective optimization problems (MOPs) (Chutima and Wong, 2014; Costa e Silva et al., 2012; Jamuna and Swarup, 2012; Ma et al., 2012).

The aim of this paper is to propose and study an ensemble of multi-objective biogeography-based optimization (MBBO) algorithms, including vector evaluated biogeography-based optimization (VEBBO), non-dominated sorting biogeography-based optimization (NSBBO), and niched Pareto biogeography-based optimization (NPBBO) (Simon, 2013), and apply the new algorithm to the automated warehouse scheduling problem. This paper shows how MBBO algorithms can be integrated to obtain a new algorithm called ensemble multi-objective biogeography-based optimization (EMBBO), and then presents a comparative study on multi-objective benchmark functions and automated warehouse scheduling problems. The methods in this paper could also serve as a template for the extension of any other evolutionary algorithm to multi-objective optimization.

The motivation of proposing EMBBO in this research is two-fold. First, we have observed that ensemble EAs outperform constituent EAs in many applications, as noted above, because of their greater adaptability. Second, we have observed that MBBO has proven itself to be an effective multi-objective optimization algorithm, also noted above. Combining these two observations leads us to propose an ensemble BBO algorithm, EMBBO, as a high-performing multi-objective optimization algorithm.

The original contributions of this paper include the following. (a) A new real-world-based automated warehouse scheduling model is formulated as a constrained multi-objective optimization problem. (b) The idea of ensemble learning is applied to MBBO to establish the new EMBBO algorithm. Results show that EMBBO outperforms its constituent MBBO algorithms for most of the unconstrained and constrained multi-objective benchmark functions that we study. (c) EMBBO has a lower computational cost than its constituent algorithms because it simultaneously uses multiple parallel populations to reduce running time in comparison with single MBBO algorithms which use populations whose number of individuals is the sum of the three constituent algorithms. (d) EMBBO solves the automated warehouse scheduling problem.

The remainder of this paper is organized as follows. Section 2 builds a mathematical model of the automated warehouse scheduling problem. Section 3 reviews the standard BBO and various MBBO algorithms, and then integrates them to realize EMBBO. Section 4 presents performance comparisons on benchmark functions between EMBBO, constituent MBBO algorithms, and 2009 Congress on Evolutionary Computation (CEC) algorithms, and then applies EMBBO to the automated warehouse scheduling model. Section 5 presents conclusions and suggests directions for future work. The abbreviations, notations, and symbols used in this paper are summarized in Appendix A.

2. Automated warehouse scheduling model

The layout of the automated warehouse system is shown in Fig. 1 (Yang et al., 2013), and is called a multi-aisle automated storage and retrieval system (multi-aisle AS/RS) with a curve-going S/R machine. It includes six components: S/R machine, picking aisles, cross warehouse aisle, storage racks (SR), rolling conveyor, and I/O location. As shown in the figure, the S/R machine can go in and out at both ends of every picking aisle, pick up products at the I/O location and store them at specific storage units in SR, and then retrieve outgoing products from other storage units and deliver them to the I/O location. The aim of an automated warehouse scheduling system is to improve scheduling efficiency.

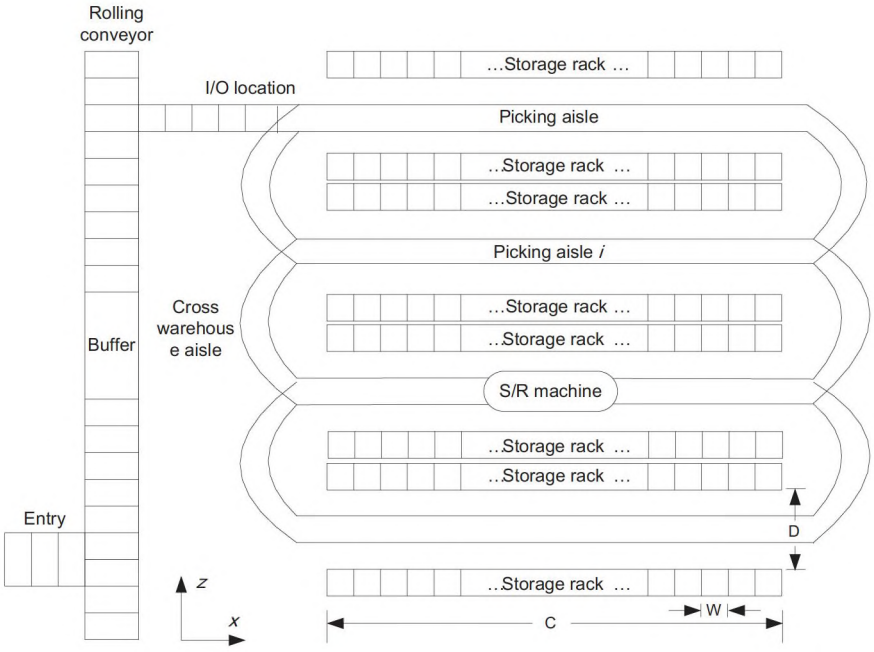

Fig. 1. Layout of the warehouse system.

In this model, there are many storage units in each SR. There are m storage products, storage units of which are denoted as $(p_{u1}, p_{u2}, p_{u3}, \ldots, p_{um})$. There are n outgoing products, storage units of which are denoted as $(p_{o1}, p_{o2}, p_{o3}, \ldots, p_{on})$. In general, the numbers and units of the storage products are not the same as those of the outgoing products; namely, $m \neq n$ and $p_{ui} \neq p_{oi}$.

Suppose the S/R machine can hold N products at a time. Then the S/R machine picks up N or fewer products from the I/O location, and puts them into storage units. Then it retrieves N or fewer outgoing products from the other storage units, and delivers them to the I/O location. For m storage products and n outgoing products, there are $m+n$ storage units, namely $(p_1, p_2, \ldots, p_m, p_{m+1}, \ldots, p_{m+n})$, and the S/R machine needs to execute $max\{\lceil m/N \rceil, \lceil n/N \rceil\}$ tasks to store and transport all products, where $\lceil \varphi \rceil$ denotes the smallest integer greater than or equal to φ. When each task corresponds to one route, the automated warehouse scheduling problem is translated into an optimization problem of selecting the optimal $max\{\lceil m/N \rceil, \lceil n/N \rceil\}$ routes from all possible routes to complete the storage/retrieval tasks while satisfying real-world constraints.

In Fig. 1, x and z are the two directions in the horizontal plane. The velocities in the x and z directions are called horizontal velocities, and the corresponding horizontal velocities are v_x and v_z, which are equal to each other. y is the vertical direction and the corresponding vertical velocity is v_y, which is independent from v_x. The distance between adjacent SRs in the same picking aisle is D, and the number of storage units in each SR is C. The width, length and height of each storage unit are denoted as W, L and H, respectively. The Euclidean coordinates of each storage unit is denoted as $p(x, y, z)$, and the I/O location of the S/R machine is denoted as $p_0(0, 0, 0)$.

When developing the automated warehouse scheduling system, the following assumptions are made:

- The multi-aisle AS/RS is divided into picking aisles with SRs on both sides, so there are double SRs between the picking aisles and a single SR along each warehouse wall.
- There is one S/R machine.
- The S/R machine is able to move along the cross warehouse aisle by using the curved rails at the end of the picking aisles.
- The S/R machine travels at a constant velocity both in the horizontal and in the vertical directions. Although the inclusion of acceleration and deceleration will affect the scheduling results, it will not affect the optimization approach considered in this paper.
- The S/R machine simultaneously begins the lifting and travelling in the pick aisle. Although the inclusion of non-simultaneous lifting and travelling will affect the scheduling results, it will not affect the optimization approach considered in this paper.
- The input station dwell-point strategy is used. That is, the S/R machine stays always at input location when it is idle. Although the use of other dwell-point strategies will affect the scheduling results, it will not affect the optimization approach considered in this paper.
- The randomized storage assignment policy is used. That is, any storage location within the S/R is equally likely to be selected for the storage or retrieval request. Similar to the constant velocity model assumed above, this is a simplified model. Although the use of non-random storage assignment will affect the scheduling results, it will not affect the optimization approach considered in this paper.
- As a first-order approximation, the pickup and set down times, and additional overhead times for manipulating the S/R machine, are ignored.

Definition 1. If the S/R machine travels the route $r \in [1, max(\lceil m/N, n/N \rceil)]$, then $l_r = 1$; otherwise, $l_r = 0$. Similarly, if the storage unit p_i belongs to route r, then $g_{ir} = 1$; otherwise, $g_{ir} = 0$.

Definition 2. If the S/R machine travels from storage unit $p_i(x_i, y_i, z_i)$ to another storage unit $p_j(x_j, y_j, z_j)$ for $i, j \in [1, m+n]$, then $e_{ij} = 1$; otherwise, $e_{ij} = 0$. The travel distance d_{ij} and time t_{ij} of the S/R machine is denoted as

$$d_{ij} = \begin{cases} W \times |x_i - x_j| + H \times |y_i - y_j| & \text{if } z_i = z_j \\ min\begin{pmatrix} W \times |x_i - x_j| + H \times |y_i - y_j| + D \times |z_i - z_j|, \\ W \times |(C - x_i) - (C - x_j)| + H \times |y_i - y_j| + D \times |z_i - z_j| \end{pmatrix}, & \text{if } z_i \neq z_j \end{cases} \tag{1}$$

and

$$t_{ij} = \begin{cases} max\left(W \times |x_i - x_j|)/v_x, \; H \times |y_i - y_j|\right)/v_y\right) & \text{if } z_i = z_j \\ max\left(min\begin{pmatrix} (W \times |x_i - x_j| + D \times |z_i - z_j|)/v_x, \\ (W \times |(C - x_i) - (C - x_j)| + D \times |z_i - z_j|)/v_x \end{pmatrix}, \left(H \times |y_i - y_j|\right)/v_y\right) & \text{if } z_i \neq z_j \end{cases} \tag{2}$$

The first expressions of each of the above two equations denote that the two storage units are the same SR because $z_i = z_j$. The second expressions in each of the equations denote that two storage units are not the same SR because $z_i \neq z_j$, so we need to first compute the minimum distance that the S/R machine drives between two ends of a picking aisle to compute travel distance and travel time.

Definition 3. The execution time of each task must be less than or equal to the specified scheduling time T_r. If it is larger than T_r, it will affect the scheduling quality by an amount equal to the product of the time exceeding T_r and the weight coefficient w_r.

2.1. Optimization

Next, the mathematic model of the automated warehouse scheduling problem is formulated as a multi-objective optimization problem. Suppose the warehouse throughput capacity is Q, and the number of products in the warehouse is q. The automated warehouse scheduling problem has two objectives: the scheduling quality effect should be minimized, and the travel distance should be minimized. The two objectives are defined as follows.

$$\min f(e) = \min \; (f_1(e), f_2(e)) \tag{3}$$

$$f_1(e) = \sum_{r=1}^{max\{\lceil m/N \rceil, \lceil n/N \rceil\}} \left(max\left\{0, \left(\sum_{i=1}^{m+n} \sum_{j=1}^{m+n} t_{ij} e_{ij} g_{ir} g_{jr} \right) - T_r \right\} \cdot w_r \right) \cdot l_r \tag{4}$$

$$f_2(e) = \sum_{r=1}^{max\{\lceil m/N \rceil, \lceil n/N \rceil\}} \left(\sum_{i=1}^{m+n} \sum_{j=1}^{m+n} d_{ij} e_{ij} g_{ir} g_{jr} \right) \cdot l_r \tag{5}$$

where (3) denotes that two objectives are to be minimized, (4) denotes the scheduling quality effect, and (5) denotes the travel distance. Furthermore, the solution must satisfy the following constraints.

Total number of routes

$$\sum_{r=1}^{max\{\lceil m/N\rceil,\lceil n/N\rceil\}} l_r = max\{\lceil m/N\rceil, \lceil n/N\rceil\} \quad (6)$$

Total number of storage and outgoing products

$$\sum_{r=1}^{max\{\lceil m/N\rceil,\lceil n/N\rceil\}} \sum_{j=1}^{m+n} g_{jr} l_r = m+n \quad (7)$$

Each storage or outgoing product is handled exactly once

$$\sum_{r=1}^{max\{\lceil m/N\rceil,\lceil n/N\rceil\}} g_{jr} = 1, \quad \text{for } j \in \{1, 2, \ldots, m+n\} \quad (8)$$

S/R machine load

$$\sum_{j}^{m+n} g_{jr} \le 2N, \quad \text{for } r \in \{1, \ldots, max\{\lceil m/N\rceil, \lceil n/N\rceil\}\} \quad (9)$$

Input location

$$\sum_{r=1}^{max\{\lceil m/N\rceil,\lceil n/N\rceil\}} e_{0j} l_r = \lceil m/N\rceil, \quad \text{for } j \in \{1, 2, \ldots, m+n\} \quad (10)$$

Output location

$$\sum_{r=1}^{max\{\lceil m/N\rceil,\lceil n/N\rceil\}} e_{i0} l_r = \lceil n/N\rceil, \quad \text{for } i \in \{1, 2, \ldots, m+n\} \quad (11)$$

Storage product priorities

$$p_u \in p_{oj} \quad for\ r\left(p_i, p_j, \ldots, p_u\right) \quad (12)$$

Throughput capacity

$$q+m+n \le Q \quad (13)$$

In the above constraints, (6) constrains the number of S/R machine routes to the total number of tasks, (7) constrains the total number of storage and outgoing products, (8) constrains each storage and outgoing product to exactly one route, (9) constrains the total number of storage and outgoing products to no more than twice the S/R machine capacity each route, (10) and (11) constrain the I/O location, where the output location is the same as the input location since the S/R machine returns to the input location after each task is completed, (12) constrains storage products to be handled before outgoing products, and (13) constrains the number of products to be no more than the throughput capacity of the warehouse.

Now that a warehouse model and scheduling problem have been presented, the following section develops the optimization algorithm that will be used to solve the warehouse scheduling problem.

3. Ensemble multi-objective biogeography-based optimization

This section first reviews BBO (Section 3.1), then reviews several MBBO algorithms, including VEBBO, NSBBO, and NPBBO (Section 3.2), and finally introduces the new EMBBO algorithm (Section 3.3).

3.1. Biogeography-based optimization

BBO is an evolutionary algorithm inspired by biogeography to solve general optimization problems. Each candidate solution is comprised of a set of features, which are similar to genes in genetic algorithms (GAs), and which are also called independent variables or decision variables in the optimization literature. Like other evolutionary algorithms, BBO probabilistically shares information between candidate solutions to improve candidate solution fitness. In BBO, each candidate solution immigrates decision variables from other candidate solutions based on its immigration rate, and emigrates decision variables to other candidate solutions based on its emigration rate. In the original BBO paper (Simon, 2008), immigration rates are first used to decide whether or not to immigrate decision variables for a given candidate solution. Then, if immigration is selected for a given candidate solution, emigration rates are used to choose the emigrating candidate solution. Migration can be expressed as

$$a_i(s) \leftarrow a_j(s) \quad (14)$$

where a_i denotes the immigrating solution, a_j denotes the emigrating solution, and s denotes a decision variable. In BBO, each candidate solution a has its own immigration rate λ and emigration rate μ. A good candidate solution has relatively high μ and lowλ, while the converse is true for a poor candidate solution. The immigration rate and the emigration rate are functions of candidate solution fitness. According to Simon (2008), these functions can be calculated as

$$\begin{aligned} \lambda &= 1 - f(a) \\ \mu &= f(a) \end{aligned} \quad (15)$$

where f denotes candidate solution fitness, which is normalized to the range [0, 1]. The probability of immigrating to a_i and the probability of emigrating from a_j are calculated, respectively, as

$$\begin{aligned} &\text{Prob(immigration to } a_i) = \lambda_i \\ &\text{Prob(emigration from } a_j) = \frac{\mu_j}{\sum_{k=1}^{N_0} \mu_k} \end{aligned} \quad (16)$$

where N_0 is the population size. After migration, probabilistic mutation occurs for each decision variable of each candidate solution. Mutation of candidate solution a_i is implemented as follows.

```
For each candidate solution decision variable s in a_i
  If rand(0, 1) < θ
    a_i(s) ← rand(L_s, U_s)
  End if
Next s
```

In the above mutation logic, $rand(L_s, U_s)$ is a uniformly distributed random number between L_s and U_s, θ is the mutation probability, and L_s and U_s are the lower and upper search bounds of the sth independent variable, respectively. The above logic mutates each independent variable with a probability of θ. If mutation occurs for a given independent variable, the independent variable is replaced with a random number within its search domain. A description of one generation of BBO is given in Algorithm 1.

Algorithm 1. One generation of the BBO algorithm. a_i is the ith candidate solution, and $a_i(s)$ is the sth decision variable of a_i.

Biogeography-based optimization (BBO) algorithm

```
For each a_j, define emigration rate μ_j proportional to fitness of
  a_j, with μ_j ∈ [0, 1]
For each a_j, define immigration rate λ_j = 1 − μ_j
For each a_i
  For each candidate solution decision variable s
    Use λ_i to probabilistically decide whether to immigrate to
  a_i (Eq. (16))
    If immigrating then
      Use {μ} to probabilistically select the emigrating solution
  a_j (Eq. (16))
      a_i(s) ← a_j(s)
    End if
    Next candidate solution decision variable
    Probabilistically decide whether to mutate a_i (see mutation
  logic following Eq. (16))
```

Biogeography-based optimization (BBO) algorithm

Next candidate solution

In Algorithm 1, the statement "use λ_i to probabilistically decide whether to immigrate to a_i" can be implemented with the following logic.

```
If λ_i < rand(0,1) then
  Immigrate
Else
  Do not immigrate
End if
```

In Algorithm 1, the statement "Use $\{\mu\}$ to probabilistically select the emigrating solution a_j" can be implemented with any fitness-based selection method since μ is proportional to the fitness of a. For instance, tournament selection could be used by randomly choosing two or more solutions for a tournament, and then selecting a_j as the fittest solution in the tournament. In this paper, as in most other BBO implementations, $\{\mu\}$ is used in a roulette-wheel algorithm so that the probability that each individual a_j is selected for emigration is proportional to its emigration rate μ_j.

3.2. Multi-objective BBO

This section is based on Simon (2013, Section 20.5). First, this section reviews the VEBBO algorithm, which combines BBO with the vector evaluated genetic algorithm (VEGA). Recall that VEGA was one of the original multi-objective evolutionary algorithms (MOEAs), and operates by performing selection on the population using one objective function at a time (Coello et al., 2004; Schaffer, 1985). VEBBO produces a set of subpopulations, one set for each objective function. Then individuals are selected from the subpopulations to obtain parents, which create children by using the BBO migration method. The outline of VEBBO for a k-objective optimization problem is shown in Algorithm 2, where MBBO immigration is based on the $k_i th$ objective function value of each individual, and k_i is a random objective function index at the ithmigration trial. Then emigration is based on the $k_e th$ objective function value of each individual, where k_e is also a random objective function index.

Algorithm 2. Outline of VEBBO for solving an n_0-dimensional optimization problem with k objectives and a population size of N_0. Each generation, the best individual a_b with respect to the ithobjective value has rank $\gamma_{bi} = 1$, and the worst individual a_w has rank $\gamma_{wi} = N_0$.

Vector evaluated biogeography-based optimization (VEBBO) algorithm

Randomly initialize a population of candidate solutions $P = \{a_j\}$ for $j \in [1, N_0]$
While termination criterion is not satisfied **do**
  Compute the cost $f_i(a_j)$ for each objective i and for each individual $a_j \in P$
  For each objective i where $i \in [1, k]$ **do**
    $\gamma_{ji} \leftarrow$ rank of a_j with respect to the ithobjective function for $j \in [1, N_0]$
    Immigration rates $\lambda_{ji} \leftarrow \gamma_{ji} / \sum_{q=1}^{N_0} \gamma_{qi}$ for $j \in [1, N_0], i \in [1, k]$
    Emigration rates $\mu_{ji} \leftarrow 1 - \lambda_{ji}$ for $j \in [1, N_0], i \in [1, k]$
    For each individual a_j where $j \in [1, N_0]$ **do**
      For each independent variable $s \in [1, n_0]$ **do**
        $k_i \leftarrow$ rand(1, k)=uniformly distributed integer between 1 and k
        $\delta \leftarrow$ rand(0, 1)=uniformly distributed real number between 0 and 1
        If $\delta < \lambda_{j,k_i}$ **then**
          $k_e \leftarrow rand(1, k)$=uniformly distributed integer between 1 and k
          Probabilistically select emigrant a_e, where $\Pr(a_e = a_\beta) = \mu_{\beta,k_e} / \sum_{q=1}^{N_0} \mu_{q,k_e}$ for $\beta \in [1, N_0]$
          $a_j(s) \leftarrow a_e(s)$
        End if
      End for
    End for
  End for
  Probabilistically mutate the population P as described in the standard BBO algorithm
End while

Next, the NSBBO algorithm is reviewed, which combines BBO with the non-dominated sorting genetic algorithm (NSGA). Recall that NSGA was one of the original MOEAs, and assigns the cost of each individual based on its dominance level (Deb et al., 2002; Srinivas and Deb, 1995). First, all individuals are copied to a temporary population T. Then we find all non-dominated individuals in T; these individuals, which are denoted as the set B, are assigned the lowest cost value. Recall that an individual x is dominated by an individual y if y performs at least as good as x in all objectives, and performs better than x in at least one objective (Simon, 2013, Chapter 20). An individual is called non-dominated if there are no individuals in the population (T in this case) that dominate it. Next, B is removed from T, and we then find all non-dominated individuals in the reduced set T. These individuals are assigned the second-lowest cost value. This process is repeated to obtain a cost for each individual that is based on its level of non-domination. We combine BBO with NSGA by changing the recombination logic in NSGA to BBO migration operations, which results in NSBBO, as shown in Algorithm 3.

Algorithm 3. Outline of NSBBO for solving an n_0-dimensional optimization problem with k objectives and a population size of N_0.

Non-dominated sorting biogeography-based optimization (NSBBO) algorithm

Randomly initialize a population of candidate solutions $P = \{a_j\}$ for $j \in [1, N_0]$
While termination criterion is not satisfied **do**
  Temporary population $T \leftarrow P$
  Non-domination level $c \leftarrow 1$
  While the temporary population size $|T| > 0$ **do**
    $B \leftarrow$ non-dominated individuals in T
    Cost $f(a) \leftarrow c$ for all $a \in B$
    Remove B from T
    $c \leftarrow c + 1$
  End while
  Immigration rates $\lambda_j \leftarrow f(a_j) / \sum_{q=1}^{N_0} f(a_q)$ for $j \in [1, N_0]$
  Emigration rates $\mu_j \leftarrow 1 - \lambda_j$ for $j \in [1, N_0]$
  For each individual a_j where $j \in [1, N_0]$ **do**

Non-dominated sorting biogeography-based optimization (NSBBO) algorithm

For each independent variable $s \in [1,\ n_0]$ **do**
$\delta \leftarrow$ rand(0, 1)=uniformly distributed real number between 0 and 1
If $\delta < \lambda_j$ **then**
Probabilistically select emigrant a_e, where $\Pr(a_e = a_\beta) = \mu_\beta / \sum_{q=1}^{N_0} \mu_q$ for $\beta \in [1,\ N_0]$
$a_j(s) \leftarrow a_e(s)$
End if
End for
End for
Probabilistically mutate the population P as described in the standard BBO algorithm
End while

Finally, the NPBBO algorithm is reviewed, which combines BBO with the niched Pareto genetic algorithm (NPGA). Recall that NPGA was one of the original MOEAs, which is similar to NSGA in its assignment of cost on the basis of domination (Horn et al., 1994). NPGA is an attempt to reduce the computational effort of NSGA. Two tournament individuals a_1 and a_2 are selected randomly from the population, and then a subset S of the population is also randomly selected, which is typically around 10% of the population. If one of the individuals a_1 and a_2 is dominated by any of the individuals in S, and the other is not, then the non-dominated individual, denoted as a_0, wins the tournament and is selected for recombination. If both individuals a_1 and a_2 are dominated by at least one individual in S, or both individuals are not dominated by any individuals in S, then fitness sharing is used to decide the tournament winner; that is, the individual that is in the least crowded region of the objective function space wins the tournament. This selection process can be described as follows:

$$
\begin{aligned}
&b_i = |y_0 \in S : y_0 \succ a_i| \quad for\ \ i \in [1,2]\\
&c_i = Crowding\ \ distance\ \ of\ \ a_i \quad for\ \ i \in [1,2]\\
&a_0 = \begin{cases} a_1 & if \begin{cases} (b_1 = 0)\ \ and\ \ (b_2 > 0),\ \ or\\ (b_1 > 0)\ \ and\ \ (b_2 > 0)\ \ and\ \ (c_1 < c_2),\ \ or\\ (b_1 = 0)\ \ and\ \ (b_2 = 0)\ \ and\ \ (c_1 < c_2) \end{cases}\\ a_2 & otherwise \end{cases}. \qquad (17)
\end{aligned}
$$

where $y_0 \succ a_i$ denotes that y_0 dominates a_i; that is, y_0 is at least as good as a_i for all objective function values, and it is better than a_i for at least one objective function value. b_i is the number of individuals that dominate a_i, c_i is the crowding distance of a_i, and a_0 is the individual (either a_1 or a_2) that is finally selected for recombination. The crowding distance c could be computed with a method from Simon (2013, pp. 541). For each objective function dimension, the closest larger value and the closest smaller value in the population are found as follows:

$$
\begin{aligned}
f_j^-(a) &= \max_{a^*}\left\{f_j(a^*)\ \ such\ \ that\ \ f_j(a^*) < f_j(a)\right\}\\
f_j^+(a) &= \min_{a^*}\left\{f_j(a^*)\ \ such\ \ that\ \ f_j(a^*) > f_j(a)\right\} \qquad (18)
\end{aligned}
$$

where $f_j(a)$ is the objective function value of a with respect to the jth objective. The crowding distance of a is then computed as

$$
c = \sum_{j=1}^{k} \left(f_j^+(a) - f_j^-(a)\right) \qquad (19)
$$

We combine BBO with NPGA by changing the recombination logic in NPGA to BBO migration operations, which results in NPBBO, as shown as Algorithm 4.

Algorithm 4. Outline of NPBBO for solving an n_0-dimensional optimization problem with k objectives and a population size of N_0.

Niched Pareto biogeography-based optimization (NPBBO) algorithm

Randomly initialize a population of candidate solutions $P = \{a_j\}$ for $j \in [1,\ N_0]$
While termination criterion is not satisfied **do**
Temporary population $T \leftarrow \phi$
While the temporary population size $|T| < N_0$ **do**
Randomly select two individuals a_1 and a_2 from P
Randomly select a population subset $S \subset P$
Use Eq. (17) to select a_0 from $\{a_1,\ a_2\}$
$T \leftarrow \{T,\ a_0\}$
End while
For each individual $a_j \in T$, where $j \in [1,\ N_0]$ **do**
For each independent variable $s \in [1,\ n_0]$ **do**
$\delta \leftarrow$ rand(0, 1)=uniformly distributed real number between 0 and 1
If $\delta < 1/N_0$ **then**
Probabilistically select emigrant a_e, where $\Pr(a_e = x_\beta) = 1/N_0$ for $\beta \in [1,\ N_0]$
$a_j(s) \leftarrow a_e(s)$
End if
End for
End for
Probabilistically mutate the population P as described in the standard BBO algorithm
End while

3.3. Ensemble multi-objective BBO

Now the ensemble multi-objective BBO (EMBBO) algorithm is proposed, which is an ensemble of the previously-discussed VEBBO, NSBBO, and NPBBO algorithms. The proposed approach is motivated as follows. According to the no free lunch (NFL) theorem, it is impossible for a single optimization algorithm to outperform all other algorithms on every problem. Each problem has its own characteristics, such as the search space, constraint conditions, objective function landscape, and so on. Different problems require different algorithms, depending on the nature of the problem and available computing resources. In addition, for a given problem, each optimization algorithm may perform differently during different phases of the search process. For example, an algorithm with better exploration may have better search ability during the initial phase of the optimization process, and another algorithm with better exploitation may have better search ability during the later phases of the optimization process. Hence, different optimization algorithms may be suitable during different stages of the search process.

Motivated by these observations, an ensemble of MBBO algorithms composed of VEBBO, NSBBO and NPBBO is implemented in parallel populations in the EMBBO algorithm. The best individuals among the parallel populations are used to yield improved performance using multiple MBBO algorithms. In this paper, three populations are used based on the three previously-discussed MBBO algorithms. More populations can be used if additional MBBO algorithms are used. Each population generates its own offspring. All of the offspring populations are combined to select a fixed number of the best individuals, which updates all of the parent populations. In this way, EMBBO always keeps the individuals that are generated by the more suitable MBBO algorithm, leading to performance that is better than any individual MBBO

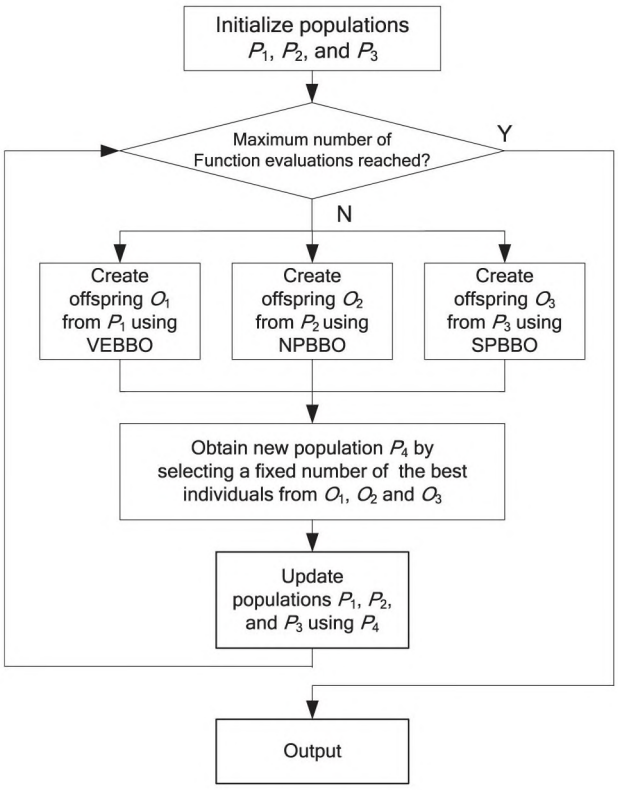

Fig. 2. Flowchart of EMBBO, with three parallel populations P_1, P_2, and P_3. P_4 is the updating population, and O_1, O_2, and O_3 are offspring populations.

algorithm. The flowchart of EMBBO is shown in Fig. 2, where there are three parallel populations, although less or more could be used, depending on the application. The basic procedure of EMBBO is summarized from Fig. 2 as follows.

Step 1. Randomly initialize each parallel parent population P_1, P_2, and P_3. Use the same population size for each parallel parent population, so each parallel parent population is one-third of the total EMBBO population size.
Step 2. Create offspring populations O_1, O_2, and O_3, from P_1, P_2, and P_3, respectively, using VEBBO, NSBBO, and NPBBO.
Step 3. Obtain population P_4 by selecting a fixed number of the best individuals from the union of O_1, O_2, and O_3. First, O_1, O_2, and O_3 are combined into a single population, and then the $\dot{N}$ best individuals are selected from the combined population using non-dominated sorting, where $\dot{N}$ is the population size of P_1 (which is also the population size of P_2 and P_3). That is, the best individuals are iteratively added to the new population P_4 based on levels of non-domination until the desired population size $\dot{N}$ is reached.
Step 4. Update the parallel parent populations P_1, P_2, and P_3, with P_4. Each parent population is completely replaced by the new population P_4. That is, P_1, P_2, and P_3 are composed of the same individuals after this replacement. Their offspring will be different in the following generation because of the different algorithms in Step 2.
Step 5. If the termination criterion is not met, go to Step 2; otherwise, terminate. Here the termination criterion is the maximum number of function evaluations, and P_4 is the output of EMBBO after termination.

4. Simulations and results

In this section the performance of the proposed EMBBO algorithm is investigated on a set of unconstrained and constrained multi-objective benchmark functions, and on the automated warehouse scheduling problem. Section 4.1 discusses the simulation setup, Section 4.2 presents performance comparisons on the 2009 CEC benchmark functions, and Section 4.3 applies EMBBO to the automated warehouse scheduling problem.

4.1. Simulation setup

The performance of EMBBO and its constituent algorithms, including VEBBO, NPBBO, and SPBBO, is evaluated on a set of 10 unconstrained functions and 10 constrained functions, which are taken from the CEC 2009 benchmark set. These functions are briefly summarized in Table A1 in Appendix A, where U01–U10 are unconstrained multi-objective benchmark functions, and C01–C10 are constrained multi-objective benchmark functions. U01–U07 and C01–C07 are two-objective problems, and U08–U10 and C08–C10 are three-objective problems. The constrained multi-objective benchmark functions include one or two inequality constraints. The complete definition of each function is available in the literature (Zhang, et al., 2008).

For all algorithms, population size and mutation rate have to be determined. In the literature (Simon, 2008) these parameters have been discussed in detailed. We use a population size of 50 for each parallel constituent algorithm of EMBBO, so the total EMBBO population size is 150 because there are three parallel populations. To obtain fair comparisons, a population size for each individual MBBO (when running by itself apart from EMBBO) is set to 150. The mutation rate is set to 0.01 per solution decision variable per generation. If mutation occurs, the mutated value of the new independent variable is uniformly distributed in the search space. We evaluate each algorithm 30 times, with a maximum number of function evaluations equal to 300,000 for each simulation.

Hypervolume is used as the performance metric. Suppose that a MOEA has found M points in an approximate Pareto front $\hat{P}_f = \{f(a_j)\}$ for $j \in [1, M]$, where $f(a_j)$ is a k-objective function. The hypervolume can be computed as

$$Y\left(\hat{P}_f\right) = \sum_{j=1}^{M} \prod_{i=1}^{k} f_i(a_j) \tag{20}$$

Given two algorithms that compute two Pareto front approximations to a given MOP, we can use hypervolume to quantify how good the two approximations are relative to each other. For a minimization problem, a smaller hypervolume indicates a better Pareto front approximation.

EMBBO is also compared with the 5 best algorithms from the 2009 CEC competition for unconstrained benchmark functions, including MOEAD, MTS, DMOEADD, LiuLiAlgorithm, and GDE3; and with the 3 best algorithms from the 2009 CEC competition for constrained benchmark functions, including DMOEADD, LiuLiAlgorithm, and MTS. These are the best algorithms from the 13 accepted algorithms in the 2009 CEC competition (Suganthan, 2014).

(a) MOEAD is based on decomposition (Chen, et al., 2009).
(b) MTS is a multiple trajectory search algorithm (Tseng and Chen, 2009).
(c) DMOEADD is an improvement of the dynamic multi-objective evolutionary algorithm and is based on domain decomposition (Liu et al., 2009).
(d) LiuLi-Algorithm is based on sub-regional search (Liu and Li, 2009).
(e) GDE3 is the third version of a generalized differential evolution algorithm with a diversity maintenance technique (Kukkonen and Lampinen, 2009).

In addition, a Holm multiple comparison test determines statistically significant differences between EMBBO used as the

control method, and the other algorithms. The Holm multiple comparison test is a nonparametric statistical test that obtains a probability (p-value) that determines the degree of difference between a control algorithm and a set of alternative algorithms, assuming that the algorithms have statistically significant differences as a whole. To determine whether a set of algorithms shows a statistically significant difference as a whole, Friedman's test is used with a significance level $\alpha=0.1$ to the mean error rankings. If the test rejects the null hypothesis that all of the algorithms perform similarly, the best algorithm is considered as the control method and is then compared with the other algorithms according to their rankings. Additional details about the Holm multiple comparison test can be found in the literature (Derrac et al., 2011).

4.2. Performance comparisons

Tables A2 and A3 in Appendix A summarize the performance comparison of EMBBO, VEBBO, NPBBO, SPBBO, and the 2009 CEC competition algorithms on each benchmark function with respect to the hypervolume and normalized hypervolume. Note that hypervolume cannot be blindly used as an indicator of Pareto front quality because if M is larger, then S becomes larger also. But that means a Pareto front with more points would be worse than a Pareto front with fewer points. Normalization takes the number of Pareto front points into account.

For the comparison of EMBBO with its constituent algorithms, Table A2 shows that for unconstrained benchmark functions, EMBBO performs best on 7 functions (U01, U03, U04, U05, U06, U07 and U09), and NPBBO performs best on 3 functions (U02, U08 and U10). Table A3 shows that for constrained benchmark functions, EMBBO performs best on 6 functions (C01, C03, C04, C06, C07 and C09), NPBBO performs best on 3 functions (C02, C05 and C10), and VEBBO performs best on C08. The proposed EMBBO performs significantly better than the constituent MBBO algorithms for both the unconstrained and constrained multi-objective benchmark functions.

For the comparison of EMBBO with the 2009 CEC competition algorithms, Table A2 shows that for unconstrained benchmark functions, MOEAD performs best on 5 functions (U01, U02, U04, U09, and U10), MTS performs best on 3 functions (U05, U06, and U08), and EMBBO performs best on 2 functions (U03 and U07). Table A3 shows that for constrained benchmark functions, DMOEADD performs best on 7 functions (C01, C02, C05, C06, C07, C08 and C10), and EMBBO performs best on 3 functions (C03, C04 and C09). This indicates that for unconstrained benchmark functions, MOEAD is the best, MTS is the second best and EMBBO is the third best. For constrained benchmark functions, DMOEADD is the best and EMBBO is the second best.

The empirical results show that EMBBO is a competitive algorithm for multi-objective benchmark functions. The reasons for its competitive performance are that, first, BBO has distinctive migration behavior compared to other EAs. Second, EMBBO effectively uses ensemble learning. That is, EMBBO uses different MBBO algorithms in multiple parallel populations to create different offspring populations throughout the search process. By comparing the multiple offspring populations, the best individuals are retained in EMBBO. The average running times of the algorithms are shown in the last row of Tables A2 and A3 in Appendix A. The algorithms are programmed in MATLAB® on a 2.40 GHz Intel Pentium® 4 CPU with 4 GB of memory. From the tables, we see that the average running time of EMBBO is much less than the constituent MBBO algorithms. The reason is that EMBBO uses multiple parallel subpopulations of relatively small size, while the constituent MBBO algorithms use a single population that is three times as large as that of EMBBO. Certain EA operations, such as roulette-wheel selection, require computational effort on the order of N_0^2, where N_0 is the population size, and so multiple subpopulations are more computationally efficient than a single large population. Multiple subpopulations are also amenable to parallel processing, which can further reduce computational effort.

Table 1 shows the results of the Holm multiple comparison test between EMBBO and its constituent algorithms and the 2009 CEC competition algorithms. For unconstrained functions EMBBO is the third best, and for constrained functions it is the second best. EMBBO is compared with its constituent algorithms and the 5 best CEC competition algorithms for unconstrained optimization, and with its constituent algorithms and the 3 best CEC competition algorithms for constrained optimization. Note that Table 1 shows that EMBBO is outperformed by MOEAD and MTS for unconstrained functions and by DMOEADD for constrained functions, as indicated by p-values smaller than 0.1. Table 1 also shows that EMBBO is statistically significantly better than its constituent algorithms VEBBO, NPBBO, and SPBBO.

4.3. Application to automated warehouse scheduling

In this section, the proposed EMBBO is applied to the automated warehouse scheduling problem from Section 2. The width, length and height of each storage unit are $W=0.3$ m, $L=0.5$ $m$and $H=0.4$ m, the distance between adjacent SRs is $D=1.8$ m, the number of storage units in each SR is $C=75$, the warehouse throughput capacity is $Q=600$, and the number of products in the warehouse is $q=250$. The S/R machine capacity is $N=4$, its horizontal velocity is $v_x=1$ m/s, and its vertical velocity is $v_y=0.5$ m/s. The required time for each task is the same, which is 120 s. All these parameters are taken from a real-world automated warehouse scheduling problem. We consider five implementation schemes. Scheme 1 (number of storage products $m=20$, and number of outgoing products $n=20$) is described as follows.

$p_1:(12,6,2,50,1)$, $p_2:(51,8,1,45,1)$, $p_3:(40,3,5,46,1)$, $p_4:(37,3,4,113,1)$,
$p_5:(14,7,1,40,1)$, $p_6:(13,5,2,50,1)$, $p_7:(45,5,5,50,1)$, $p_8:(18,7,3,33,1)$,
$p_9:(7,3,2,35,1)$, $p_{10}:(11,4,3,110,1)$, $p_{11}:(15,2,1,34,1)$, $p_{12}:(36,8,4,40,1)$,
$p_{13}:(8,6,5,47,1)$, $p_{14}:(56,2,3,34,1)$, $p_{15}:(42,8,3,30,1)$, $p_{16}:(23,4,1,50,1)$,
$p_{17}:(4,6,2,50,1)$, $p_{18}:(13,1,1,36,1)$, $p_{19}:(39,6,4,42,1)$, $p_{20}:(45,6,5,55,1)$
$p_{21}:(50,8,1,67,2)$, $p_{22}:(3,1,2,74,2)$, $p_{23}:(55,4,3,68,2)$, $p_{24}:(6,7,4,85,2)$,
$p_{25}:(17,4,5,74,2)$, $p_{26}:(60,7,3,80,2)$, $p_{27}:(35,6,2,67,2)$, $p_{28}:(15,2,1,70,2)$,
$p_{29}:(57,4,2,80,2)$, $p_{30}:(2,1,4,62,2)$, $p_{31}:(25,8,2,76,2)$, $p_{32}:(5,2,4,64,2)$,
$p_{33}:(17,5,3,76,2)$, $p_{34}:(41,2,5,91,2)$, $p_{35}:(19,3,2,70,2)$, $p_{36}:(20,1,1,82,2)$,
$p_{37}:(42,6,2,88,2)$, $p_{38}:(32,6,3,75,2)$, $p_{39}:(9,7,1,82,2)$, $p_{40}:(58,5,4,69,2)$

Each storage or retrieval operator is denoted by (x, y, z, w, u), where x, y, and z are the Euclidean coordinates of each storage unit (meters), w is the weighting coefficient described in Section 2 which affects the scheduling quality, and $u=1$ indicates a storage product and $u=2$ indicates an outgoing product.

Table 1
Holm multiple comparison test results of EMBBO and its constituent algorithms and 2009 CEC competition algorithms.

Unconstrained functions			Constrained functions		
Algorithm	Rank	p-value	Algorithm	Rank	p-value
MOEAD	1.4	0.0076	DMOEADD	1.3	0.0198
MTS	2.6	0.0227	EMBBO	2.7	–
EMBBO	3.7	–	LiuLiAlgorithm	3.7	0.0402
DMOEADD	4.1	0. 0910	NPBBO	3.8	0.0421
NPBBO	4.3	0.0613	MTS	4.7	0.0094
LiuLiAlgorithm	5.8	0.0086	VEBBO	4.8	0.0086
GDE3	6.6	0.0021	NSBBO	7.0	0.0007
VEBBO	7.2	0.0010			
NSBBO	8.8	0.0008			

Table 2
MBBO results for 5 schemes of the automated warehouse scheduling problem. "Distance" denotes the shortest travel distance, which is measured in meters, and "Effect" denotes the lowest scheduling quality effect. The best results in each row are shown in **boldface** font.

Problem	(m, n)	VEBBO		NSBBO		NPBBO		EMBBO	
		Distance	Effect	Distance	Effect	Distance	Effect	Distance	Effect
Scheme 1	(20, 20)	129.3	2591.3	122.2	2504.3	124.3	2555.3	**120.3**	**2175.6**
Scheme 2	(20, 16)	92.5	1531.0	98.3	1562.7	89.7	1453.6	**77.2**	**1308.5**
Scheme 3	(20, 12)	71.0	826.2	63.2	857.7	62.3	798.7	**57.1**	**756.8**
Scheme 4	(16, 20)	109.4	1823.8	97.6	1467.9	**88.0**	**1295.5**	88.9	1394.0
Scheme 5	(12, 20)	69.8	840.0	60.5	839.4	67.7	908.6	**58.1**	**836.0**

Table 3
Scheduling orders as optimized by the proposed EMBBO algorithm. "Route" denotes the route number index, and "scheduling orders" denote the scheduling orders that the S/R machine implements each route.

Route	Scheduling orders
1	$p_6 \to p_{12} \to p_3 \to p_7 \to p_{34} \to p_{23} \to p_{31} \to p_{24}$
2	$p_{16} \to p_8 \to p_{20} \to p_2 \to p_{26} \to p_{35} \to p_{28} \to p_{22}$
3	$p_{13} \to p_{10} \to p_{11} \to p_{15} \to p_{29} \to p_{40} \to p_{36} \to p_{30}$
4	$p_9 \to p_1 \to p_{18} \to p_4 \to p_{37} \to p_{21} \to p_{25} \to p_{39}$
5	$p_{17} \to p_5 \to p_{19} \to p_{14} \to p_{27} \to p_{38} \to p_{33} \to p_{32}$

Table 4
Comparisons between EMBBO, DMOEADD, LiuLiAlgorithm, and MTS for the automated warehouse scheduling problem. "Distance" denotes the shortest travel distance, which is measured in meters, and "Effect" denotes the lowest scheduling quality effect. The best results in each row are shown in **boldface** font.

Problem	(m, n)	DMOEADD		LiuLiAlgorithm		MTS		EMBBO	
		Distance	Effect	Distance	Effect	Distance	Effect	Distance	Effect
Scheme 1	(20, 20)	**114.4**	**2047.5**	130.1	2314.6	136.7	2411.5	120.3	2175.6
Scheme 2	(20, 16)	84.5	1612.2	89.4	1639.1	92.1	1694.3	**77.2**	**1308.5**
Scheme 3	(20, 12)	**50.3**	**721.5**	62.4	779.2	62.9	801.6	57.1	756.8
Scheme 4	(16, 20)	91.5	1486.1	93.4	1507.6	99.1	1611.2	**88.9**	**1394.0**
Scheme 5	(12, 20)	**52.4**	**811.9**	61.0	853.7	62.7	892.3	58.1	836.0

Scheme 2 ($m = 20$, $n = 16$) includes all the storage products of Scheme 1 but only the first 16 outgoing products of Scheme 1. Scheme 3 ($m = 20$, $n = 12$) includes all the storage products of Scheme 1 but only the first 12 outgoing products. Scheme 4 ($m = 16$, $n = 20$) includes the first 16 storage products of Scheme 1 and all of the outgoing products. Scheme 5 ($m = 12$, $n = 20$) includes the first 12 storage products of Scheme 1 and all of the outgoing products.

The tuning parameters of the MBBO algorithms are the same as those used in the benchmark simulations. The optimization results are summarized in Table 2. It is seen from Table 2 that EMBBO performs best for all of the schemes except Scheme 4, for which NPBBO is the best because of its shortest travel distance and its lowest scheduling quality effect. Based on these results, it is concluded that the proposed EMBBO algorithm generally provides better performance than the single-constituent MBBO algorithms for our automated warehouse scheduling problem.

A sample EMBBO scheduling route output is shown in Table 3 for Scheme 1. It is seen that the automated warehouse scheduling problem is divided into 5 routes, and each route includes 8 storage units, where the first 4 storage units are used to store products, and the last 4 storage units are used to retrieve products.

Next we compare EMBBO with the 3 best constrained optimization algorithms from the CEC 2009 competition, which include DMOEADD, LiuLiAlgorithm, and MTS. The optimization results are summarized in Table 4. It is seen that DMOEADD performs best on 3 cases (Scheme 1, Scheme 3, and Scheme 5), and EMBBO performs best on the other 2 cases (Scheme 2 and Scheme 4). In the 3 schemes for which DMOEADD performed best, EMBBO performed second best. Although EMBBO is not the best algorithm, it is a consistently competitive algorithm for the automated warehouse scheduling problem.

5. Conclusions

In this paper an ensemble multi-objective biogeography-based optimization algorithm called EMBBO was proposed to solve general multi-objective optimization problems, both constrained and unconstrained. In addition, an automated warehouse scheduling model was built based on common industrial warehouse characteristics, and the scheduling problem was formulated as a constrained multi-objective optimization problem.

The performance of EMBBO was investigated on a set of CEC 2009 multi-objective benchmark functions. The numerical simulations showed that EMBBO is better than single-constituent MBBO algorithms for the most of the benchmark functions. In particular, EMBBO is better than its constituent algorithm on 7 of 10 unconstrained benchmarks, and 6 of 10 constrained benchmarks. Furthermore, EMBBO is competitive with the best CEC 2009 algorithms for multi-objective optimization. In particular, EMBBO is better than the best CEC 2009 algorithms on 5 of 10 unconstrained benchmarks, and 3 of 10 constrained benchmarks.

Finally, EMBBO was applied to the automated warehouse scheduling problem and the results again showed that EMBBO is a competitive optimization method. In particular, EMBBO was better than its constituent algorithms on 4 of 5 warehouse scheduling problems, and it was better than the best CEC 2009 algorithms on 2 of 5 warehouse scheduling problems.

Based on the optimization tests in this paper, EMBBO can effectively improve the performance of MBBO on multi-objective optimization problems, including real-world warehouse scheduling. The fundamental contributions of this paper are twofold. First, multiple MBBO algorithms are combined into a single algorithm, leveraging the best features of multiple algorithms. The framework presented here could be extended for other types of optimization algorithms also. Second, a real-world warehouse scheduling problem was formulated in a way that is amenable to multi-objective evolutionary algorithms.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant no. 1344954, the National Natural Science Foundation of China under Grant nos. 61305078, 61074032 and 61179041, and the Shaoxing City Public Technology Applied Research Project under Grant no. 2013B70004.

Appendix A

The abbreviations, notations, and symbols used in this paper are summarized as follows:

BBO: biogeography-based optimization
EMBBO: ensemble multi-objective biogeography-based optimization
MBBO: multi-objective biogeography-based optimization
VEBBO: vector evaluated biogeography-based optimization
NSBBO: non-dominated sorting biogeography-based optimization
NPBBO: niched Pareto biogeography-based optimization
GA: genetic algorithm
VEGA: vector evaluated genetic algorithm
NSGA: non-dominated sorting genetic algorithm
NPGA: Niched Pareto genetic algorithm
ES: evolution strategies
EP: evolutionary programming
HS: harmony search
DE: differential evolution
MOEA: multi-objective evolutionary algorithm
MOP: multi-objective optimization problem
CEC: Congress on Evolutionary Computation
AS/RS: automated storage and retrieval systems
S/R: storage and retrieval
SR: storage rack
I/O: input/output
$rand(\omega,\sigma)$: uniformly distributed random number between ω and σ
$\lceil\varphi\rceil$: smallest integer greater than or equal to φ
B: population of non-dominated individuals
C: number of storage units in each SR
D: distance between adjacent SRs
H: height of each storage unit
N: number of products S/R machine bears
N_0: population size
O: offspring population
P: population of candidate solutions
Q: warehouse throughput capacity

Table A1
CEC 2009 multi-objective benchmark functions, where n_0 denotes the number of dimensions in each objective. More detailed about these functions can be found in Zhang, et al. (2008).

Function name	Num. of objectives	Search space
Unconstrained multi-objective benchmark functions		
U01: Unconstrained problem 1	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U02: Unconstrained problem 2	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U03: Unconstrained problem 3	2	$[0,1]^{n_0}$, $n_0=30$
U04: Unconstrained problem 4	2	$[0,1]\times[-2,2]^{n_0-1}$, $n_0=30$
U05: Unconstrained problem 5	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U06: Unconstrained problem 6	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U07: Unconstrained problem 7	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=30$
U08: Unconstrained problem 8	3	$[0,1]^2\times[-2,2]^{n_0-2}$, $n_0=30$
U09: Unconstrained problem 9	3	$[0,1]^2\times[-2,2]^{n_0-2}$, $n_0=30$
U10: Unconstrained problem 10	3	$[0,1]^2\times[-2,2]^{n_0-2}$, $n_0=30$
Constrained multi-objective benchmark functions		
C01: Constrained problem 1	2	$[0,1]^{n_0}$, $n_0=10$
C02: Constrained problem 2	2	$[0,1]\times[-1,1]^{n_0-1}$, $n_0=10$
C03: Constrained problem 3	2	$[0,1]\times[-2,2]^{n_0-1}$, $n_0=10$
C04: Constrained problem 4	2	$[0,1]\times[-2,2]^{n_0-1}$, $n_0=10$
C05: Constrained problem 5	2	$[0,1]\times[-2,2]^{n_0-1}$, $n_0=10$
C06: Constrained problem 6	2	$[0,1]\times[-2,2]^{n_0-1}$, $n_0=10$
C07: Constrained problem 7	2	$[0,1]\times[-2,2]^{n_0-1}$, $n_0=10$
C08: Constrained problem 8	3	$[0,1]^2\times[-4,4]^{n_0-2}$, $n_0=10$
C09: Constrained problem 9	3	$[0,1]^2\times[-2,2]^{n_0-2}$, $n_0=10$
C10: Constrained problem 10	3	$[0,1]^2\times[-2,2]^{n_0-2}$, $n_0=10$

S: subset of the population
T: temporary population
T_r: specified scheduling time
W: width of each storage unit
Y: hypervolume
L_s: lower search bounds of the *sth* independent variable
U_s: upper search bounds of the *sth* independent variable
a: candidate solution
a_0: non-dominated individual
b_i: number of individuals that dominate a_i
c: non-domination level
c_i: crowding distance of a_i
d_{ij}: travel distance of S/R machine
f: candidate solution fitness
k: number of objectives
l, g, e: flag sign
m: number of storage products
n: number of outgoing products
n_0: optimization problem dimension
p: storage unit
q: number of products in the warehouse
r: route S/R machine goes through
s: decision variable
t_{ij}: time of S/R machine operation
v_x: horizontal velocity
v_y: vertical velocity
w_r: weight coefficient
x, y, z: Euclidean coordinates

Table A2
MBBO and CEC competition algorithm results for 10 unconstrained multi-objective benchmark functions. The table shows the relative hypervolume and normalized relative hypervolume. The best results with respect to MBBO in each row are shown in **boldface** font. The best results among all algorithms are underlined. Average CPU times are shown in the last row of the table.

Fun.	MBBO				CEC competition algorithms				
	VEBBO	NSBBO	NPBBO	EMBBO	MOEAD	MTS	DMOEADD	LiuLiAlgorithm	GDE3
U01	(58.59, 0.133)	(73.61, 0.167)	(51.56, 0.117)	**(40.48, 0.092)**	(37.47, 0.085)	(39.45, 0.089)	(44.36, 0.101)	(47.59, 0.108)	(47.35, 0.107)
U02	(24.39, 0.123)	(30.58, 0.155)	**(17.37, 0.088)**	(20.85, 0.106)	(17.16, 0.087)	(17.85, 0.091)	(22.44, 0.113)	(22.82, 0.115)	(23.78, 0.120)
U03	(288.9, 0.122)	(398.4, 0.168)	(232.9, 0.098)	**(202.1, 0.085)**	(203.1, 0.086)	(230.9, 0.097)	(247.3, 0.104)	(280.2, 0.118)	(294.6, 0.124)
U04	(8.549, 0.109)	(11.45, 0.146)	(8.373, 0.107)	**(7.900, 0.100)**	(7.413, 0.095)	(8.070, 0.103)	(9.334, 0.119)	(8.940, 0.114)	(8.311, 0.106)
U05	(286.7, 0.192)	(282.9, 0.190)	(130.8, 0.088)	**(117.1, 0.079)**	(110.7, 0.074)	(102.8, 0.069)	(135.6, 0.091)	(155.2, 0.104)	(167.1, 0.112)
U06	(697.5, 0.145)	(795.4, 0.169)	(485.4, 0.103)	**(477.2, 0.101)**	(425.5, 0.090)	(416.3, 0.088)	(446.9, 0.095)	(438.5, 0.093)	(532.2, 0.113)
U07	(55.72, 0.124)	(57.96, 0.129)	(40.51, 0.090)	**(40.40, 0.089)**	(42.03, 0.093)	(45.91, 0.102)	(51.34, 0.114)	(60.86, 0.135)	(55.34, 0.123)
U08	(550.4, 0.111)	(763.2, 0.153)	**(428.8, 0.086)**	(633.5, 0.127)	(456.7, 0.092)	(422.5, 0.085)	(559.3, 0.112)	(572.1, 0.115)	(578.9, 0.116)
U09	(2003.4, 0.124)	(2447.6, 0.152)	(1870.1, 0.116)	**(1476.8, 0.092)**	(1305.9, 0.081)	(1403.7, 0.087)	(1866.5, 0.116)	(1749.3, 0.109)	(1988.5, 0.123)
U10	(2526.9, 0.123)	(3505.4, 0.170)	**(1811.7, 0.088)**	(2113.5, 0.103)	(1755.6, 0.085)	(2007.5, 0.098)	(2016.8, 0.098)	(2044.1, 0.099)	(2789.6, 0.136)
Time	347.16	645.42	304.71	237.59	221.72	277.50	242.31	292.13	198.75

Table A3
MBBO and CEC competition algorithm results for 10 constrained multi-objective benchmark functions. The table shows the relative hypervolume and normalized relative hypervolume. The best results with respect to MBBO in each row are shown in **boldface** font. The best results among all algorithms are underlined. Average CPU times are shown in the last row of the table.

Fun.	MBBO				CEC competition algorithms		
	VEBBO	NSBBO	NPBBO	EMBBO	MTS	DMOEADD	LiuLiAlgorithm
C01	(6.240, 0.185)	(6.545, 0.193)	(4.254, 0.126)	**(3.853, 0.114)**	(5.256, 0.156)	(3.471, 0.103)	(4.178, 0.124)
C02	(15.53, 0.171)	(20.41, 0.225)	**(8.040, 0.088)**	(9.844, 0.108)	(17.20, 0.190)	(7.233, 0.081)	(7.639, 0.085)
C03	(723.6, 0.181)	(730.5, 0.183)	(529.6, 0.133)	**(458.1, 0.115)**	(607.8, 0.152)	(508.6, 0.122)	(519.5, 0.130)
C04	(9.890, 0.141)	(15.30, 0.219)	(7.199, 0.103)	**(5.912, 0.084)**	(13.17, 0.188)	(5.989, 0.085)	(6.361, 0.099)
C05	(39.29, 0.123)	(73.61, 0.232)	**(30.89, 0.097)**	(34.56, 0.109)	(62.90, 0.198)	(30.80, 0.096)	(40.13, 0.126)
C06	(0.280, 0.155)	(0.561, 0.311)	(0.238, 0.131)	**(0.191, 0.105)**	(0.185, 0.102)	(0.162, 0.089)	(0.187, 0.103)
C07	(54.11, 0.105)	(152.1, 0.296)	(50.28, 0.097)	**(36.62, 0.071)**	(110.4, 0.215)	(31.30, 0.061)	(48.65, 0.090)
C08	**(111.7, 0.109)**	(186.1, 0.181)	(121.6, 0.118)	(159.3, 0.155)	(138.7, 0.135)	(109.5, 0.105)	(177.3, 0.173)
C09	(137.0, 0.152)	(176.9, 0.197)	(155.2, 0.173)	**(105.0, 0.117)**	(112.4, 0.125)	(109.41, 0.120)	(120.8, 0.134)
C10	(1439.5, 0.113)	(2631.5, 0.207)	**(1334.8, 0.105)**	(1508.1, 0.118)	(2170.2, 0.171)	(1157.6, 0.091)	(2438.4, 0.192)
Time	322.30	513.75	284.24	214.42	256.87	227.63	270.35

γ: rank of a solution with respect to a given objective
λ: immigration rate
μ: emigration rate
θ: mutation rate

See Tables A1–A3.

References

Berg, J.P., 1999. A literature survey on planning and control of warehousing systems. IIE Trans. 31, 751–762.

Chan, F.T.S., Kumar, V., 2009. Hybrid TSSA algorithm-based approach to solve warehouse-scheduling problems. Int. J. Prod. Res. 47 (4), 919–940.

Chen, C., Chen, Y., Zhang, Q., 2009. Enhancing MOEA/D with guided mutation and priority update for multi-objective optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. Trondheim, pp. 209–216.

Choi, T.M., Yeung, W.K., Cheng, T.C.E., 2013. Scheduling and co-ordination of multi-suppliers single-warehouse-operator single-manufacturer supply chains with variable production rates and storage costs. Int. J. Prod. Res. 51 (9), 2593–2601.

Chutima, P., Wong, N., 2014. A Pareto biogeography-based optimisation for multi-objective two-sided assembly line sequencing problems with a learning effect. Comput. Ind. Eng. 69 (1), 89–104.

Coello, C.A.C., Pulido, G.T., Lechuga, M.S., 2004. Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. 8 (3), 256–279.

Costa e Silva, E.M.A., Coelho, L.D.S., Lebensztajn, L., 2012. Multiobjective biogeography-based optimization based on predator-prey approach. IEEE Trans. Magn. 48 (2), 951–954.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197.

Derrac, J., Garcia, S., Molina, D., Herrera, F., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1 (1), 3–18.

Gagliardi, J.P., Renaud, J., Ruiz, A., 2012. Models for automated storage and retrieval systems: a literature review. Int. J. Prod. Res. 50 (24), 7110–7125.

Horn, J., Nafpliotis, N., Goldberg, D. E., 1994. A niched Pareto genetic algorithm for multi-objective optimization. In: Proceedings of the IEEE World Congress on Computational Intelligence, pp. 82–87.

Jamuna, K., Swarup, K.S., 2012. Multi-objective biogeography based optimization for optimal PMU placement. Appl. Soft Comput. 12 (5), 1503–1510.

Kukkonen, S., Lampinen, J., 2009. Performance Assessment of Generalized Differential Evolution3 (GDE3) with a given set of constrained multi-objective optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation. Trondheim, pp. 3593–3600.

Lerher, T., Edl, M., Rosi, B., 2013. Energy efficiency model for the mini-load automated storage and retrieval systems. Int. J. Adv. Manuf. Technol. 70 (1), 97–115.

Lerher, T., Ekren, B.Y., Dukic, G., Rosi, B., 2015a. Travel time model for shuttle-based storage and retrieval systems. Int. J. Adv. Manuf. Technol. 40 (3), 101–121.

Lerher, T., Ekren, B.Y., Sari, Z., Rosi, B., 2015b. Simulation analysis of shuttle based storage and retrieval systems. Int. J. Simul. Modell. 14 (1), 11–23.

Lerher, T., Potrc, I., Sraml, M., Tollazzi, T., 2010a. Travel time models for automated warehouses with aisle transferring storage and retrieval machine. Eur. J. Oper. Res. 205 (3), 571–583.

Lerher, T., Sraml, M., Potrc, I., Tollazzi, T., 2010b. Travel time models for double-deep automated storage and retrieval systems. Int. J. Prod. Res. 48 (11), 3151–3172.

Lerher, T., Sraml, M., Potrc, I., 2011. Simulation analysis of mini-load multi-shuttle automated storage and retrieval systems. Int. J. Adv. Manuf. Technol. 54 (1), 337–348.

Liu, M., Zou, X., Chen, Y., 2009. Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances. In: Proceedings of the IEEE Congress on Evolutionary Computation. Trondheim, pp. 2913–2918.

Liu, H., Li, X., 2009. The multiobjective evolutionary algorithm based on determined weights and sub-regional search. In: Proceedings of the IEEE Congress on Evolutionary Computation. Trondheim, pp. 1928–1934.

Mallipeddi, R., Suganthan, P.N., 2010a. Ensemble strategies with adaptive evolutionary programming. Inf. Sci. 180 (9), 1571–1581.

Mallipeddi, R., Suganthan, P.N., 2010b. Ensemble of constraint handling techniques. IEEE Trans. Evol. Comput. 14 (4), 561–579.

Mallipeddi, R., Suganthan, P.N., Pan, Q.K., Tasgetiren, M.F., 2011. Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl. Soft Comput. 11 (2), 1679–1696.
Ma, H., Simon, D., 2011. Blended biogeography-based optimization for constrained optimization. Eng. Appl. Artif. Intell. 24 (3), 517–525.
Ma, H., Ruan, X., Pan, Z., 2012. Handling multiple objectives with biogeography-based optimization. Int. J. Autom. Comput. 9 (1), 30–36.
Pan, Q.K., Suganthan, P.N., Tasgetiren, M.F., 2009. A harmony search algorithm with ensemble of parameter sets. In: Proceedings of the IEEE Congress on Evolutionary Computation. Norway, pp. 1815–1820.
Roodbergen, K.J., Vis, I.F.A., 2009. A survey of literature on automated storage and retrieval systems. Eur. J. Oper. Res. 194, 343–362.
Srinivas, N., Deb, K., 1995. Multi-objective optimization using non-dominated sorting in genetic algorithms. Evol. Comput. 2 (3), 221–248.
Schaffer, C., 1985. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the International Conference on Genetic Algorithms and Their Application. Pittsburgh, Pennsylvania, pp. 93–100.
Suganthan, P.N., 2014. ⟨http://www3.ntu.edu.sg/home/EPNSugan/⟩.
Simon, D., 2008. Biogeography-based optimization. IEEE Trans. Evol. Comput. 12 (6), 702–713.
Simon, D., 2013. Evolutionary Optimization Algorithms. John Wiley & Sons, Hoboken, NJ.
Tasgetiren, M.F., Suganthan, P.N., Pan, Q.-K., Mallipeddi, R., Sarman, S., 2010a. An ensemble of differential evolution algorithms for constrained function optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. Barcelona, Spain, pp. 967–975.
Tasgetiren, M.F., Suganthan, P.N., Pan, Q.K., 2010b. An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. Appl. Math. Comput. 215 (9), 3356–3368.
Tseng, L., Chen, C., 2009. Multiple trajectory search for unconstrained/constrained multi-objective optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. Trondheim, pp. 1951–1958.
Wang, L., Fang, C., 2011. An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem. Inf. Sci. 181 (20), 4804–4822.
Yang, W., Deng, L., Niu, Q., Fei, M., 2013. Improved shuffled frog leaping algorithm for solving multi-aisle automated warehouse scheduling optimization. Commun. Comput. Inf. Sci. 402, 82–92.
Yeung, W.K., Choi, T.M., Cheng, T.C.E., 2010. Optimal scheduling in a single-supplier single-manufacturer supply chain with common due windows. IEEE Trans. Autom. Control 55, 2767–2777.
Yeung, W.K., Choi, T.M., Cheng, T.C.E., 2011. Supply chain scheduling and coordination with dual delivery modes and inventory storage cost. Int. J. Prod. Econ. 132, 223–229.
Yu, E.L., Suganthan, P.N., 2009. Evolutionary programming with ensemble of external memories for dynamic optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. Norway, pp. 431–438.
Zhao, S.Z., Suganthan, P.N., 2010. Multi-objective evolutionary algorithm with ensemble of external archives. Int. J. Innov. Comput., Inf. Control 6 (1), 1713–1726.
Zhang, Q., Zhou, A., Zhao, S.Z., Suganthan, P.N., Liu, W., Tiwari, S., 2008. Multi-objective Optimization Test Instances for the CEC 2009 Special Session and Competition. Technical Report.