A Novel Evolutionary Root System Growth Algorithm for Solving Multi-objective Optimization Problems

Lianbo Ma ^a, Xingwei Wang ^{a*}, Min Huang ^b, Hao Zhang ^c, Hanning Chen ^d,

^a College of Software, Northeastern University, Shenyang, 110819, China

^b College of Information Science and Engineering, Northeastern University, Shenyang, 110819, China

^c Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang, 110016, China

^d School of Computer Science and Software, Tianjin Polytechnic University, Tianjin, 300387, China

Abstract: This paper proposes a novel multi-objective root system growth optimizer (MORSGO) for the copper strip burdening optimization. The MORSGO aims to handle multi-objective problems with satisfactory convergence and diversity via implementing adaptive root growth operators with a pool of multi-objective search rules and strategies. Specifically, the single-objective root growth operators including branching, regrowing and auxin-based tropisms are deliberately designed. They have merits of appropriately balancing exploring & exploiting and self-adaptively varying population size to reduce redundant computation. The effective multi-objective strategies including the fast non-dominated sorting and the farthest-candidate selection are developed for saving and retrieving the Pareto optimal solutions with remarkable approximation as well as uniform spread of Pareto-optimal solutions. With comprehensive evaluation against a suit of benchmark functions, the MORSGO is verified experimentally to be superior or at least comparable to its competitors in terms of the IGD and HV metrics. The MORSGO is then validated to solve the real-world copper strip burdening optimization with different elements. Computation results verifies the potential and effectiveness of the MORSGO to resolve complex industrial process optimization.

Key Words: Multi-objective optimization, Root growth, Burdening calculation

1. Introduction

The recent developmental trends of copper strip processing focus on the large scale, lower energy consume, cost decreasing and integrated product chains in the non-ferrous metals industry [1, 2]. This gives rise to some challenging issues regarding resources optimization operations of the high-precision copper strip production [3, 4, 5]. Especially, the industrial burdening process is substantially important to improve the products quality and quantity in the automatic production line of copper strip. Thus, the optimization scheme and related system for the burdening process are the key to ensure the rational charging ratio for various raw materials to maintain the burdening quality [5, 6]. More specifically, it is responsible for determining the optimal formula of raw materials

(e.g., charging weight and charging time) according to feed manner, feed rate, and feed molar ratio of raw materials in the copper smelting process.

It is worthy noted that the burdening optimization problem usually involves multiple interrelated variables, multi objectives and strict constraints [6]. Particularly, in addition to explicit consideration of rational proportion of elemental composition, there are many other significant factors that should be considered, such as total cost, feed rate, inventory management, and burning loss of raw materials for copper smelting [6, 7]. Due to the fact that the traditional linear mathematical model is incapable of handling these complex factors and constrains appropriately, a nonlinear and multi-objective burdening calculation (or optimization) model should be established in order to simulate the actual burdening process more accurately.

Obviously, optimizing the copper strips burdening calculation (CSBC) is essentially a typical multi-objective problem (MOP) with strict constrains [7]. Since Schaffer's pioneer work on multi-objective evolutionary algorithms, a series of multi-objective optimization algorithms have been proposed and developed [8-15], prominent examples being non-dominated sorting genetic algorithm II (NSGAI) [8], multi-objective evolutionary algorithm based on decomposition (MOEA/D) [9], strength Pareto evolutionary algorithm (SPEA2) [10], and multi-objective particle swarm optimization (MOPSO) [11]. More recently, Gong et al. [12] introduce effective selection and learning strategies into multi-objective artificial immune algorithms to improve algorithmic convergence and diversity maintenance. J. Palacios et al. [13] propose a novel dominance-based tabu search method for the multi-objective evolutionary algorithm. Zhan et al. [14] utilize the multiple populations coevolutionary technique to let each population straightforwardly correspond with one specific objective. And, some techniques for evolutionary many-objective algorithms (i.e., more than four objectives) are also developed [15]. However, how to improve the diversity of population or overcome the local convergence of algorithms is still a challenging issue in multi-objective (MO) optimization.

Different from these animal (or microbe)-oriented computation paradigms, the development of plant-based evolutionary computation attract little attention due to the plant's specific lifestyle [16, 17, 18]. Plant roots have evolved the promising ability to sense local environmental factors and use this information to drive continuous changes in growth direction and root system development [17, 18]. Especially, these behavioral adaptations include the increased lateral branching, root biomass, root length and auxin-based tropisms. All these growth behaviors sharp the final spatial architecture of the root system via auxin transport and signaling [19]. Many studies have revealed that plants can adjust root demography and the length per unit mass of roots in response to heterogeneity [20, 21]. Logically, such iterative growth progress has exhibited great potential for the development and design of new optimization models and algorithms.

One motivation of this paper is to develop a novel multi-objective bionic algorithm derived from the plant root growth, namely multi-objective root system growth optimizer (MORSGO). The proposed MORSGO can appropriately balance the local search and the global search as well as dynamically vary population size. That is, the local search based on the branching operator can generate a number of new individuals by smaller elongation unite and the global search based on the regrowing operator uses relatively bigger elongation unite. This self-adaptive population variation indicates that each individual can dynamically switch its state from branching, to death throughout the growing process. As a result, the population size varies dynamically according to the local fitness landscape [22].

Another feature of the MORSGO lies in its effective multi-objective strategies. Generally, the fast non-dominated sorting used in MORSGO is an elite retention strategy derived from NSGAI [8] to effectively eliminate the infeasible individuals and retain feasible individuals throughout the search process. Furthermore, in order to retain the desired spread of the non-dominated solutions, a new diversity preservation mechanism called the farthest-candidate selection (FCS) is adopted in MORSGO, essentially inspired from the best-candidate sampling theory [23, 24, 25]. Furthermore, MORSGO utilizes a self-adaptive penalty function to tackle strict constraints in CSBC optimization [26].

Noted that recent works [6, 27] have developed the plant-inspired algorithm and copper strip burdening model. In [6], the ABC algorithm is modified by using summation of normalized objective values, diversified selection (SNOVDS) and non-dominated sorting approach. In [27], a single-objective root growth algorithm called RGA is proposed. Obviously, our proposed algorithm is completely different with RGA because MORSGO is a multi-objective algorithm. Although the work [6] and our work share some commonalities: they both establish an copper strip burdening model, and their solving algorithms both use the fast non-dominated sorting approach, it is apparent that our scheme is essentially different from another one in the following aspects.

(1) The principles and realizations of the two optimization algorithms are essentially different. Compared with HMOABC that simply modifies the traditional ABC algorithm by the non-dominated sorting approach, our proposed MORSGO is easy to realize and deliberately designed by the single-objective local searching operators and the multi-objective Pareto-based techniques. Specifically, a set of newly defined growth operators have merits of keeping appropriate balance of exploring & exploiting and self-adaptively varying its population size during search process. And the incorporated FCS method and the fast non-dominated sorting are more effective to search a set of representative Pareto optimal solutions in a single run. The performance superiority of MORSGO have been validated by comparative results in the following experiments.

(2) Compared with the previous model, our proposed CSBC model is more close to industrial reality. The

previous model doesn't consider the factor of material reuse. In contrary, our objective function incorporates the proximity metric between the production standards of copper materials in different grades. Its parameters are also re-established according to our new imported reproduction line of copper strips. More specifically, our model introduces a new definition Eq. (31) to represent the proximity metric between the production standards of copper materials in different grades for the goal of material reuse. It should be highlighted that how to promote reuse ratio of the old material is a very important yet challenging factor in the actual industrial production. Fortunately, by incorporating these improvements, our proposed model makes the reutilized old material in other grades significantly meet the production standards of current grade in melting furnaces, essentially promoting the ratio of material reuse.

The rest of this paper is organized as follows. In Section 2 the MORSGO algorithm is presented in detail. Section 3 experimentally compares MORSGO with several successful algorithms on a set of multi-objective benchmarks. In Section 4, the multi-objective CSBC model is established, and the performance of the MORSGO on several CSBC instances is verified. The conclusions are finally drawn in Section 5.

2. Multi-objective Root System Growth Optimizer

2.1 Root System Growth Algorithm (RSGA)

In order to model these growth dynamic of plant root, some criteria should be firstly defined as follows.

(1) **Criterion-1: Auxin-regulated mechanism.** Auxin plays an important role in the root development and its transport and signaling essentially control different stages of the root growth [27, 28]. Here the auxin concentration is defined to reflect this effect, dynamically reallocated after each growth generation.

(2) **Criterion-2: Taxonomy of roots.** The root population is grouped into main roots, lateral roots and dead roots according to their auxin concentration values in descending order, which respectively implement different operations. Accordingly, each root will undergo three survive stages, namely branch, decay, and death. As a result, the population size variation evolves throughout the growing process.

(3) **Criterion-3: Root tropisms and self-adaptive growth.** The directional growth of plant roots relative to the direction of environmental stimuli is regarded as the tropism [29]. Among these tropisms, hydrotropism means roots sense environmental nutrient gradient to orient the directional growth towards the current best position obtained so far. Gavotropism means main roots usually grow down (i.e., orthogravitropism), and lateral roots always grow sideways (i.e., diagravotropism).

(4) **Criterion-4: Root-to-root communication and memory.** Plants show memory marked by their historical

experiences or experiences from their parents to adjust growth behaviors [30]. And plant roots also communicate with each other via exchanging information about neighbor's state and resource distribution [31, 32], which indicates that plant roots can utilize this hybrid information to determine their root foraging behaviors.

Based on above criteria, the original root system growth algorithm (RSGA) is designed as below.

2.1.1 Auxin concentration

Assume to seek the minimum of $F(x)$, $x \in R^D$, $F(x)$ symbolizes the nutrient distribution in the soil environment, and it can be mathematically defined as a single-objective or multi-objective objective function. In RSGA, the plant root system consists of a collection of root tips as

$$RS = \{\theta_i^t \mid i = 1, 2, \dots, P^t; t = 1, 2, \dots, T\} \quad (1)$$

where

$$\theta_i^t = \langle x_i^t, f_i^t, n_i^t, \alpha_i^t, \phi_i^t \rangle \quad (2)$$

represents a single root tip; P^t is the population size at time t ; t denotes the growth interval; θ_i^t consists of its own position x_i^t , fitness f_i^t , nutrient n_i^t , auxin α_i^t and angle ϕ_i^t . And ϕ_i^t is a D-1-dimensional growth angle. Note that the auxin α_i^t is used to as the quality metric of the root.

In the initial phase (i.e., $t=0$), the initial population P^0 is randomly initialized in a D-dimensional decision variable space. The position and heading angle of the i th root tip can be expressed as $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ and $\phi_i = (\phi_{i1}, \phi_{i2}, \dots, \phi_{i(D-1)})$, respectively. Here $x_{id} \in [l_d, u_d]$, $d \in [1, D]$, l_d , u_d are the lower and upper bounds of d th dimension, respectively. During each foraging time step t , each individual i searches for nutrient and its corresponding nutrient n_i^t is updated by:

$$n_i^{t+1} = \begin{cases} n_i^t + 1 & \text{if } f_i^{t+1} < f_i^t \\ n_i^t - 1 & \text{else} \end{cases} \quad (3)$$

Note that the main motivation of updating the nutrient n_i is to access the variation trend or gradient variation information of the function fitness landscape, which is computed as a constituent part of Auxin to determine whether one root is strong. Generally, at the initial state, the nutrient n_i of each root i can be initialized as 0. During the foraging process, n_i^t can be added to a positive number or decreased to a negative number.

Then by combining the health and energy states of i th individual, the auxin concentration α_i^t can be manipulated as

$$health_i^t = \frac{f_i^t - f_{worst}^t}{f_{best}^t - f_{worst}^t} \quad (4)$$

$$energy_i^t = \frac{n_i^t - n_{worst}^t}{n_{best}^t - n_{worst}^t} \quad (5)$$

$$\alpha_i^t = \xi \frac{health_i^t}{\sum_{j=1}^{P'} health_j^t} + (1 - \xi) \frac{energy_i^t}{\sum_{j=1}^{P'} energy_j^t}, \quad \xi \in [0, 1] \quad (6)$$

where f_{worst}^t / f_{best}^t denotes current worst/best fitness and n_{worst}^t / n_{best}^t is the worst/best nutrient of the whole population at time t , ξ is a uniform random quantity varying from 0 to 1.

In each growing cycle, all individuals are sorted according to their auxin concentration values in descending order. That is, the strong individuals (with higher auxin concentrations) can be selected as main roots to branch. In our model, half of current sorted roots are selected as main roots:

$$S_m^t = P^t / 2 \quad (7)$$

where S_m^t is the number of selected main roots, P^t is current population size.

2.1.2 Main roots growth: tropisms-based regrowing

The growth operator of main roots is designed as the tropisms-based regrowing, as shown below.

Step.1 In each cycle, the group of main roots is constructed by Eq. (7).

Step.2 Considering the effect of hydrotropism, select half of main roots to search towards the optimal position of individuals, given by:

$$x_i^{t+1} = x_i^t + R_3 (x_{best}^t - x_i^t) \quad (8)$$

where $i \in [1, S_m^t / 2]$, R_3 is random value in the range (0, 1), and x_{best}^t is the best position in the root tip group.

Step.3 Considering the effect of gravitropism depending on the root memory, the rest of main roots will grow along their original directions as:

$$x_i^{t+1} = \begin{cases} x_i^t + R_4 I_{\max}(\phi_i^t) & \text{if } x_i^t > x_i^{t-1} \\ x_i^{t+1} = x_i^t + R_3 (x_{best}^t - x_i^t) & \text{if } x_i^t \leq x_i^{t-1} \end{cases} \quad (9)$$

where $i \in [S_m^t / 2, S_m^t]$, $I_{\max}$ is the maximum root elongation length, R_4 is a normally distributed random number with mean 0 and standard deviation 1; $H(\phi_i^t)$ is a D-dimensional growth direction of the main root i ; $\phi_i^t = (\phi_{i1}^t, \phi_{i2}^t, \dots, \phi_{i(D-1)}^t) \in R^{D-1}$ is a D-1-dimensional growth angle, given by:

$$\{\text{forward, left, right} | \Delta\phi_i^t, r\} = \begin{cases} \text{forward: } \Delta\phi_i^t=0 & \text{if } 0 \leq r < 1/3 \\ \text{left: } \Delta\phi_i^t=\lambda \phi_{\max} & \text{if } 1/3 \leq r < 2/3 \\ \text{right: } \Delta\phi_i^t=-\lambda \phi_{\max} & \text{if } 2/3 \leq r \leq 1 \end{cases} \quad (12)$$

where $\Delta\phi^t$ is the angle increment of one branching root ϕ_i^t , $\phi_{\max}$ is the maximum growing turning angle, which is limited to $\pi/2$, λ is a random inertial coefficient varying within $[0,1]$, and r is a random value between 0 and 1, which is used to select the growth direction.

Step.2-2 After the growth direction is determined, the new branching point x_i^{t+1} can be generated by following equations:

$$\phi_i^{t+1} = \phi_i^t + \Delta\phi_i^t \quad (13)$$

$$x_i^{t+1} = x_i^t + R_2 l_{\max} H(\phi_i^{t+1}) \quad (14)$$

where R_2 is a random value between 0 and 1, $l_{\max}$ is the maximum root elongation length, and $H(\phi_j^{t+1}) = (h_{j1}^{t+1}, h_{j2}^{t+1}, \dots, h_{jD}^{t+1}) \in R^D$ is a Polar to Cartesian coordinates transform function, which can be calculated as:

$$\begin{aligned} h_{j1}^{t+1} &= \prod_{p=1}^{D-1} \cos(\phi_{jp}^{t+1}) \\ h_{jk}^{t+1} &= \sin(\phi_{j(k-1)}^{t+1}) \prod_{p=j}^{D-1} \cos(\phi_{jp}^{t+1}) \\ h_{jD}^{t+1} &= \sin(\phi_{j(D-1)}^{t+1}) \end{aligned} \quad (15)$$

2.1.4 Random walk of lateral roots

During each foraging bout, all the lateral root tips will perform the random walk, which are thought to be the most efficient foraging strategy for randomly distributed nutrition [33]. At the t^{th} iteration, each lateral root tip generates a random head angle and a random elongation length, given by:

$$\phi_i^{t+1} = \phi_i^t + R_5 \phi_{\max} \quad (16)$$

$$x_i^{t+1} = x_i^t + R_6 l_{\max} H(\phi_{\max}^{t+1}) \quad (17)$$

where $i \in [0, S_t]$, R_5 and R_6 are random values in the range $(0, 1)$, $\phi_{\max}$ is the maximum growing turning angle, and $l_{\max}$ is the maximum root elongation length.

2.1.5 The elimination operation and self-adaptive population variation

In RSGA, all roots whose n_i^t values are less than one certain threshold will be eliminated from the current population, and the population size is calculated as

$$P^t = \begin{cases} P^t + w_i & \text{if } n_i^t > \text{BranchG} \\ P^t - 1 & \text{if } n_i^t < \text{Nmority} \end{cases} \quad (18)$$

where w_i is the branching number defined by Eq. (11), BranchG is the branching threshold defined by Eq. (11), Nmority is the death threshold.

In order to avoid the non-equilibrium condition that the population size is too large or too small, the branching criterion and the death criterion are delicately designed as

$$\text{BranchG} = \max(F_{\text{branch}}, F_{\text{branch}} + \frac{(P^t - P)}{F_{\text{adapt}}}) \quad (19)$$

$$\text{Nmority} = \min(0, \frac{(P^t - P)}{F_{\text{adapt}}}) \quad (20)$$

where P is the initial population size and P^t is current population size, F_{branch} and F_{adapt} are used to adjust the branching process. In our experiments, the F_{branch} and F_{adapt} are empirically set to 5/10.

Based on these mathematic models and operations from above sections, the pseudo code of RSGA algorithm is listed in Table 1.

Table 1. Pseudo-code of RSGA

RSGA pseudo code	
Step 1: Initialization:	Randomize positions of the population; Set iteration=0.
Step 2: Roots Classification:	<p>Compute the nutrient and auxin concentrations of root tips by Eq. (3) and Eq.(6);</p> <p>Classify the root tips into two groups, i.e., main roots and lateral roots group by Eq.(7).</p>
Step 3: Main roots operations:	<p>Calculate the branching number of all individuals by Eq. (11);</p> <p>Branching new roots for each parent individual by Eq. (13), Eq. (14) and Eq. (15),</p> <p>Update the population size by Eq. (18);</p> <p>Calculated the regrowing angle by Eq. (10);</p> <p>Select part of mainroots to regrow by Eq. (8) and the rest ones regrow by Eq. (9);</p> <p>Evaluate the fitness of the renewal main roots.</p>
Step 4: Lateral-roots operations:	<p>The renewal lateral roots implement random walk by Eq.(16) and Eq.(17).</p> <p>Evaluate the fitness of the renewal lateral-root;</p> <p>Update corresponding nutrient concentrations by Eq. (3).</p>
Step 5: Dead roots operations:	<p>Remove low-auxin-concentration individuals from current population by Eq.(18);</p> <p>Update population size.</p>
Step 6:	Set iteration= Iteration +1; If termination criteria is met, record the best solution obtained; Otherwise, go to Step 2.

2.2 The MORSGO Algorithm

2.2.1 Fast non-dominated sorting

The main motivation of fast non-dominated sorting approach is to rank all the solutions into multi-level non-dominated fronts. In this approach, the root population is sorted by computing the dominance relationship between individuals or solutions. Each solution x should be compared with other solutions from current population to determine whether it is a non-dominated solution. Specifically, in each computation iteration, two significant operations should be calculated, namely domination count D_x , and the set of solutions S_x dominated by solution x . D_x denotes the number of solutions dominated by x from current population P . Main procedures of this algorithm are given in Algorithm 1.

Algorithm 1. Fast non-dominated sorting

Step 1: For each $x \in P$, initialize

$D_x = 0;$ //Domination counter for solution x
 $S_x = \phi;$ //Set of solutions dominated by x

Step 2: Let $q \in P$. for each q ,

if $x \prec q$, let $S_x = S_x \cup \{q\};$ // q is added into the solution set of S_x
 else if $q \prec x$, let $D_x = D_x + 1;$ //Domination counter for x is accordingly added

Step 3: If $D_x = 0,$ //x falls within the first front set

let $x_{rank} = 1$ and $F_1 = F_1 \cup \{x\};$ //x is added into the Pareto front set

Step 4: $i = 1;$

Step 5: Let $Q = \phi$, for each $x \in F_i$, //Q is defined to memorize the solutions of the next front

for each $q \in S_x$,
 Let $D_q = D_q + 1;$
 If $D_q = 0$, let $x_{rank} = i + 1$, $Q = Q \cup \{x\};$

Step 6: Let $i = i + 1$ and $F_i = Q$,

Step 7: If $F_i \neq \phi$, return 5; Else, stop.

2.2.2 Objective fitness normalization

The Chebyshev approach is used to normalize each objective function value of all M objective functions [34,35]. Then, for each objective function $f_i(x)$, its normalization is defined as

$$f_i^l(x) = \frac{f_i(x) - z_i^1}{z_i^2 - z_i^1} \quad (21)$$

where $z_i^1 = \text{sub}(f_i(x))$ and $z_i^2 = \text{inf}(f_i(x))$, which are often known as *a priori*. In the optimization process, z_i^{l*} and z_i^{2*} are used to represent the dynamical maximum and minimum fitness values of $f_i(x)$, respectively. By using this

operation, all the normalized objective functions can be summed into a single value.

2.2.3 Modified operators of auxin concentration and tropisms

- *Handling the auxin concentration*

On the basis of the objective fitness normalization, the auxin concentration and regrowing operator can be handled for the multi-objective optimization. For the j th objective function of the i th solution at time t , namely $\alpha_i^j(t)$, the membership function $u_i^j(t)$ is redefined to normalize $\alpha_i^j(t)$, given as

$$u_i^j(t) = \begin{cases} 1 & \text{if } \alpha_i^j(t) > \alpha_{\max}^j \\ \frac{\alpha_i^j(t) - \alpha_{\min}^j}{\alpha_{\max}^j - \alpha_{\min}^j} & \text{if } \alpha_{\min}^j \leq \alpha_i^j(t) \leq \alpha_{\max}^j \\ 0 & \text{if } \alpha_i^j(t) < \alpha_{\min}^j \end{cases} \quad (22)$$

where $j \subseteq [1, \dots, M]$, M is the number of objective functions; $\alpha_{\min}^j$ and $\alpha_{\max}^j$ are the minimum and maximum auxin concentration values of the j th objective function, respectively.

For the i th solution, its normalized auxin concentration function u^k is defined to replace its original operator (i.e., Eq. (6)) as

$$u^k = \frac{\sum_{j=1}^M u_i^j(t)}{\sum_{i=1}^N \sum_{j=1}^M u_i^j(t)} \quad (23)$$

where N is current population size, M is the number of objective functions.

- *Handling pbest in hydrotropism-based regrowing*

In the regrowing operator, the x_{best} (or $gbest$) is the best solution found by current root population. However, in the multi-objective optimization, how to select a single $gbest$ out of a set of non-dominated solutions which are all equally good is difficult. An alternative scheme is adopting crowding-distance method to determine the priority of a non-dominated solution [8] as below.

The crowding-distance of a dominant root x in the population F_1 obtained by the fast non-dominated sorting approach can be calculated by

$$\zeta(x, F_1) = \Delta \sum_{i=1}^M \frac{\zeta_i(x, F_1)}{f_i^{\max} - f_i^{\min}} \quad (24)$$

where $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values of the i th objective function, respectively; M is the number of the objective functions and

$$\zeta_i(x, F_1) = \begin{cases} \infty, & \text{if } f_i(x) = \min\{f_i(x') \mid x' \in F_1\} \text{ or } f_i(x) = \max\{f_i(x') \mid x' \in F_1\} \\ \min\{f_i(x') - f_i(x'') \mid x', x'' \in F_1 : f_i(x') < f_i(x) < f_i(x''), \text{ otherwise}\} \end{cases} \quad (25)$$

According to $\zeta(x, F_1)$, the density of dominant roots surrounding x in F_1 can be estimated effectively. By

taking the inverse density value $\frac{1}{\zeta_i(x, F_1)}$ as fitness, the roulette wheel selection is implemented to select an individual as the *gbest*.

2.2.4 The farthest-candidate selection (FCS) method

In MO algorithms, the crowded distance method [8] is widely adopted to measure the density of ambient solutions for maintaining wide spread of non-dominated solutions. However, this mechanism is sometimes inefficient in some situations, e.g., in which the most original points distribute very closely to each other while the others do not [23]. In the MORSGO, an improved selection method, namely the farthest-candidate selection (FCS) method is adopted based on the works of multi-objective optimization theory to overcome this issue [23, 24]. Detailed procedures of this method are given in Algorithm 2.

Algorithm 2. The FCS method

Step 1: Initialization.

To selecting K solutions from a given population P consisting of S individuals, set

$S_{accept} = \phi$; // S_{accept} memorizes the selected solutions
 $D[x_i] = 0, i = 1, 2, \dots, S$; // $D[x_i]$ memorizes the minimum Euclidean distance

Step 2: Select boundary solutions.

For each objective function $f_j(x), j = 1, 2, \dots, m$ // m is the number of objective functions

$S_{accept} = S_{accept} \cup \arg \min_{x \in P} (f_j(x)) \cup \arg \max_{x \in P} (f_j(x))$;

Step 3: Euclidean distance calculation.

Let $Sm[x] = P - S_{accept}$, for each individual $x \in Sm$,

$D[x_i] \leftarrow \arg \min_{x' \in S_{accept}} (dis(x, x'))$; // dis calculates the Euclidean distance

Step 4: Select the farthest solutions.

For $i = 1$ to $K - |S_{accept}|$

$x1 = \arg \max_{x \in (P - S_{accept})} (D[x])$;

For each $x2 \in P - S_{accept}$

$D[x_2] \leftarrow \min(D[x_2], dis(x, x'))$;

$S_{accept} \leftarrow S_{accept} \cup x1$

End For

End For

The main concept of Algorithm 2 is that, in order to select K best individuals from current population, for each new individual to be selected, it prioritizes selecting the unselected individuals with farthest Euclidean distance from current selected solutions as the accepted candidate solution. Specifically, in Algorithm 2, the boundary

solutions with the smallest and largest fitness values are added into the set of selected solutions (i.e., S_{accept}). Then, the Euclidean distance between each solution and unselected ones are calculated and the minimum values of Euclidean distance are memorized. Finally, the farthest solutions are selected into S_{accept} . Accordingly, the computational complexity of this method is $O(mS^2)$ where S is the population size and m is the number of objective functions.

In MORSGO, after fast non-dominated sorting operation, there is usually a crowded scenario that the number of current non-domination solutions significantly exceeds the required size of population used for next evolutionary generation. Accordingly, Algorithm 2 can effectively select K elitists from S non-dominated solutions especially in the particular case where most original points distribute very closely to each other, which essentially enhances diversity of non-dominated solutions.

Table 2. Main procedures of MORSGA

MORSGA algorithm
<p>Step. 1: Randomly initialize the population P_t with N individuals; Then set $t = 0$;</p> <p>Step. 2: Calculate the fitness, auxin concentration values of each solution of population P_t by using Eq.(4)-Eq.(6), and Eq.(23). And divide current population into main roots group MR_t and lateral roots group LR_t, where the strongest individuals with higher auxin concentration values are selected as main roots according to the Eq.(7).</p> <p>Step. 3: Implement root growth operators and generate new solutions as MN_t by using Eq.(8)-Eq.(14), and Eq.(24) and then construct a combined main root population $MC_t = MR_t \cup MN_t$.</p> <p>Step. 4: Calculate the fitness, auxin concentration values of each solution and sort the combined population MC_t by implementing Algorithm 1. Then each solution has a specific domination level, which is identified by a rank value. The size of MC_t is $size(MR_t) + size(MN_t)$ where $size(MN_t)$ usually seems bigger due to its branching operation.</p> <p>Step. 5: Select exactly $size(MR_t)$ best roots by using Algorithm 2 new MR_t from current main root population MC_t.</p> <p>Step. 6: Implement lateral roots growth operators and generate new solutions as LN_t by using Eq.(16) and Eq.(17) and then construct a combined lateral root population $LC_t = LR_t \cup LN_t$.</p> <p>Step. 7: Calculate the fitness, auxin concentration values of each solution and sort the combined population LC_t by implementing Algorithm 1. Then each solution has a specific domination level, which is identified by a rank value..</p> <p>Step. 8: Select exactly $size(LR_t)$ best roots by using Algorithm 2 as new LR_t from current main root population LC_t.</p> <p>Step. 9: Eliminate the deteriorated roots if they have no enough energy or auxin concentration values by using Eq.(18).</p> <p>Step. 10: Set $t = t + 1$.</p> <p>Step. 11: If the termination conditions are met, stop; otherwise, return to step 2.</p>

2.2.5 The procedures of MORSGO

By using the fast non-dominated sorting approach, the proposed MORSGO preserves the non-dominated roots in the first-level front in an external population, called non-dominant population. Throughout the FCS, only

partial well-distributed non-dominated roots are chosen to implement adaptive growth behaviors and evolution.

The main procedures of MORSGO are listed in Table 2.

Table 3. Test multi-objective benchmarks

Problems	Range	Dimension	Formulation (minimized)
ZDT1	[0,1]	30	$f_1(x) = x_1 \quad f_2(x) = g(x)[1 - \sqrt{x_1 / g(x)}] \quad g(x) = 1 + 9(\sum_{i=2}^n x_i) / (n-1)$
ZDT2	[0,1]	30	$f_1(x) = x_1 \quad f_2(x) = g(x)[1 - (x_1 / g(x))^2] \quad g(x) = 1 + 9(\sum_{i=2}^n x_i) / (n-1)$
ZDT3	[0,1]	30	$f_1(x) = x_1 \quad f_2(x) = g(x)[1 - \sqrt{x_1 / g(x)} - \frac{x_1}{g(x)} \sin(10\pi x_1)]$ $g(x) = 1 + 9(\sum_{i=2}^n x_i) / (n-1)$
ZDT4	$x_1 \in [0,1]$ $x_j \in [-5,5]$ $j = 2, \dots, n$	10	$f_1(x) = 1 - \exp(-4x_1) \sin^6(6\pi x_1) \quad f_2(x) = g(x)[1 - (f_1(x) / g(x))^2]$ $g(x) = 1 + 9[(\sum_{i=2}^n x_i) / (n-1)]^{0.25}$
ZDT6	[0,1]	10	$f_1(x) = 1 - \exp(-4x_1) \sin^6(6\pi x_1) \quad f_2(x) = g(x)[1 - (f_1(x) / g(x))^2]$ $g(x) = 1 + 9[(\sum_{i=2}^n x_i) / (n-1)]^{0.25}$
DTLZ1	[0,1]	$M+k-1,$ $k= x_M =10$	$f_1(x) = 0.5x_1x_2 \dots x_{M-1}(1 + g(x_M))$ $f_2(x) = 0.5x_1x_2 \dots (1 - x_{M-1})(1 + g(x_M))$ $\vdots$ $f_M(x) = 0.5(1 - x_1)(1 + g(x_M))$ $where \quad g(x_M) = 100 \left[x_M + \sum_{x_i \in x_M} ((x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))) \right]$
DTLZ2	[0,1]	$M+k-1,$ $k= x_M =10$	$f_1(x) = (1 + g(x_M)) \cos(x_1\pi/2) \dots \cos(x_{M-1}\pi/2)$ $f_2(x) = (1 + g(x_M)) \cos(x_1\pi/2) \dots \sin(x_{M-1}\pi/2)$ $\vdots$ $f_M(x) = (1 + g(x_M)) \sin(x_1\pi/2)$ $where \quad g(x_M) = \sum_{x_i \in x_M} (x_i - 0.5)^2$
DTLZ6	[0,1]	$M+k-1,$ $k= x_M =20$	$f_1(x) = x_1$ $f_2(x) = x_2$ $\vdots$ $f_{M-1}(x) = x_{M-1}$ $f_M(x) = (1 + g(x_M))h(f_1, f_2, \dots, f_{M-1}, g)$ $g(x_M) = 1 + \frac{g}{ x_M } \sum_{x_i \in x_M} x_i$ $h = M - \sum_{i=1}^{M-1} [\frac{f_i}{1+g} (1 + \sin(3\pi f_i))]$

3. Benchmark Test

3.1 Test Problems

Six representative multi-objective benchmarks are deliberately selected to evaluate the performance of the proposed algorithm. The first five instances are bi-objective ZDT benchmarks, including ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 [36]. The next three DTLZ instances consist of tri-objective DTLZ1, DTLZ2 and DTLZ6 [37]. Detailed formulas of these test instances are listed in Table 3.

3.2 Performance Measures

The inverted generational distance (IGD) indicator [38], the spread criterion Δ [8] and the hypervolume (HV) [39] are suggested as the performance metrics. In general, the IGD computes the average Euclidean distance from well-distributed points along the whole PS to their closest solution in the obtained solution set, and a smaller value is preferable [40-44], the spread Δ is used to calculate the extent of distributed non-uniformity of solutions [8], and the HV calculates the volume of the objective space between the obtained solution set and a specified reference point to examine convergence and diversity of a solution set simultaneously [39, 45-50]. Due to paper length limitation, the detailed procedures of the IGD, Δ and HV can refer to their original literatures [38, 8, 39].

3.3 Experimental Configuration

In this experiment, the four representative MO algorithms including hybrid multi-objective ABC (HMOABC) [6], NSGAII [8], MOEA/D [9], generalized DE 3 (GDE3) [47], are selected for test comparison. NSGAII and MOEA/D are two state-of-the-art algorithms, GDE3 and HMOABC are two most recently well-performing MOEAs. For fair comparison, the parameters configures of aforementioned algorithms are set to the same to their original references [6, 8, 9, 47], as summarized in Table 4. The maximal number of function evaluations (FEs) is set to 500000, and the independent run number is 20. The results found by different algorithms are compared based on the Wilcoxon rank sum test with significant level $\alpha = 0.05$.

In order to validate the FCS method in the MORSGO, we respectively implement the FCS and the crowded distance strategy in the MORSGO, resulting in two MORSGO variants, namely MORSGO and MORSGO-C. Then, a reasonable set of their parameter values are employed empirically as following: $BranchG=50$, $Nmority=0$, $F_{branch}=5$, $F_{adapt}=10$ and the maximum population size is 200. Note that in our experiments the empirical optimal parameter values are focused on, yet the best parameter configuration will be investigated further in future work.

Table 4. Parameters setting

Algorithms	Parameters
NSGAI	$N=100, p_x=0.9, p_m=1/D, \eta_c = \eta_m = 20$
MOEA/D	$N=100, CR=1.0, F=0.5, \eta=20, p_m=1/D, T=20, \delta=0.9, n_r=2$
GDE3	$N=100, CR=0.0, F=0.5$
HMOABC	$N=100, P=90, Limit=1000*D/2, D$ is the dimension of problem
MORSGO/MORSGO_C	$Maxpop=200, BranchG=50, Nmority=0, F_{branch}=5, F_{adapt}=10$

Table 5. Performance comparison on 30-D ZDT1, ZDT2, ZDT3 and 10-D ZDT4, ZDT6 by each algorithm

Func.			MORSGO	NSGAI	MOEA/D	GDE3	HMOABC
ZDT1	IGD	Mean	1.112e-3	1.900e-2	3.932e-3	1.270e-2	4.680e-3
		Sd	1.210e-4	2.282e-2	4.109e-4	1.560e-3	1.760e-4
		Rank	1	5-	2-	4-	3-
	HV	Mean	9.156e-1	9.173e-1	8.901e-1	8.745e-1	9.096e-1
		Sd	1.311e-2	1.228e-2	1.472e-2	1.489e-2	1.745e-2
		Rank	2	1+	4-	5-	3-
ZDT2	IGD	Mean	8.047e-4	3.483e-2	7.653e-3	2.970e-2	2.640e-2
		Sd	9.618e-5	2.527e-2	2.159e-2	1.820e-3	3.000e-2
		Rank	1	5-	2-	4-	3-
	HV	Mean	7.984e-2	7.276e-1	7.310e-1	7.332e-1	6.872e-1
		Sd	7.042e-2	1.223e-2	9.622e-2	1.283e-2	1.180e-1
		Rank	1	4-	3-	2-	5-
ZDT3	IGD	Mean	7.758e-4	3.023e-2	1.631e-3	1.160e-2	5.510e-3
		Sd	4.385e-5	3.513e-2	5.086e-4	2.240e-3	2.490e-4
		Rank	1	5-	2-	4-	3-
	HV	Mean	7.749e-2	7.416e-1	7.752e-1	7.415e-1	7.761e-1
		Sd	1.573e-2	1.269e-2	1.825e-2	1.203e-2	1.446e-2
		Rank	3	4-	2+	5-	1+
ZDT4	IGD	Mean	7.166e-1	7.203e-1	7.331e-1	3.400e-1	3.550
		Sd	1.874e-1	3.853e-1	9.176e-1	3.700e-1	1.350
		Rank	2	3-	4-	1+	5-
	HV	Mean	8.440e-1	5.625e-1	8.310e-2	7.349e-1	6.103e-1
		Sd	3.210e-2	4.972e-2	1.475e-1	3.501e-2	1.642e-2
		Rank	1	5-	2-	3-	4-
ZDT6	IGD	Mean	4.123e-3	2.300e-2	5.204e-3	7.36e-2	4.550e-3
		Sd	3.237e-4	1.756e-2	4.466e-4	9.16e-2	6.953e-4
		Rank	1	4-	3-	5-	2+
	HV	Mean	8.180e-01	7.110e-01	7.524e-1	7.541e-1	7.570e-1
		Sd	1.069e-02	6.286e-02	6.373e-2	1.812e-2	1.434e-2
		Rank	1	5-	4-	3-	2-
Final Rank			13	37	28	37	28

'+', '-', and '=' indicate that the results of the algorithm are significantly better than, worse than, and similar to the ones of

MORSGO by Wilcoxon rank sum test $\alpha = 0.05$

Table. 6. Performance comparison on 7-D DTLZ1, 12-D DTLZ2 and 22-D DTLZ6 by each algorithm

Func.			MORSGO	NSGAI	MOEA/D	GDE3	HMOABC
DTLZ1	IGD	Mean	3.130	1.250e+1	2.364 e+1	1.27e-2	4.800
		Sd	3.854	3.350	3.841	1.56e-3	4.760
		Rank	2	4-	5-	1+	3-
	HV	Mean	4.842e-1	4.101e-1	4.193e-1	4.821e-1	4.295e-1
		Sd	1.003e-1	3.034e-2	3.845e-2	1.135e-1	6.945e-2
		Rank	1	4	5	2	3
DTLZ2	IGD	Mean	1.564e-2	5.095e-2	1.200 e-1	2.97e-2	1.200e-1
		Sd	1.641 e-3	6.738 e-3	7.965 e-3	1.82e-3	3.00e-1
		Rank	1	3-	4-	2-	4-
	HV	Mean	7.423E-1	7.302E-1	7.003E-1	7.167E-1	7.100E-1
		Sd	1.754E-2	3.045E-2	2.782E-2	8.592E-2	1.012E-1
		Rank	1	2	5	3	4
DTLZ6	IGD	Mean	1.885e-2	1.014e-1	4.957e-2	1.160e-2	3.490e-2
		Sd	2.704e-3	6.103e-2	1.052e-2	2.240e-3	2.490e-2
		Rank	2	5-	4-	1+	3-
	HV	Mean	8.845e-1	6.818e-1	7.845e-1	8.401e-1	8.013e-1
		Sd	2.431e-2	6.535e-2	4.983e-2	2.431e-2	2.043e-2
		Rank	1	5-	4-	3-	2-
Final Rank		11	23	26	14	16	

'+', '-', and '=' indicate that the results of the algorithm are significantly better than, worse than, and similar to the ones of MORSGO by Wilcoxon rank sum test $\alpha = 0.05$

3.4 Results and Analysis

Table 5 and Table 6 show results obtained by MORSGO, NSGAI, MOEA/D, GDE3 and HMOABC in 20 runs ZDTs and DTLZs, respectively, where the mean and standard deviation of the IGD-metric and HV-metric values are compared. Here the reference point for HV is defined as a vector of worst objective function values. It is noticed that the IGD results of GDE3 are directly taken from the literature [14].

3.4.1 Results on ZDT instances

From Table 5, it is observed that MORSGO yields better IGD values than its compared algorithms on most ZDTs instances such as ZDT1, ZDT2, ZDT3 and ZDT 6. On ZDT4, GDE3 performs best and MORSGO also performs the second best. On ZDT6, MORSGO also obtains the first rank among all algorithms in terms of the mean and standard deviation. In general, it can be said that MORSGO obtains satisfactory IGD results on these bi-objective benchmarks. ZDT1 is a convex problem. For this easy benchmark, all involved algorithms obtain satisfactory IGD values. ZDT2 and ZDT3 have non-convex and discontinuous Pareto fronts, respectively. Fig.2 shows plots of the final non-dominated fronts with lowest IDG-metric values obtained by each algorithm on the these instances where the true Pareto fronts of ZDTs are also offered. As shown in Fig.2, the final non-dominated solutions on ZDT2 and ZDT3 found by MORSGO are with better approximation to the true Pareto front than those found by other algorithms. ZDT4 contains a number of local and discontinuous Pareto fronts and leads

NSGAI to be trapped into local optima set. MORSGA also suffers from this problem. ZDT6 is a non-uniform problem with low density of solutions towards the ideal Pareto front, MORSGO converges very closely to MOEA/D and significantly better than HMOABC.

The HV results from Table 5 further confirms the performance advantage of MORSGO. From this table, we can see that MORSGO performs superior to other compared algorithms on ZDT2, ZDT4 and ZDT6. These results also indicate that MORSGO can keep pace with the Pareto front whereas compared algorithms are insufficient. And it also gets the second best rank on ZDT1, only slightly worse than NSGAI. For illustration, Fig. 3 shows the HV evolution process obtained by each algorithm on ZDTs. And we can see that MORSGO exhibits stable and superior performance on most ZDTs, which can be explained by the fact that the FCS method in MORSGO plays an important role to maintain diversity, which has been intuitively illustrated as shown in Figs.2 and 3. Fig.3 also shows that MORSGO and other compared algorithms exhibit fast convergence characteristics in the search process on several bio-objective benchmarks (e.g., ZDT4). However, MORSGO still obtains satisfactory HV results in terms of the mean and standard deviation. This fast convergence may be generated due to that the involved algorithms are so suitable or effective to handle these bio-objective benchmarks especially with concave PFs, and they can find satisfactory results in a few of iterations.

3.4.2 Results on DTLZ instances

The results on the DTLZ instances are given and compared in Table 6. The IGD results show that MORSGO performs competitively on DTLZ1 and DTLZ6, and especially performs the best on DTLZ2. More importantly, the HV results, which are more effective to measure convergence and diversity of solutions simultaneously, further confirm the superiority of MORSGO over other algorithms because MORSGO obtains the first rank on DTLZ1, DTLZ2 and DTLZ3 in term of the HV mean values. DTLZ1 is a linear tri-objective problem. For DTLZ1, MORSGO obtains the best mean values, followed by GDE3 and then other algorithms. For the concave DTLZ2, MORSGO performs a little better than NSGAI in approximation and uniformity, significantly than other algorithms. For the more complex DTLZ6 with discontinuous Pareto fronts, MORSGO obtains the first rank of the mean value, and also gets the second best standard deviation value. The statistics by the Wilcoxon rank sum tests also confirm the performance advantage of MORSGO, which can be validated by the plotted distribution of non-dominated solutions in Fig. 4. And Fig. 5 shows the HV-metric evolution processes of all algorithms on DTLZ1, DTLZ2 and DTLZ6. It is clear from Fig. 5 that MORSGO obtains faster convergence and better HV-metric value than other algorithms on most DTLZs (DTLZ1 and DTLZ2) in every generation, which shows evidence that the proposed schema is feasible and efficient.

ZDT1

ZDT1

ZDT1

ZDT1

ZDT1

ZDT2

ZDT2

ZDT2

ZDT2

ZDT2

ZDT3

ZDT3

ZDT3

ZDT3

ZDT3

Fig.2. Non-dominated fronts with lowest IGD values obtained by each algorithm on bi-objective ZDT instances

Fig.3. Evolution process of HV obtained by each algorithm on bi-objective ZDT instances

DTLZ1

DTLZ1

DTLZ1

DTLZ1

DTLZ1

DTLZ2

DTLZ2

DTLZ2

DTLZ2

DTLZ2

DTLZ6

DTLZ6

DTLZ6

DTLZ6

DTLZ6

Fig.4. Non-dominated fronts with lowest IGD values obtained by each algorithm on tri-objective DTLZ instances

Fig.5. Evolution process of HV obtained by each algorithm on tri-objective DTLZ instances

Table. 7. Performance comparison on 30-D ZDT1, ZDT2, ZDT3 and 10-D ZDT6 by MORSGO and MORSGO_C

Test		ZDT1		ZDT2		ZDT3		ZDT4		ZDT6	
problem		MORS	MORS	MORS	MORS	MORS	MORS	MORS	MORS	MORS	MORS
		GO	GO-C	GO	GO-C	GO	GO-C	GO	GO-C	GO	GO-C
Δ	mean	1.11e-3	1.23e-3	8.04e-4	8.02e-4	7.75e-4	7.70e-1	7.16e-1	8.34e-1	4.12e-3	4.65e-3
	std	1.21e-4	5.26e-4	9.61e-5	8.34e-5	4.38e-5	5.00e-5	1.87e-1	3.25e-1	3.23e-4	8.53e-4
HV	mean	9.16e-1	6.12e-1	7.98e-2	6.54e-1	7.75e-2	6.01e-1	8.40e-1	6.00e-1	8.18e-1	6.26e-1
	std	1.31e-2	4.54e-2	7.04e-2	5.25e-2	1.57e-2	4.20e-2	3.21e-2	8.55e-1	1.07e-2	6.66e-2

Table. 8. Performance comparison on 7-D DTLZ1, 12-D DTLZ2 and 22-D DTLZ6 by MORSGO and MORSGO_C

Test problem		DTLZ1		DTLZ2		DTLZ6	
		MORSGO	MORSGO-C	MORSGO	MORSGO-C	MORSGO	MORSGO-C
Δ	mean	3.13	3.20	1.56e-2	1.54e-2	1.88e-2	1.89e-2
	std	3.85	6.14	1.64e-3	2.03e-3	2.70e-3	3.20e-3
HV	mean	4.84e-1	4.22e-1	7.42E-1	4.00e-1	8.84e-1	5.21e-1
	std	1.00e-1	6.27e-1	1.75E-2	2.54e-2	2.43e-2	4.00e-2

3.4.3 Effect of the FCS method

MORSGO uses the FCS method to select a number of feasible non-dominated solutions for next generation evolution. In this subsection, in order to investigate the benefit of the FCS approach on spread of solutions, MORSGO is compared directly with its variant called MORSGO_C that replaces the FCS method by the crowded comparison mechanism. The Δ [8] and HV [39] metrics are employed here. Especially, the Δ metric aims to measure the extent of spread obtained among final solutions, which can examine whether the FCS approach really plays a vital role in MORSGO. Experimental results are compared in Tables 7 and 8.

From results in Tables 7 and 8, it can be visibly observed that MORSGO performs better than its variant MORSGO_C over the spread Δ metric on most of the bi-objective problems, and all of the tri-objective problems, including ZDT1, ZDT4, ZDT6, DTLZ1, DTLZ2, and DTLZ6. For ZDT2, the two algorithms obtain similar results. For ZDT3, MORSGO_C performs slightly better than MORSGO in terms of the Δ mean values. Additionally, the comparison of their HV results in these tables also reflect the superiority of the MORSGO algorithm, because it obtains better HV results on ZDT1, ZDT2, ZDT4, ZDT6, DTLZ1 and DTLZ6. According to these exciting results, the FCS method is essentially validated to make the algorithm have a better spread than that of the crowded comparison mechanism.

Table. 9. Comparison of computation time consumed by each algorithm on each benchmark (unit:s)

Fun.	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6	DTLZ1	DTLZ2	DTLZ6
MORSGO	4120	4032	4015	6616	7662	8205	9369	9804
NSGAI	3450	5256	4422	7024	7508	8364	9062	11025
MOEA/D	3202	4112	3410	7723	6901	9034	8778	9880
GDE3	3740	3894	4065	6406	7553	9845	8002	13045
HMOABC	4204	5435	4240	8548	8023	10452	9645	15055

3.4.4 Comparison of computation time

In order to evaluate the computation efficiency on different benchmark functions, the average results of computing time in 20 sample runs of all algorithms are recorded in Table 9. It is worthy noted the extra spending in the algorithm is different to each other, such as the operator execution, crowed-distance computation, and population communication of the algorithm. This causes the differences in runtimes between different algorithms.

From Table 9, we can see that MORSGO cannot obtain satisfactory results on ZDT1 and ZDT3. However, on more complex problems (ZDT4, ZDT6, DTLZ2 and DTLZ6), MORSGO consumes less time than other algorithms. This may be due to the fact that the branching of main roots and the elimination of dead roots enable the population size of the MORSGO to be dynamically adaptive to the complexity of the objective functions, which can reduce the computational complexity of the optimization process.

4. Multi-objective Copper Strip Burdening Optimization Based on MORSGO

4.1 Multi-objective Burdening Optimization Formulation

● *Decision variables*

Suppose that the copper strip elements consist of main elements and impurities. We firstly define the set of copper strip element, the set of main elements, and the set of impurities as E , E_1 , and E_2 , respectively. Then $E = \{E | E_1 \cup E_2\}$. And the weights of raw materials in burdening process are defined as W , where the new copper (i.e., raw materials) is W_{new} , the remaining copper in the same grade is W_{resg} , the remaining copper in different grade is W_{redg} and the chemical waste is W_{cw} . Apart from the new copper W_{new} , the remaining vector of W is regarded as the old raw materials $W^{old} = \{W^{old} | W_{resg} \cup W_{redg} \cup W_{cw}\}$; The charging time (or feeding sequence) is defined as T ; The melting time is defined as t_m ; Then, the decision variables are given as below:

$$W = (w_1, w_2, \dots, w_i, \dots, w_d) \quad (26)$$

where x_i is charging weight of the i th raw material, d is the number of raw materials;

$$T = (t_1, t_2, \dots, t_i, \dots, t_d) \quad (27)$$

where t_i is charging time of the i th raw material;

$$\begin{aligned} W_{new} &= (w_1, w_2, \dots, w_k) \\ W_{resg} &= (w_{k+1}, w_{k+2}, \dots, w_m) \\ W_{redg} &= (w_{m+1}, w_{m+2}, \dots, w_p) \\ W_{cw} &= (w_{p+1}, w_{p+2}, \dots, w_d) \end{aligned} \quad (28)$$

where k, m, p, n are pre-set positive integers.

● *Minimization of the total cost of raw materials (f_1)*

This objective function is designed to quantify level of cost in the burdening process. The cost of raw materials can be reduced expectantly via regulating charging weight with the appropriate proportion of each element. Then, it can be formulated as:

$$F_1(W) = \sum_{i=1}^d c_i w_i \quad (29)$$

where c_i denotes a coefficients of the cost for the i th raw material, d is the number of raw materials.

In the smelting process, due to the nature of the chemical elements, there is inevitably burning loss of elements, which should be considered for compensation before furnace in the burdening calculation. This burning loss rate is significantly affected by the charging time and the melting time, which causes different compensation weights, affecting the total cost. Then, the compensated cost of raw materials is calculated as:

$$F_2(W, T, t_m) = \sum_{i=1}^d \xi_i w_i h_i (t_m - t_i) + C w_Q h_m (t_m) \quad (30)$$

In order to improve the ratio of material reuse, the reutilized old material in other grades should cater to the production standards of current grade of copper material in melting furnaces. Thus, considering this factor, the proximity metric between the production standards of copper materials in different grades is defined as:

$$F_3 = \sum_{j=1}^p \xi_e x_i \sum_{i=1}^{n_e} |\lambda_{ij} - \lambda_j| \quad (31)$$

where λ_{ij} is the proportion of the i th element of the j th material, λ_i denotes the proportion of the i th element of the original fused mass, and n_e is the element number of the copper strip element set- E . And the total cost can be calculated as:

$$f_1(W, T, t_m) = F_1(W) + F_2(W, T, t_m) + F_3 = \sum_{i=1}^d c_i w_i \xi_i + h_i (t_m - t_i) + C w_Q h_m (t_m) + \sum_{j=1}^p x_e x_i \sum_{i=1}^{n_e} |l_{ij} - l_j| \quad (32)$$

where c_i is the coefficients of the cost for the i th raw material, t_m is the melting time, C is the cost of the original fused mass, w_Q is the weight of the original fused mass, $\eta_i(t_m - t_i) = g_i \ln(h_i t_i + 1)$ is the burning loss function; $\eta_m(t_m) = g_m \ln(h_m t_m + 1)$ is the burning loss function of the original fused mass.

● Maximization of old and waste materials thrown into melting furnace (f_2)

Due to low volume of copper production with a higher price, it is essentially significant to achieve reutilization and recycling of wastes and old copper materials. In line with the principle of standard burdening process, the amount of waste and old copper materials thrown into melting furnaces should be maximized as far as possible. Then the second objective function to represent the amount of waste and old copper materials can be defined as:

$$f_2(W^{old}) = f_2(w_{k+1}, w_{k+2}, L, w_n) = \sum_{i=k+1}^m \alpha_i w_i + \sum_{i=m+1}^p \beta_i w_i + \sum_{i=p+1}^d \delta_i w_i \quad (33)$$

where α , β , and δ denotes the penalty factor of the remainder in the same grade, remainder in different grade and chemical wastes respectively.

● Constraints

(1) The constraints of copper strip elements

Suppose that u_j and l_j are upper and lower boundaries of the j th element respectively, λ_j denotes the proportion of the j th element of the original fused mass. Then, based on the standard burdening process, the constraints of the termed main elements and impurities follow:

$$l_j \leq \sum_{i=1}^d \lambda_{ij} w_i + \lambda_j w_Q \leq u_j, j \in E_1 \quad (34)$$

$$0 \leq \sum_{i=1}^d \lambda_{ij} x_i + \lambda_j w_Q \leq u_j, j \in E_2 \quad (35)$$

where w_Q is the weight of the original fused mass.

(2) The constraint of stock follows:

$$w_i \leq D_i \quad (36)$$

where D_i is safety stock of the i th copper material.

(3) The equality constraints of total charging weight of raw materials are given below:

$$\sum_{i=1}^d w_i + w_Q = G \quad (37)$$

where G is the total weight of copper materials, and the boundary of charging weight w_i is defined as:

$$L_i \leq w_i \leq U_i \quad (38)$$

where U_i , and L_i are upper and lower boundaries of w_i respectively.

(4) The boundary of charging time is defined as:

$$0 < Tl_i \leq t_m - t_i \leq Tu_i \quad (39)$$

$$Tl_m \leq t_m \leq Tu_m \quad (40)$$

where Tu_i , and Tl_i are upper and lower boundaries for charging time of the i th copper material respectively, Tu_m , and Tl_m denotes upper and lower boundaries for melting time respectively.

● The transformation of decision variables

In order to facilitate calculation of the burdening optimization model, the charging time T needs to be transformed into feeding time T_f as following:

$$T_f = t_m^u - T = (t_m - t_1, t_m - t_2, \dots, t_m - t_d) = (t_1', t_2', \dots, t_d') \quad (41)$$

Let $U_i^m = \min\{U_i, D_i\}$, then the proposed multi-objective objective model can be transformed below:

$$\min \{f_1(W, T_f, t_m), -f_2(W^{old})\} \quad (42)$$

$$\begin{aligned} & L_i \leq w_i \leq U_i^m \\ & 0 < Tl_i \leq t_i' \leq Tu_i \\ & Tl_m \leq t_m \leq Tu_m \\ s.t. & \begin{cases} l_j \leq \sum_{i=1}^d l_{ij} w_i + l_j w_Q \leq u_j & j \in E_1 \\ 0 \leq \sum_{i=1}^d l_{ij} w_i + l_j w_Q \leq u_j & j \in E_2 \\ \sum_{i=1}^d w_i + w_Q = G \end{cases} \end{aligned} \quad (43)$$

4.2 Constraints Handling Approach Based on Penalty Function

The constraints including equality and inequality in the proposed model are handled by using the principle of self-adaptive penalty function [6]. According to this method, the number of feasible individuals (i.e., individuals that satisfy all constraints) in current population is self-adaptively tracked to accordingly determine the amount of penalty, which will be imposed on infeasible individuals (i.e., individuals that cannot satisfy all constraints). Accordingly, the involved objective function fitness is modified with distance measure and adaptive penalty. The detailed procedures are presented below.

Step. 1 Make inequality constraint converted into the equality form to combine all the constraints as:

$$H'(x) = \begin{cases} \max\{g_i(x), 0\}, & i = 1, 2, \dots, j \\ \max\{h_i(x) - \sigma, 0\}, & i = j+1, \dots, j+k \end{cases} \quad (44)$$

where j and k are the numbers of inequality and equality functions respectively, σ is a tolerance compensation.

Step. 2 Normalize $v_f(x)$ (i.e., constraint violation)

$$v_f(x) = \frac{1}{k} \sum_{i=1}^k \frac{H_i'(x)}{H_i'^{\max}} \quad (45)$$

where $H_i'^{\max}$ denotes maximum violation of constraint i .

Step. 3 According to Eq.(21), normalize each objective function for each individual as $f_j'(x)$, $j=1, 2, \dots, M$, where M is the number of objectives.

Step. 4 Estimate the distance value of each individual in the objective function j :

$$d_j(x) = \begin{cases} v_f(x) & \text{if } cr = 0 \\ \sqrt{f'_j(x)^2 + v_f(x)^2} & \text{otherwise} \end{cases} \quad (46)$$

where $cr = \frac{\text{Number of feasible individuals}}{\text{population size}}$.

Step. 5 Two additional penalty functions for infeasible individuals are given below:

$$c_j(x) = (1 - cr)X_j(x) + crY_j(x) \quad (47)$$

where

$$\begin{aligned} X_j(x) &= \begin{cases} 0, & \text{otherwise} \\ v_f(x), & \text{if } cr = 0 \end{cases} \\ Y_j(x) &= \begin{cases} 0, & \text{if } x \text{ is a feasible individual} \\ f'_j(x), & \text{if } x \text{ is an infeasible individual} \end{cases} \end{aligned} \quad (48)$$

Step. 6 Then, the combined objective function value for the objective function j is modified as:

$$F_j(x) = d_j(x) + c_j(x) \quad (49)$$

4.3 Implementation of MORSGO for Multi-objective CSBC Problem

To apply the proposed MORSGO to deal with the multi-objective CSBC problem, the following procedures should be made and repeated.

Step. 1 Initialization

(1) Encoding:

Each individual can be encoded into a vector x with $2 \times (d+1)$ variables where d is the number of raw materials. As shown in Table 10, the vector of each individual is combined by the weights of raw materials (W), the weight of original fused mass (w_Q), the charging time (T) and the melting time (t_m). Note that the $(d+1)$ th variable (i.e., w_Q) is usually constant, and can be skipped in the fitness calculation.

Table. 10. Representation of the vector x for each individual

Weight of raw materials				Weight of original fused mass		Charging time			Melting time
w_1	w_2		w_d	w_Q		t_1	t_2	...	t_d
x_1	x_2	...	x_d	x_{d+1}		x_{d+2}	x_{d+3}	...	x_{2d+1}
									x_{2d+2}

(2) Population initialization

Randomize an initial population P_0 with N individuals where each vector can be encoded with constraints by Eqs. (38), (40) and (41). As a result, the vector of each individual consists of $2 \times (d+1)$ variables.

Step. 2 Optimization process

(1) Auxin concentration evaluation

Evaluate each objective value for each individual by Eq.(29) and Eq. (32). And then calculate the normalized fitness and corresponding auxin concentration values by Eq.(4)-Eq.(6), and Eq.(23). And then divide current population into main roots group and lateral roots group according to auxin concentration values.

(2) Main root growth operation

Implement the main root growth operations including main roots regrowth and branching operator as depicted in Step. 3 in Table 2.

(3) Fast non-dominated sorting

Sort the combined population by implementing Step.4-Step.5 as given in Table 2 to construct a new population with different domination level.

(4) Lateral root growth operation

Implement the lateral root growth operation operator to renew population as depicted in Step. 6 in Table 2.

(5) Fast non-dominated sorting and population renewing

Sort the combined population by implementing Step.7 - Step.9 as given in Table 2 where the deteriorated roots are also eliminated.

Step. 3.Termination condition

If the number of current iterations exceeds the maximum pre-set number, stop algorithm execution, otherwise return to Step 2.

Table. 11. Component content of various elements in Bc_4 (unit: %)

Elements	Copper Materials								
	1(W_{new})	2(W_{new})	3(W_{new})	4(W_{new})	5(W_{resg})	6(W_{cw})	λ_i	u_i	l_i
e_1	98.21	16.13	91.23	2.7171	93.42	94.02	96.32	98.65	98.06
e_2	0	88.418	0	4.5647	3.64	1.512	3.54	1.153	0.847
e_3	1.32	0	8.328	0	0.153	0.432	0.034	0.036	0.019
e_4	0.47	5.14	0	95.34	2.749	0	0.084	0.120	0.078
Impurity	0	0.312	0.442	0.0932	0.038	4.045	0.043	0.041	0

Table. 12. Limits of parameters in Bc_4 (unit: %).

Parameters	Copper Materials						
	1(W_{new})	2(W_{new})	3(W_{new})	4(W_{new})	5(W_{resg})	6(W_{cw})	w_Q
L_i/t	3.01	0.15	0.031	0.12	1.32	0.22	/
U_i/t	3.23	0.53	0.042	0.35	11.12	0.84	/
D_i/t	4.34	3.22	0.53	0.74	4.32	1.43	/
TL_i/min	60	30	30	35	35	60	75
Tu_i/min	80	50	50	55	55	80	85
$c_i/(¥/t)$	58000	5800	66300	16100	29000	15100	/
$h_i(h_q)$	0.022	0.021	0.018	0.013	0.023	0.020	0.03

Table. 13. Component content of various elements in Bc_6 (unit: %)

Elements	Copper Materials										
	1(W_{new})	2(W_{new})	3(W_{new})	4(W_{new})	5(W_{resg})	6(W_{redg})	7(W_{resg})	8(W_{cw})	λ_j	u_j	l_j
e_1	98.42	91.2	90.1	0.003	91.3	99.21	93.14	87.63	96.26	99.373	98.32
e_2	0	8.8	0.01	0.005	1.23	0.42	6.347	3.42	0.23	0.25	0.073
e_3	1.57	0	9.68	0	0.01	0.029	0.04	3.75	0.014	0.032	0.031
e_4	0	0	0	0	6.69	0.297	0	0	3.417	0.017	1.531
e_5	0	0	0.21	94.1	0.01	0.027	0.32	0	0.031	0.041	0.023
e_6	0	0	0	5.835	0.12	0.014	0.15	1.05	0.036	0.027	0.022
Impurity	0.01	0	0	0.057	0.64	0.003	0.003	4.15	0.012	0.26	0

Table. 14. Limits of parameters in Bc_6 (unit: %).

Parameters	Copper Materials								
	1(W_{new})	2(W_{new})	3(W_{new})	4(W_{new})	5(W_{resg})	6(W_{resg})	7(W_{redg})	8(W_{cw})	w_Q
L_i/t	3.02	0.17	0.035	0.13	0.012	0.48	1.32	0.31	/
U_i/t	3.41	0.23	0.042	0.41	0.302	9.62	11.2	0.81	/
D_i/t	4.76	1.25	0.52	0.92	3.21	5.17	8.01	1.62	/
Tl_i/min	60	25	30	35	35	40	45	60	75
Tu_i/min	80	45	50	55	55	60	65	80	85
$c_i/(\text{¥/t})$	58500	36200	66500	16300	13100	21200	22200	17200	/
$h_i(h_a)$	0.02	0.02	0.03	0.02	0.02	0.01	0.03	0.02	0.03
$g_i(g_q)$	1.2	1.3	1.1	1.25	1.1	1.2	1.1	1.3	1.2

Table. 15. Component content of various elements in Bc_8 (unit: %)

Elem	Copper Materials												
ents	1	2	3	4	5	6	7	8	9	10	λ_j	u_j	l_j
e_1	95.1	90.5	88.2	0.02	87.0	90.2	92.3	85.1	99.2	86.0	94.2	93.3	93.2
e_2	0	6.3	0.02	0.01	1.23	0.42	6.34	3.45	0.40	2.40	1.20	0.25	0.05
e_3	4.5	0	6.08	0	0.01	0.03	0.03	3.05	0.05	2.00	0.1	0.03	0.03
e_4	0	3.1	0	0	6.21	0.30	0	0	0.02	0	2.07	0.01	1.05
e_5	0.39	0	0.6	94.1	0.51	0.02	0.30	0	0.02	1.5	0.10	0.04	0.12
e_6	0	0	5.1	3.81	0.12	0.01	0.20	1.10	0.01	1.35	0.03	0.02	0.12
e_7	0	0	0	0.05	4.5	7.15	0.12	7.19	0.13	2.45	1.10	1.14	3.03
e_8	0	0.1	0	2.01	0.41	1.87	0.21	0.10	0.15	2.30	1.20	4.90	2.40
Impu	0.01	0	0	0	0.01	0	0.50	0.01	0.01	2.00	0	0.31	0

Table. 16. Limits of parameters in Bc_8 (unit: %).

Parameters	Copper Materials										
	1	2	3	4	5	6	7	8	9	10	w_Q
L_i/t	3.00	0.15	0.03	0.12	0.03	0.20	0.01	0.50	1.5	0.30	/
U_i/t	3.40	0.25	0.04	0.40	0.04	0.30	0.30	10.0	8.50	0.60	/
D_i/t	4.75	1.25	0.55	0.90	0.50	0.50	3.20	6.10	5.20	1.53	/
Tl_i/min	60	25	30	35	30	30	35	45	40	60	75
Tu_i/min	80	45	50	55	50	50	55	65	60	80	85
$c_i/(\text{¥/t})$	58000	36000	66500	16200	66500	16000	13100	22200	21300	17200	/
$h_i(h_a)$	0.02	0.02	0.03	0.02	0.02	0.01	0.02	0.03	0.01	0.02	0.03
$g_i(g_a)$	1.25	1.3	1.10	1.2	1.2	1.0	1.1	1.1	1.0	1.1	1.1

4.4 Results and Analysis

4.4.1 Three CSBC instances with different elements

The proposed MORSGO is evaluated against three CSBC instances with different elements, namely Bc_4, Bc_6 and Bc_8. These scalable instances are derived from actual data of the copper strips automatic production line. The NSGAI, MOEA/D and HMOABC are selected as the compared algorithms because they were widely used for resolving the burdening optimization of copper strips in [6]. In this section, the parameter configurations for MORGO, NSGAI, MOEA/D and HMOABC are the same as Section 3.3..

In the Bc_4 instance, the copper strips production involves 4 elements (i.e., e_1, e_2, e_3 , and e_4) and their contents and corresponding limits are listed in Tables 11 and 12 respectively. In these tables, 1 to 4 denotes the termed W_{new} (i.e., new raw materials), 5 is the W_{resg} (i.e., remainder of the same grade) and 6 is the W_{cw} (i.e., chemical waste). According to the actual melting characteristics, the W_{redg} (i.e., the remaining materials in different grade) is not employed as the burdening materials, thus the penalty factors in f_2 (i.e., Eq.(32)) can be pre-set as: $\alpha=8$, $\beta=0$ and $\delta=50$, the tolerance parameter σ in Eq.(44) can be set to 0.15 and other related parameters can be set as: $G=12t$, $w_Q=3t$ and $C= 80000$ ¥/t.

In the Bc_6 instance, the corresponding contents and limits of the 6 elements (i.e., e_1, e_2, e_3, e_4, e_5 , and e_6) are given in Tables 13 and 14 respectively. Similarly to Bc_4, in these tables, 1 to 4 denotes the W_{new} , 5 and 7 denote the termed W_{resg} , 7 is the W_{redg} and 8 denotes the W_{cw} . According to the actual melting process, the penalty factors α , β and δ in the second objective function f_2 can be pre-set to 20, 5, and 50, respectively. Similar to the Bc_4 instance, the tolerance parameter σ is set to 0.15 and other related parameters are set as: $G=12t$, $w_Q=3t$ and $C= 80000$ ¥/t.

The Bc_8 instance includes 8 elements ($e_1, e_2, e_3, e_4, e_5, e_6, e_7$ and e_8), whose contents and corresponding limits are listed in Tables 15 and 16 respectively. In these tables, 1 to 6 denotes W_{new} , 7 and 8 denotes W_{resg} , 9 represents W_{redg} and 10 denotes W_{cw} . Similarly to BC_6, according to the actual melting process, the penalty factors α , β and δ in the second objective function f_2 can be pre-set to 20, 5, and 50, respectively. And the tolerance parameter σ is set to 0.15 and other related parameters are set as: $G=12t$, $w_Q=3t$ and $C= 80000$ ¥/t.

Table. 17. A set of optimal solutions found by MORSGO on BC_4

Solutions	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	f_1	f_2	μ^k
X1	3.021	0.152	0.033	0.121	1.423	0.735	3.0	25	50	50	45	45	25	0	6.039	4.102e+1	5.412e-2
X2	3.021	0.152	0.033	0.121	1.653	0.741	3.0	20	50	50	45	45	25	0	3.934e+5	5.983e+1	4.821e-2
X3	3.021	0.152	0.033	0.121	1.734	0.840	3.0	20	50	50	45	45	25	0	3.142e+5	5.162e+1	4.661e-2
X4	3.021	0.152	0.033	0.121	1.903	0.840	3.0	20	50	50	45	45	25	0	3.565e+5	4.768e+1	4.255e-2
X5	3.021	0.154	0.032	0.121	2.34	0.840	3.0	20	50	50	45	45	25	0	4.119e+5	6.173e+1	4.120e-2
X6	3.021	0.152	0.033	0.121	3.031	0.840	3.0	45	50	25	50	50	45	0	4.216e+5	5.802e+1	3.913e-2
X7	3.021	0.153	0.033	0.121	3.551	0.840	3.0	20	50	50	45	45	25	0	5.828e+5	5.771e+1	3.891e-2
X8	3.021	0.152	0.033	0.121	3.943	0.840	3.0	20	50	50	45	45	25	0	5.188e+5	5.933e+1	3.800e-2
X9	3.021	0.153	0.033	0.121	4.234	0.830	3.0	20	50	50	50	45	25	0	6.101e+5	6.412e+1	3.722e-2
X10	3.021	0.154	0.033	0.121	4.553	0.830	3.0	20	50	50	45	45	25	0	6.519e+5	6.537e+1	3.641e-2

Table. 18. Results found by MORSGO, NSGAII, MOEA/D and HMOABC on BC_4 (the best values are marked bold)

Solutions	MOEA/D			NSGAII			HMOABC			MORSGO		
	f_1	f_2	μ^k	f_1	f_2	μ^k	f_1	f_2	μ^k	f_1	f_2	μ^k
X1	7.539E+05	5.09E+01	5.36E-02	6.774E+05	5.58E+01	5.36E-02	5.48E+5	6.85E+01	5.41E-02	6.04E+00	4.10E+01	5.41E-02
X2	7.646E+05	5.33E+01	4.81E-02	6.723E+05	5.45E+01	4.65E-02	5.60E+5	6.81E+01	5.11E-02	3.93E+05	5.98E+01	4.82E-02
X3	7.417E+05	4.77E+01	4.74E-02	7.034E+05	6.25E+01	4.34E-02	4.67E+5	6.70E+01	4.81E-02	3.14E+05	5.16E+01	4.66E-02
X4	7.221E+05	4.28E+01	4.25E-02	7.063E+05	6.32E+01	4.16E-02	5.80E+5	6.63E+01	4.42E-02	3.57E+05	4.76E+01	4.26E-02
X5	7.304E+05	4.47E+01	4.13E-02	6.618E+05	4.94E+01	3.89E-02	4.67E+5	5.84E+01	4.01E-02	4.12E+05	6.17E+01	4.12E-02
X6	7.673E+05	5.98E+01	3.83E-02	6.965E+05	5.68E+01	3.34E-02	5.29E+5	5.73E+01	3.86E-02	4.22E+05	5.80E+01	3.91E-02
X7	6.782E+05	5.02E+01	3.64E-02	6.912E+05	5.54E+01	3.33E-02	5.13 E+5	6.72E+01	3.80E-02	5.83E+05	5.77E+01	3.89E-02
X8	7.550E+05	5.62E+01	3.59E-02	7.232E+05	6.37E+01	3.33E-02	5.52E+5	5.64E+01	3.65E-02	5.19E+05	5.93E+01	3.80E-02
X9	6.350E+05	6.09E+01	3.54E-02	7.263E+05	6.44E+01	3.31E-02	5.16E+5	5.61E+01	3.60E-02	6.10E+05	6.41E+01	3.72E-02
X10	6.435E+05	5.30E+01	3.52E-02	6.805E+05	5.02E+01	3.19E-02	5.86E+5	6.23E+01	3.50E-02	6.52E+05	6.53E+01	3.64E-02
Average	7.192E+05	5.20E+01	4.14E-02	6.94E+05	5.76E+01	3.89E-02	5.32E+5	6.28E+01	4.20E-02	4.26E+05	5.66E+01	4.22E-02

Table. 19. A set of optimal solutions found by MORSGO on BC_6

Solution	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	f_1	f_2	μ^k
X1	3.02	0.17	0.036	0.13	0.013	1.32	0.48	0.31	3.0	65	45	50	40	40	50	40	60	0	5.69E+05	1.22E+02	2.76E-02
X2	3.02	0.17	0.035	0.13	0.014	1.33	0.48	0.31	3.0	65	45	50	40	40	50	40	60	0	5.64E+00	1.18E+02	2.72E-02
X3	3.02	0.18	0.035	0.14	0.012	1.32	0.49	0.33	3.0	65	45	50	40	40	50	40	60	0	5.84E+05	1.23E+02	2.69E-02
X4	3.02	0.17	0.036	0.13	0.012	1.32	0.48	0.31	3.0	65	45	50	40	40	50	40	60	0	5.58E+05	1.14E+02	2.69E-02
X5	3.02	0.17	0.035	0.13	0.014	1.32	0.49	0.31	3.0	65	45	50	40	40	50	40	60	0	5.53E+05	1.11E+02	2.65E-02
X6	3.02	0.18	0.036	0.15	0.014	1.33	0.48	0.32	3.0	65	45	50	40	40	50	40	60	0	6.64E+05	1.35E+02	2.80E-02
X7	3.02	0.17	0.035	0.13	0.012	1.32	0.49	0.31	3.0	65	45	50	40	40	50	40	60	0	6.05E+05	1.14E+02	2.79E-02
X8	3.02	0.17	0.035	0.15	0.015	1.32	0.48	0.31	3.0	65	45	50	40	40	50	40	60	0	6.00E+05	1.11E+02	2.75E-02
X9	3.02	0.18	0.035	0.14	0.012	1.33	0.48	0.32	3.0	65	45	50	40	40	50	40	60	0	6.78E+05	1.34E+02	2.75E-02
X10	3.02	0.17	0.036	0.13	0.014	1.32	0.48	0.31	3.0	65	45	50	40	40	50	40	60	0	5.99E+05	1.09E+02	2.74E-02
Average	3.02	0.173	0.035	0.14	0.013	1.32	0.48	0.31	3.0	65	45	50	40	40	50	40	60	0	5.41E+05	1.19E+02	2.73E-02

Table. 20. Results found by MORSGO, NSGAII and MOEA/D on BC_6 (the best values are marked bold)

Solutions	MOEA/D			NSGAII			HMOABC			MORSGO		
	f_1	f_2	μ^k	f_1	f_2	μ^k	f_1	f_2	μ^k	f_1	f_2	μ^k
X1	7.41E+05	5.19E+01	5.29E-02	6.65E+05	1.46E+02	5.27E-02	5.12E+5	6.60E+01	5.33E-02	1.96E+05	4.71E+01	5.54E-02
X2	7.52E+05	4.96E+01	4.75E-02	6.71E+05	1.56E+02	4.57E-02	5.25 E+5	5.50E+01	5.15E-02	4.07E+05	5.91E+01	4.93E-02
X3	7.29E+05	4.65E+01	4.68E-02	6.96E+05	6.12E+01	4.09E-02	5.07E+5	6.00E+01	4.32E-02	3.10E+05	5.10E+01	4.77E-02
X4	7.10E+05	4.36E+01	4.20E-02	7.16E+05	6.30E+01	4.27E-02	5.03E+5	6.62E+01	4.32E-02	3.52E+05	4.05E+01	4.36E-02
X5	7.18E+05	4.17E+01	4.08E-02	6.55E+05	5.42E+01	3.83E-02	4.98E+5	7.06E+01	4.15E-02	3.88E+05	6.10E+01	4.22E-02
X6	7.39E+05	5.83E+01	3.78E-02	7.19E+05	5.33E+01	3.27E-02	5.09E+5	5.63E+01	3.76E-02	4.17E+05	5.74E+01	4.00E-02
X7	7.55E+05	5.44E+01	3.56E-02	6.99E+05	4.83E+01	3.27E-02	5.43 E+5	4.22E+01	3.28E-02	6.02E+05	6.34E+01	3.98E-02
X8	8.22E+05	6.60E+01	3.51E-02	6.84E+05	6.23E+01	3.28E-02	5.22E+5	4.05E+01	3.19E-02	5.12E+05	5.86E+01	3.89E-02
X9	6.91E+05	6.09E+01	3.47E-02	6.89E+05	6.19E+01	3.25E-02	5.76E+5	4.06E+01	3.11E-02	5.76E+05	5.71E+01	3.81E-02
X10	7.01E+05	5.74E+01	3.45E-02	6.74E+05	4.91E+01	3.14E-02	6.42E+5	4.13E+01	3.11E-02	6.44E+05	6.46E+01	3.72E-02
Average	7.36E+05	5.30E+01	4.08E-02	6.87E+05	1.56E+02	3.83E-02	6.12E+5	5.40E+01	3.97E-02	4.31E+05	5.60E+01	4.32E-02

Table. 21. A set of optimal solutions found by MORSGO on BC_8

Solution	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	f_1	f_2	μ^k
X1	3.00	0.15	0.03	0.13	0.03	0.13	0.01	0.45	1.50	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	6.78E+05	1.41E+02	2.78E-02
X2	3.00	0.15	0.03	0.13	0.03	0.13	0.01	0.45	1.50	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	5.65E+00	1.16E+02	2.79E-02
X3	3.20	0.15	0.030	0.13	0.03	0.13	0.01	0.50	1.50	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	5.80E+05	1.11E+02	2.82E-02
X4	3.20	0.15	0.035	0.15	0.003	0.13	0.01	0.50	1.45	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	6.34E+05	1.20E+02	2.88E-02
X5	3.20	0.18	0.035	0.15	0.003	0.13	0.01	0.50	1.45	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	5.88E+05	1.19E+02	3.43E-02
X6	3.20	0.18	0.035	0.13	0.003	0.15	0.01	0.50	1.45	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	5.67E+05	1.24E+02	3.55E-02
X7	3.00	0.15	0.035	0.15	0.003	0.15	0.01	0.50	1.50	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	5.65E+05	1.12E+02	3.76E-02
X8	3.00	0.15	0.030	0.13	0.003	0.13	0.01	0.50	1.50	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	5.56E+05	1.16E+02	4.22E-02
X9	3.00	0.15	0.030	0.13	0.003	0.13	0.01	0.45	1.50	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	6.61E+05	1.38E+02	4.76E-02
X10	3.00	0.18	0.030	0.13	0.003	0.13	0.01	0.45	1.50	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	5.50E+05	1.22E+02	4.78E-02
Average	3.08	0.159	0.032	0.136	0.003	0.134	0.01	0.48	1.48	0.30	3.0	65	30	40	50	50	40	55	65	60	80	75	6.61E+05	1.21E+02	3.58E-02

Table. 22. Results found by MORSGO, NSGAII and MOEA/D on BC_8 (the best values are marked bold)

Solutions	MOEA/D			NSGAII			HMOABC			MORSGO		
	f_1	f_2	μ^k	f_1	f_2	μ^k	f_1	f_2	μ^k	f_1	f_2	μ^k
X1	5.88E+05	1.26E+02	2.21E-02	7.02E+05	1.02E+02	2.33E-02	6.06E+05	1.33E+02	2.66E-02	6.78E+05	1.41E+02	2.78E-02
X2	6.99E+05	7.82E+01	2.32E-02	6.66E+05	6.54E+01	2.52E-02	5.47E+05	1.14E+02	2.71E-02	5.65E+05	1.16E+02	2.79E-02
X3	6.13E+05	5.79E+01	2.44E-02	7.32E+05	8.24E+01	2.79E-02	5.57E+05	1.09E+02	2.82E-02	5.80E+05	1.11E+02	2.82E-02
X4	6.43E+05	5.19E+01	2.63E-02	6.22E+05	6.32E+02	2.85E-02	5.46E+05	9.12E+01	2.89E-02	6.34E+05	1.20E+02	2.88E-02
X5	7.35E+05	1.04E+02	3.01E-02	6.32E+05	6.12E+01	2.99E-02	5.51E+05	9.86E+01	3.32E-02	5.88E+05	1.19E+02	3.43E-02
X6	5.44E+05	1.11E+02	3.22E-02	7.14E+05	7.54E+01	3.15E-02	5.65E+05	1.10E+02	3.61E-02	5.67E+05	1.24E+02	3.55E-02
X7	5.65E+05	5.42E+01	3.37E-02	6.66E+05	1.01E+02	3.39E-02	5.44E+05	9.14E+01	3.71E-02	5.65E+05	1.12E+02	3.76E-02
X8	6.23E+05	6.11E+01	3.58E-02	6.65E+05	1.02E+02	3.81E-02	5.50E+05	1.10E+02	4.28E-02	5.56E+05	1.16E+02	4.22E-02
X9	6.10E+05	6.12E+01	3.69E-02	6.60E+05	8.32E+01	4.00E-02	5.45E+05	1.41E+02	4.53E-02	6.61E+05	1.38E+02	4.76E-02
X10	7.21E+05	1.12E+02	4.02E-02	6.88E+05	6.12E+01	4.02E-02	5.47E+05	1.18E+02	4.77E-02	5.50E+05	1.22E+02	4.78E-02
Average	6.34E+05	8.18E+01	3.05E-02	6.75E+05	1.37E+02	3.39E-02	5.56E+05	1.12E+02	3.53E-02	5.94E+05	1.22E+02	3.58E-02

4.4.2 Results on Bc_4

A set of 10 best solutions obtained by MORSGO are given in Table 17 where corresponding objective function values and the best μ^k are also listed. From Table 17, we can observe that most optimal solutions obtained by MORSGO have an excellent convergence that the charging times of optimal solutions obtained are nearly consistent, and only a tiny proportion of them have different feeding amount. From another point of view, this diversity of feeding amount provides decision makers with more burdening choices. The item μ^k reflects the comprehensive quality of the solution. That is, the greater value of μ^k means the higher priority of the solution.

Then, the results of objective function values and the μ^k values corresponding to ten best solutions obtained by MORSGO, HMOABC, NSGAI and MOEA/D are presented and compared in Table 18 where the μ^k values are sorted in descending order. The results show that the proposed MORSGO performs competitively over HMOABC, NSGAI and MOEA/D on f_1 and μ^k . MORSGO always obtains the lowest cost f_1 over the ten candidate solutions, and also gets satisfactory results on f_2 with a second ranking, only slightly worse than HMOABC. Moreover, MORSGO has significantly promising comprehensive performance of Pareto-based solutions with the highest value of μ^k .

4.4.3 Results on Bc_6

Table 19 gives best solutions, corresponding objective function values and best μ^k values obtained by MORSGO. Similar to the results of Bc_4, it can be observed that most optimal solutions have a good convergence. Furthermore, there are a greater proportion of solutions with different feeding amount, which means a better diversity of feeding amount values. Accordingly, decision makers have more burdening choices. Table 20 gives comparative results of MORSGO, NSGAI, MOEA/D and HMOABC on Bc_6. From this table, it can be observed that MORSGO obtains the best results on the total cost of raw materials (f_1) and the comprehensive performance (μ^k), w.r.t the lowest average value of f_1 and the highest average value of μ^k . NSGAI obtains the best average value of f_2 (the old and waste materials thrown into melting furnace). For f_2 , MORSGO also obtains satisfactory results with the second ranking in compared algorithms.

4.4.4 Results on Bc_8

Table 21 lists ten best solutions yielded by MORSGO on the Bc_8 instance, involving two objective function values and corresponding μ^k . It is apparent from Table 21 that the charging times of the solution including x_{11} , ..., x_{22} show significant consistency, and corresponding feeding amount i.e., x_1 , ..., x_{11} remain almost the same. This also indicates that the proposed method obtains satisfactory convergence of solutions. Table 22 gives

comparative results of MORSGO, NSGAI, MOEA/D and HMOABC on Bc_8. From this table, we can observe that MORSGO obtains best results regarding maximization of old and waste materials thrown into melting furnace (f_2) and the comprehensive performance μ^k . As for the total cost of raw materials (f_1), MORSGO also gets the second best optimal solutions, only slightly worse than those of HMOABC. Generally, according to above results, it can be drawn that MORSGO gets an excellent performance on the Bc_8 instance. This also indicates that the proposed MORSGO is an effective optimization paradigm to resolve the multi-objective BCSC problem.

4.4.5 Results Analysis

Then, the HV metric is applied to further verify the performance advantage of the proposed scheme. Inspired from [13], for a set of solutions R and an objective function f_i , $f_{max}^i(R)$ and $f_{min}^i(R)$ are defined as a lower and an upper bound of f_i in R , respectively. Here, the $f_{min}^i(R)$ can be set to 0, and $f_{max}^i(R)$ is calculated as

$$f_{max}^i(R) = \max\{f_i(r) : r \in R\} + 0.05 * (\max\{f_i(r) : r \in R\} - \min\{f_i(r) : r \in R\}) \quad (50)$$

The $f_{max}^i(R)$ can be used as the reference points for the HV-metric calculation. Then, the HV results on the CSBC instances are given in Table 23. From this table, it can be observed that MORSGO gets satisfactory HV values on Bc4 and Bc6. And on Bc4, MORSGO does better than MOEA/D and NSGAI, and obtains close results to HMOABC. These results further verify the performance of MORSGO on BSBC.

Table. 23. Comparison of HV values obtained by each algorithm on each problem

Test problems	Bc4		Bc6		Bc8	
	mean	std	mean	std	mean	std
MORSGO	5.52E-01	1.43E-01	5.22E-01	1.21E-01	4.76E-01	2.02E-01
MOEA/D	5.11E-01	2.94E-01	5.11E-01	9.78E-02	4.11E-01	4.18
NSGAI	4.65E-01	4.03E-01	4.79E-01	5.52E-01	4.44E-01	2.93E-01
HMOABC	5.18E-01	2.11E-01	5.25E-01	2.00E-01	4.60E-01	3.01E-01

5. Conclusions

This paper proposes and develops a new multi-objective plant-inspired algorithm called MORSGO for dealing with the cropper strips burdening calculation. MORSGO adopts auxin-regulated mechanism and a set of growth operators derived from different types of roots to drive population evolution and obtain optimal solutions. This new framework essentially has merits of appropriate balance between exploration and exploitation, and adaptive

population variation, which can reduce redundant computation during algorithm executing. When all solutions are processed, the fast non-dominated sorting approach combined with the farthest-candidate selection is employed to select new competent population for the next iteration.

We have experimentally compared MORSGO with several representative multi-objective algorithms on a set of multi-objective benchmarks. Experimental results show that MORSGO has a promising ability of maintaining better population diversity and accordingly obtains better convergence and spread of non-dominated solutions. The CSBC model is constructed where two conflicting constrained objectives are formulated, and then MORSGO is applied for resolving two different-scale CSBC instances. Computational results show the effectiveness and efficiency of the proposed algorithm in solving actual burdening optimizations.

Note that MORSGO sometimes suffers the problem of being easily trapped into local optima on some test problems. Accordingly, we do not assert that MORSGO is always superior to other multi-objective algorithms. The strengths and weaknesses of MORSGO need to be investigated specifically based on the characteristics of test problems. A comprehensive sensitivity analysis of parameters of the algorithm, the theoretic analysis of algorithm complexity and applying MORSGO for more industrial optimization problems will be highlighted in our future work.

Acknowledgements

The authors would like to thank the editors and anonymous reviewers for their helpful comments and suggestions on improving the quality of this paper. This work is supported by the National Natural Science Foundation of China under Grant No. 61503373 and No. 61572123; National Science Foundation for Distinguished Young Scholars of China under Grant No. 71325002 and No. 6122501; Natural Science Foundation of Liaoning Province under Grand No. 2015020002; and Fundamental Research Funds for the Central Universities No. N161705001.

References

- [1] A. J. Andersson, A. T. Andersson Margareta, P. G. Jonsson, Use of an optimisation model for the burden calculation for the blast furnace process. *Scandinavian J of Metallurgy*, 2004, 33(3): 172-182.
- [2] J. J. Wang, X. U. Kai, Development Research of China's Copper Processing Industry, *Nonferrous Metals Processing*, 2009.
- [3] D. Govender, D. Meadows, D. Lelinski, et al., Large flotation cells in copper processing: Experiences and considerations, *Mining Engineering*, 2014.
- [4] J. C. Wang, J. M. Zhang, N. Li, et al., Effect of Potential Function on Molecular Dynamics Simulation of Copper

- Processing, Key Engineering Materials, 2009, 407-409:368-371.
- [5] J. Du, J. X. Feng, S. U. Li, Research on RBF network based prediction model for alloy burdening, *Steelmaking*, 2007.
 - [6] H. Zhang, Y. Zhu, W. Zou, X. Yan, A hybrid multi-objective artificial bee colony algorithm for burdening optimization of copper strip production, *Applied Mathematical Modelling*, Vol.36, No. 6, June 2012, pp. 2578-2591.
 - [7] H. Zhang, Y. Zhu, X. Yan. Multi-hive artificial bee colony algorithm for constrained multi-objective optimization[C]// *Evolutionary Computation (CEC)*, 2012 IEEE Congress on. IEEE, 2012:1-8.
 - [8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA- II , *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182 – 197, Apr. 2002.
 - [9] Q. Zhang, W. Liu and H. Li, The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances in P. of Congress on Evolutionary Computation (CEC 2009), Norway, 2009, pp. 203-208.
 - [10] E. Zitzler, M. Laumanns, and L. Thiele, SPEA2: Improving the strength pareto evolutionary algorithm, in *Proc. EUROGEN 2001: Evolutionary Methods Design Optimization Control Appl. Ind. Problems*, Athens, Greece, 2002, pp. 95–100.
 - [11] Carlos A. Coello Coello, Gregorio Toscano Pulido, Maximino Salazar Lechuga, Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 2004,8(3):256–279.
 - [12] M. Gong, L. Jiao, H. Du, L. Bo, Multiobjective immune algorithm with nondominated neighbor-based selection, *Evolutionary Computation*, 2008, 16(2):225 - 255.
 - [13] J. J. Palacios, I. González-Rodríguez, C. R. Vela, et al. "Robust multiobjective optimisation for fuzzy job shop problems". *Applied Soft Computing*, 2016.
 - [14] Z. H. Zhan, J. Li, J. Cao, et al., Multiple Populations for Multiple Objectives: A Coevolutionary Technique for Solving Multiobjective Optimization Problems, *IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics A Publication of the IEEE Systems Man & Cybernetics Society*, 2012, 43(2):445-463.
 - [15] M. Asafuddoula, T. Ray, R. Sarker, A Decomposition Based Evolutionary Algorithm for Many Objective Optimization, *IEEE Transactions on Evolutionary Computation*, 2014, 19:445-460.
 - [16] R. Karban, Plant behaviour and communication, *Ecology Letters*, 2008, 11:727–739.
 - [17] H. de Kroon, L. Mommer, Root foraging theory put to the test, *Trends in Ecology and Evolution*, 2006, 21: 113–116.
 - [18] O. Falik, P. Reides, M. Gersani, A. Novoplansky, Root navigation by self inhibition, *Plant, Cell and Environment*, 2005, 28:562–569.
 - [19] O. Leyser, Dynamic integration of auxin transport and signalling. *Current Biology*, 2006,16(11), R424-R433.
 - [20] S.W. Kembel, J.F. Cahill, Plant phenotypic plasticity belowground: a phylogenetic perspective on root foraging trade-offs, *American Naturalist*, 2005, 166: 216–230.
 - [21] S.W. Kembel, H. De Kroon, J.F. Jr. Cahill, L Mommer, Improving the scale and precision of hypotheses to explain root foraging ability, *Annals of Botany*, 2008, 101(9):1295-1301.
 - [22] L. Ma, K. Hu, Y. Zhu, H. Chen, A Hybrid Artificial Bee Colony Optimizer by combining with life-cycle, Powell's search and crossover, *Applied Mathematics and Computation*, 2015, 252:133–154,.
 - [23] D. P. Mitchell, Spectrally optimal sampling for distribution ray tracing, in *Proc. ACM SIGGRAPH*, 1991: 157–164.
 - [24] B. Chen, W. Zeng, Y. Lin, et al., A New Local Search-Based Multiobjective Optimization Algorithm, *IEEE Transactions on Evolutionary Computation*, 2014, 19(1):1-1.

- [25] M. A. Abido, Multiobjective evolutionary algorithms for electric power dispatch problem. *IEEE Transactions on Evolutionary Computation*, 2006, 10(3): 315-329.
- [26] B. Y. Qu, P. N. Suganthan, Constrained Multi-Objective Optimization Algorithm with Ensemble of Constraint Handling Methods, *Engineering Optimization*, Vol. 43, No. 4, p. 403, 2011.
- [27] H. Zhang, Y. Zhu, H. Chen. "Root growth model: a novel approach to numerical function optimization and simulation of plant root system", *Soft Computing*, 2014, 18(3):521-537.
- [28] L. Ma, Y. Zhu, Y. Liu, L. Tian, A novel bionic algorithm inspired by plant root foraging behaviors, *Applied Soft Computing*, 2015, 37(C):95-113.
- [29] A. Banks, J. Vincent, K. Phalp, Natural strategies for search, *Natural Computing*, 2009, 8: 547–570.
- [30] J. G. Dubrovsky, M. Sauer, S. Napsucialy-Mendivil, M. G. Ivanchenko, et al., Auxin acts as a local morphogenetic trigger to specify lateral root founder cells. *Proc Natl Acad Sci USA*, 2008, 105:8790–8794..
- [31] M. Laskowski, S. Biller, K. Stanley, T. Kajstura, R. Prusty, Expression profiling of auxin-treated Arabidopsis roots: toward a molecular analysis of lateral root emergence. *Plant Cell Physiol*, 2006, 47:788–792.
- [32] G. Qin, H. Gu, Y. Zhao, Z. Ma, G. Shi, et al., An indole-3-acetic acid carboxyl methyltransferase regulates Arabidopsis leaf development. *Plant Cell*, 2005, 17:2693–2704.
- [33] R. Turkington, R.S. Hamilton,, C. Gliddon, Withinpopulation variation in localized and integrated responses of *Trifolium repens* to biotically patchy environments. *Oecologia*, 1991, 86, 183–192.
- [34] N. Chen, W. N. Chen, Y. J. Gong, et al. An Evolutionary Algorithm with Double-Level Archives for Multiobjective Optimization, *IEEE Transactions on Cybernetics*, 2015, 45(9):1851-1863.
- [35] I. C. Parmee, D. Cvetkovic, Preferences and their application in evolutionary multiobjective optimization, *IEEE Trans. Evol. Comput.*, 2002, 6(1): 42–57.
- [36] E. Zitzler, Evolutionary algorithms for multiobjective optimisation: methods and applications, PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.
- [37] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable multi-objective optimization test problems, in: *Proceedings of Congress on Evolutionary Computation*, 2002, pp. 825–830.
- [38] A.M. Zhou, Y.C. Jin, Q.F. Zhang, B. Sendhoff, E. Tsang, Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion, in: *Proceedings of Congress on Evolutionary Computation*, 2001, pp. 3234–3241.
- [39] E. Zitzler, L. Thiele, Multi-objective evolutionary algorithms: a comparative case study and the strength Pareto approach, *IEEE Transactions on Evolutionary Computation* 3 (4) (1999) 257–271.
- [40] M. Gong, J. Liu, H. Li, Q. Cai, L. Su, A Multiobjective Sparse Feature Learning Model for Deep Neural Networks, *IEEE Transactions on Neural Networks & Learning Systems*, 2015, 26(12):3263-3277.
- [41] L. Li, X. Yao, R. Stolkin, M. Gong, S. He, An evolutionary multiobjective approach to sparse reconstruction, *IEEE Transactions on Evolutionary Computation*, 2014, 18(6), 827-845.
- [42] M. Gong Q. Cai, X. Chen, L. Ma, Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition, *IEEE Transactions on Evolutionary Computation*, 2014, 18(1), 82-97.
- [43] R. Varela, C. R. Vela, J. Puente, et al. A knowledge-based evolutionary strategy for scheduling problems with bottlenecks, *European Journal of Operational Research*, 2003, 145(1):57-71.

- [44] R. Varela, C. R. Vela, J. Puente, et al. Solving Job-shop Scheduling Problems by Means of Genetic Algorithms, 2001.
- [45] H. Karshenas, R. Santana, C. Bielza, P. Larranaga, Multiobjective Estimation of Distribution Algorithm Based on Joint Modeling of Objectives and Variables, *IEEE Transactions on Evolutionary Computation*, 2014, 18(4):519-542.
- [46] X. Bing, M. Zhang, W. N. Browne, Particle swarm optimization for feature selection in classification: a multi-objective approach, *IEEE Transactions on Cybernetics*, 2013, 43(6):1656-1671.
- [47] S. Kukkonen and J. Lampinen, "Performance assessment of generalized differential evolution 3 with a given set of constrained multi-objective test problems," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1943–1950.
- [48] V. L. Huang, P. N. Suganthan, and J. J. Liang, "Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems," *Int. J. Intell. Syst.*, vol. 21, no. 2, pp. 209–226, Feb. 2006.
- [49] A. Hodge, G. Berta, C. Doussan, F. Merchan, M. Crespi, Plant root growth, architecture and function. *Plant And Soil*, 321(1-2), 153-187, 2009.
- [50] Z. Cui, D. Liu, J. Zeng, "Using splitting artificial plant optimization algorithm to solve toy model of protein folding", *Journal of Bioinformatics and Intelligent Control*, vol. 2(2), pp. 146–155, 2013.

Fig.1 The flowchart of MORSGO algorithm