Received January 7, 2020, accepted January 23, 2020, date of publication February 10, 2020, date of current version February 18, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2972631

A Vascular Invasive Tumor Growth Optimization Algorithm for Multi-Objective Optimization

JING ZHOU[1], SHOUBIN DONG[1], DEYU TANG[2], AND XIAOFEI WU[1]
[1]Communication and Computer Network Lab of GD, South China University of Technology, Guangzhou 510641, China
[2]Department of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, China

Corresponding author: Shoubin Dong (sbdong@scut.edu.cn)

This work was supported in part by the Natural Science Foundation of Guangdong under Grant 2015A030308017, in part by the Guangdong Science and Technology Key Project under Grant 2015B010131009, and in part by the National Natural Science Foundation of China under Grant 61976239.

ABSTRACT Multi-objective optimization problems (MOPs) have received much attention in recent years. To deal with these problems, many multi-objective optimization algorithms have been proposed, especially the heuristic algorithms. In this paper, we proposed a multi-objective optimization algorithm called vascular invasive tumor growth optimization (VITGO), which based on the invasive tumor growth optimization and utilized a vascular mechanism to solve the MOPs. The newly proposed algorithm contains two parts: the vascular units and tumor cells. The former ones are utilized to record the Pareto solutions of the MOPs and define the search direction, and the latter ones are utilized to co-operate with vascular units to search deeper and wider. The mechanisms in the VITGO algorithm includes: endpoint generation, approximate Pareto front guidance, opposite searching, and adaptively detailed searching. Experiments showed that compared with other state-of-the-art multi-objective optimization algorithms, VITGO performs better in convergence and diversity.

INDEX TERMS Evolutionary computation, swarm intelligence, multi-objective optimization, endpoint generation, vascular mechanism.

I. INTRODUCTION

Multi-objective optimization problems (MOPs) are decision problems with multiple criteria. These criteria will appear as objective functions and constrained functions. Recently, many multi-objective optimization algorithms such as U-NSGA-III [1] and B-NSGA-III [2] have been proposed to solve these problems. The objective functions of a MOP conflicts with each other such that there is not a global best solution of the whole problem. It means, if one of the objective functions achieves its global best, the others may not. Constrained functions are mostly provided by users. There are two technical ways to solve the MOPs: 1) transforming a MOP into a single-objective optimization problem and solving the latter problem; 2) solving a MOP by determining its Pareto solutions directly. A Pareto solution [3] is a "not-bad solution" of a multi-objective optimization problem, which was introduced more than 100 years ago. Whether an algorithm can determine all of the Pareto solutions of a MOP and how good the Pareto front [3] it can establishes, are the criteria to evaluate the performance of it. They will be discussed in the next section.

The associate editor coordinating the review of this manuscript and approving it for publication was Ran Cheng.

The invasive tumor growth optimization algorithm/ITGO [4] is an algorithm that imitates the behaviour of tumor cells to solve single-objective optimization problems by searching out their global best solutions. In this paper, we introduced the vascular mechanism into ITGO and proposed a multi-objective vascular invasive tumor growth optimization, which both imitates the behaviour of vascular units and tumor cells to solve multi-objective optimization problems. The main innovation and benefit of the proposed VITGO algorithm includes the following:

1) First proposal of a search model that based on the behaviour of vascular units and tumor cells, including the growth, update and pruning operations of vascular tissues, the invasion, growth, dormancy, death of tumor cells, the interactions between different types of tumor cells, and the interactions between vascular units and tumor cells.

2) Specific proposal of a series of search schemes to solve multi-objective optimization problems. i) Endpoint detection and generation. During the process of solving multi-objective optimization problems, there may appear some

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

discrete or discontinuous parts of the approximate Pareto front in the intermediate results. The endpoints of these discrete or discontinuous structure are more likely to help search wider and farther. VITGO has an endpoints detection scheme to identify these discrete or discontinuous parts to generate new vascular units and help search. ii) Approximate PF guidance. Since the current approximate Pareto front is the most effective part of the whole population in the current generation, especially the endpoints located on the approximate Pareto front, it can be utilized to guide the movement of other individuals. VITGO has three different ways for the approximate PF guidance: invasion from the guiders to enter into the un-searched areas, moving around the guiders and retain the better ones to search wider, and tiny search around the guiders to search deeper and maintain the diversity. iii) Pruning schemes. In the VITGO algorithm, the amount of valid individuals will expands as time goes by, and the density of these individuals will be too large to avoid the computational redundancy. Therefore, a pruning scheme based on a threshold δ is provided to maintain proper density. iv) Opposite search steps. To be more effective, opposite search steps will be utilized by invasive cells and the better one will be retained.

3) Associative proposal of a series of search schemes to make the algorithm work more in-depth and exhibit stability. i) The adaptively detailed search by dormant cells. These individuals only search by 1 dimension in the decision space at a time to avoid missing any feasible search direction, and eliminate the invalid search directions. ii) If an individual reaches the boundary, it will random walk with the corresponding dimension that exceeds the boundary and other variables remain unchanged.

These search schemes can both help for the convergence and diversity of the VITGO algorithm.

The remainder of this paper is described as follows: Section 2 introduces the multi-objective optimization algorithms in detail. Section 3 is the definition of multi-objective optimization problems. Section 4 introduces the proposed VITGO algorithm in detail. Section 5 discusses the experiments and analysis. Section 6 presents the conclusion.

II. RELATED WORK

In recent years, many multi-objective optimization algorithms have been proposed to deal with MOPs. According to paper [5], these algorithms can be divided into four types: no-preference methods, priori methods, posteriori methods and interactive methods. The main difference between these four types of methods is whether there is prior knowledge or not, and whether there is a user's preference or not. Jurgen *et al.* put forward similar views in paper [6]. The first type is to solve MOPs without user's preference, such as the simplex method, the integer programming and most of the evolutionary multi-objective optimization algorithms. The second type is to solve MOPs with prior knowledge (mostly appearing as weights), such as the multi-objective ant colony optimization [7], the variable length brain storm optimization algorithm [8], and the interactive preference-based multi-objective evolutionary algorithm [9]. The third type is to solve MOPs with some posteriori knowledge (mostly appearing as weights), such as the multi-objective virtual machine consolidation algorithm in paper [10]. The fourth type is to solve MOPs with interactive operations, such as the expert system set up online.

To analyze these algorithms more, we further divided them into two categories based on the technical skills. The first category is to transform a multi-objective optimization problem into a single-objective problem (mostly with weights) and solve the latter problem. The second category is to determine the Pareto solutions of a MOP directly. Since there is not a global best solution in a multi-objective optimization problem, the first category can only determine a point located on the true Pareto front (mostly with weights as priori knowledge), and ignores the other Pareto solutions. A typical example is the multi-objective ant colony optimizaton [7], which utilizes the user-preference as weights to identify a global best solution. Other algorithms such as the multi-objective lazy ant colony optimization [11], the weight-aggregation multi-objective particle swarm optimization [12] and the user-preference multi-objective optimization [13], all utilizes the user preference as weights to search for a global optima. The second category searches out the complete Pareto solutions directly, without utilizing the user preferences, priori knowledge or posteriorior knowledge. A typical example is the vector evaluated genetic algorithm [14], which utilizes the search schemes of genetic algorithm to solve MOPs. Other similar algorithms include: the nondominated sorting genetic algorithm [15], which is based on the genetic algorithm and utilizes the nondominated sorting scheme to determine the Pareto front of MOPs; the nondominated sorting genetic algorithm - II [16], which proposes a nondominated fast sorting scheme and a selection operator to solve MOPs; the nondominated sorting genetic algorithm - III [17], which further proposes a reference-point-based scheme to solve more complex MOPs; the multi-objective particle swarm optimization [18], which imitates the behaviours of a real swami, selects elite particles by a probabilistic crowding radius-based scheme and a multilevel sieve, to determine a significant Pareto front; there is another version of MOPSO [19], which utilizes a secondary repository for storage and guidance, as well as a crossover operator, to solve the MOPs; the multi-objective artificial bee colony (MOABC) [20], [21], which imitates the behaviour of artificial bee colony to determine the Pareto front; the multi-objective gravitational search algorithm [22], which solves the MOPs with the mechanism of gravity acceleration and searches out for Pareto solutions; the multi-objective teaching-learning based optimization algorithm [23], which is based on the evolutionary mechanisms of teaching-learning-based-optimization and a nondominated sorting to search out for a significant Pareto front; the multi-objective shuffled frog-leaping algorithm [24], which utilizes the memetic schemes of shuffled frog-leaping algorithm and the scheme of crowding distance to determine a significant

Pareto front; the multi-objective bacterial foraging optimization [25], which is based on the mechanisms of bacterial foraging optimization and a new novel health sorting approach to determine the Pareto optima; the multi-objective immune algorithm [26], which is based on the features of biological immune system, and utilizes a novel gene fragment recombination and several antibody diversification schemes, to determine a significant Pareto front; the hybrid evolutionary immune algorithm [27], which is an enhancement of MOIA with hybrid approaches; the multi-objective sine-cosine algorithm [28], which is based on the mechanism of sine-cosine algorithm and utilized an elitist non-dominated sorting method to deal with MOPs; the multi-objective grey wolf optimizer [29], which is based on the characteristics of grey wolf optimizers, and utilizes a nondominated storing method and a series of leader selection schemes, to determine a significant Pareto front; the multi-objective dragonfly algorithm [30], which imitates the habitats of dragonflies, utilizes a series of schemes to store and retrieve archives, and employees a roulette-wheel mechanism, to deal with MOPs; the interval multi-objective quantum-inspired cultural algorithm [31], which is based on the mechanisms of cultural algorithm and utilizes a novel quantum-inspired strategy to search for a significant Pareto front; the multi-objective optimization algorithm based on artificial algae [32], which is based on the search technique of artificial algae algorithm and applies an elitist nondominated sorting scheme and a crowding distance scheme to obtain the Pareto optima. Most of these multi-objective optimization algorithms belong to evolutionary algorithms that 1) imitates biological characteristics to design a search model and proposes a series of novel schemes; 2) utilizes a whole population to get better solutions by iterations; and 3) considers both the convergence and diversity. They belong to the second category that search for a significant Pareto front.

However, weights cannot be utilized as the criteria to distinguish the first category from the second category, since a series of multi-objective algorithms also utilize weights to determine a complete Pareto front, such as: the multi-objective bat algorithm [33], which utilizes K randomly chosen weights to solve K-objective problems; the multi-objective evolutionary algorithm based on decomposition [34], [35], which decomposes a MOP into several sub-problems and solve them by aggregation, including the weighted sum approach, the Tchebycheff approach and the boundary intersection approach; the multi-objective evolutionary algorithm based on decomposition with adaptive replacement strategies [36], which utilizes a sigmoid function to adaptively adjust replaced neighbors of individuals in MOEA/D; the multi-objective evolutionary algorithm based on decomposition with composite operator selection [35], which introduces four types of cross-mutate operations for evolutionary computations; the multi-objective crow search algorithm [37], which utilizes a set of determined weight vectors and employees the max-min strategy; All of them are trying to search for a significant Pareto front while applying weights. To guarantee the robustness of different multi-objective optimization algorithms and solve the dynamic multi-objective optimization problems [38], prediction models such as moving average, autoregressive and single exponential smoothing can also be aggregate with weights to achieve this goal. Since these multi-objective optimization algorithms do not aggregate the MOPs into single-objective optimization problems to find out a global best solution, they all belong to the second category.

In this paper, we focused on the multi-objective optimization algorithms that search for a significant Pareto front with evolutionary schemes. A multi-objective vascular invasive tumor growth optimization algorithm was proposed based on the behaviours of vascular units and tumor cells, and the interactions between these individuals. A series of schemes were proposed to deal with MOPs. It will be discussed in detail in the next section.

III. PROBLEM DEFINITION

A multi-objective optimization problem refers to a problem that contains multiple conflicting objective-functions and the corresponding constraint functions. During the processing, these functions should be considered at the same time. In this paper, we utilize $f_1, f_2, \ldots, f_n$ to represent the 1^{th}, 2^{th}, …, and the n^{th} objective functions of a multi-objective optimization problem. g and h are different types of constraint functions. A MOP can be defined as:

$$\begin{aligned} min\ z &= F(\boldsymbol{x}) = \{f_1(\boldsymbol{x}),\ f_2(\boldsymbol{x}), \ldots f_n(\boldsymbol{x})\} \\ s.t.\ g_i(\boldsymbol{x}) &\leq 0 \\ h_j(\boldsymbol{x}) &= 0 \\ \boldsymbol{x} &= \{x_1, x_2, x_3, \ldots, x_m\} \end{aligned} \tag{1}$$

In this representation, z is the set of the conflicting objective functions. $\boldsymbol{x}$ is a solution of the MOP, and it is constructed by $x_1, x_2, \ldots, x_m$. m is the dimension of the decision space. Therefore, m is equal to the length of $\boldsymbol{x}$.

Since there is not a global best solution in a MOP, we can only find out some Pareto solutions of it. If a solution $\boldsymbol{x}$ is not worse than the others, then the solution $\boldsymbol{x}$ can be identified as a Pareto solution. A Pareto solution is determined by a dominance relationship [3]. The dominance relationship can be described as follows: if there are a solution $\boldsymbol{x}$ and a solution $\boldsymbol{y}$ that satisfy:

$$\forall i \in \{1, \ldots, m\}\, f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y}) \wedge \exists f_i(\boldsymbol{x}) < f_i(\boldsymbol{y}) \tag{2}$$

then the solution $\boldsymbol{x}$ is said to dominates the solution $\boldsymbol{y}$, or the solution $\boldsymbol{y}$ is dominated by the solution $\boldsymbol{x}$. If there is not a solution $\boldsymbol{y}$ that can dominates the solution $\boldsymbol{x}$, then $\boldsymbol{x}$ is a Pareto solution. Equation 2 is a strong dominance and it can be represented by $F(\boldsymbol{x}) \succ F(\boldsymbol{y})$. If the second half of equation 2 does not exist, it is a weak dominance that represented by $F(\boldsymbol{x}) \succcurlyeq F(\boldsymbol{y})$. A solution $\boldsymbol{x}$ is Pareto optimal [3] if and only if:

$$\neg\exists F(\boldsymbol{y}) \succ F(\boldsymbol{x}),\ \boldsymbol{x}, \boldsymbol{y} \in \Omega \tag{3}$$

Ω is the decision space of a MOP. If there is not a solution $\boldsymbol{y}$ in the decision space Ω that $\boldsymbol{y}$ dominates $\boldsymbol{x}$, $\boldsymbol{x}$ is a Pareto optimal, or a Pareto solution. The collection of the Pareto solutions is a Pareto set (PS), which can construct a Pareto front (PF) in the objective space. The Pareto solution and the Pareto front is defined as [3]:

$$\boldsymbol{PS} = \{\boldsymbol{x} | \neg\exists F(\boldsymbol{y}) \succ F(\boldsymbol{x})\}\ \boldsymbol{x}, \boldsymbol{y} \in \Omega \tag{4}$$

$$\boldsymbol{PF} = \{F(\boldsymbol{x}) = f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\}\ \boldsymbol{x} \in \boldsymbol{PS} \tag{5}$$

The structure of Pareto front in the objective space can be spots, lines, planes or other complex structures, determined by the objective functions, constraint functions and the decision space of the corresponding MOP.

IV. MULTI-OBJECTIVE INVASIVE TUMOR GROWTH OPTIMIZATION

A. INVASIVE TUMOR GROWTH OPTIMIZATION

Invasive tumor growth optimization (ITGO) is a recently proposed algorithm of swarm intelligence [4]. It imitates the behaviours of tumor cells to construct a tumor-cell model to solve single-objective optimization problems. This kind of tumor-cell model divides the population into four types: invasive cells, growing cells, dormant cells and dead cells. The concentration of the nutrients around the tumor cells refer to their corresponding fitness value. All of these tumor cells will approach a higher (or lower, determined by the fitness function) concentration of the nutrients. They include:

1) Invasive cells, which are located at the outermost layer of the whole population. They can jump out of the local optima by Levy flight. Due to the effectiveness of invasive cells, they can also guide part of the movement of growing cells. The invasion of invasive cells is represented as:

$$Icell_i(t+1) = Icell_i(t) + \alpha \cdot Levy(s) \tag{6}$$

$$\alpha = rand \cdot (\frac{t}{T}) \tag{7}$$

$$Levy(s) \sim |s|^{(-1-v)}, (1 < v \leq 2) \tag{8}$$

$$step = \frac{u}{|v|^{1/\omega}} \tag{9}$$

$$u \sim N(0, \sigma_u^2), v \sim N(0, \sigma_v^2) \tag{10}$$

$$\sigma_v = \{\frac{\Gamma(1+\omega sin(\pi\omega/2)}{\Gamma[(1+\omega/2]\omega 2^{(\omega-1)/2)}}\}^{1/\omega} \tag{11}$$

$$\sigma_u = 1 \tag{12}$$

Icell$_i$ represents the i^{th} invasive cell. *Levy(s)* is the function for Levy flight. t is the current number of the iterations and T is the total number of iterations. s is the step size of invasive cells. u and v are used to calculate the step size. α is a control parameter for the step size. ω is a constant and in paper [4] the value of ω is 1.5. It is the value found in Paper [4] that makes Levy flight have the best search results.

2) Growing cells, which are located at the second layer of the whole population. They are guided by invasive cells and their own historical trajectory. The outermost portion of growing cells can be transformed into invasive cells and the innermost portion of growing cells can be transformed into dormant cells. They are the largest component of the entire population, taking on major search tasks. More details of growing cells are discussed in paper [4].

3) Dormant cells, which are located at the third layer of the whole population and are transformed from growing cells. Their step size is smaller than that of growing cells and invasive cells. A dormant cell can return to being a growing cell if it encounters a better concentration of nutrients. However, if the concentration of the surrounding nutrients worsens, it will be transformed into a dead cell. More details of dormant cells are discussed in paper [4].

4) Dead cells, which are located at the innermost layer of the whole population and are transformed from dormant cells due to insufficient searching. They will not perform searching anymore. At the end of an iteration, dead cells will release their computational resources and produce new invasive cells. More details of dead cells are discussed in paper [4].

The process of ITGO can be described as follows: 1) initialize the population; 2) the four types of cells search step-by-step according to the predefined rules; 3) when becoming trapped in a local optima, invasive cells will escape by Levy flight; and 4) at the end of an iteration, dead cells release their computational resources, and new invasive cells can be reproduced. During the whole movement, each type of cell performs its own search strategy until the whole population attains the global optimal solution. Figure 1 is the distribution of individuals in ITGO [4].

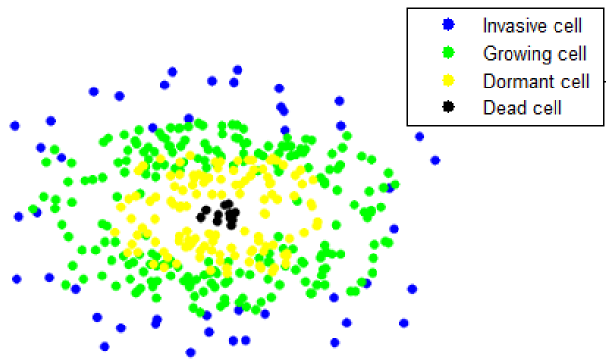


FIGURE 1. Cell distribution in ITGO.

B. MULTI-OBJECTIVE INVASIVE TUMOR GROWTH OPTIMIZATION

In this paper, we proposed a vascular invasive tumor growth optimization algorithm VITGO, which imitates the behaviours of vascular units and tumor cells to search for a significant Pareto front of a MOP. The newly proposed algorithm includes five types of individuals: vascular units, which locate on the current approximate Pareto front during the iteration; four types of tumor cells; which are inherited from the ITGO. The search schemes of four types of tumor cells are different from the original ones. An individual in VITGO contains both a position in the decision space and a corresponding position in the objective space. The former

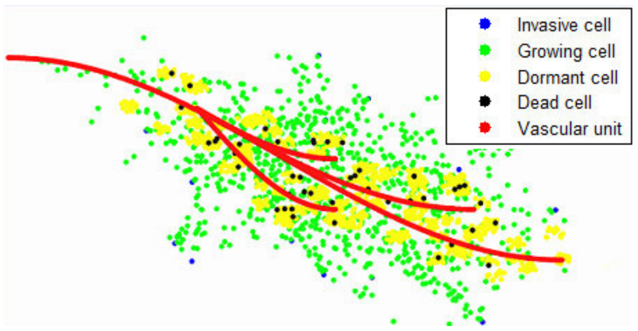
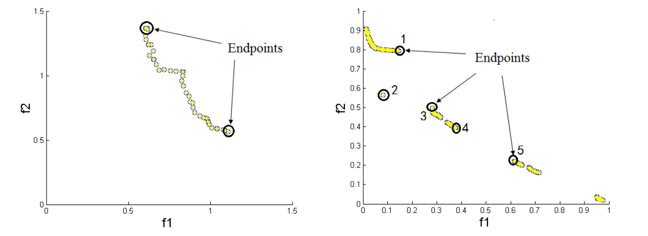


FIGURE 2. **Distribution of vascular units and tumor cells in VITGO.**



(a) Endpoints of one part of the approximate Pareto front

(b) Endpoints of several parts of the approximate Pareto front

FIGURE 3. **Endpoints of an approximate Pareto front.**

position is the center of an individual, and the latter position is its fitness. Figure 2 demonstrates five types of individuals in the objective space. The red dendritic organization is identified to be the vascular tissue, which is constructed by a lot of vascular units. It is inserted into the whole tumor cell population and partially guides the movement of tumor cells. At the end of an iteration, the vascular tissue will update to locate on the current approximate Pareto front. The outermost blue dots are identified to be invasive cells, which approach to the unsearched areas. The green dots located between the invasive cells and the dendritic organization are identified to be growing cells, which guarantee the whole solution space to be completely searched. The individuals that collect together (yellow dots) are identified to be dormant cells, which are responsible for the farthest search. The black dots located in the center of dormant cells' clusters are identified bo be dead cells, which are transformed from dormant cells due to a terrible fitness value. Totally speaking, the identification of different types of individuals is based on their positions in the objective space, and the generation and movement of an individual is applied in the decision space. At the end of each iteration, these structures and distributions will be reconstructed by some predefined rules, pushing the current approximate Pareto front to become closer to the true Pareto front.

1) VASCULAR UNIT GENERATION

Before introducing the generation of vascular units, some preparations should first be introduced. During the processing of a MOP, there may appear some incomplete or discontinuous structures in the approximate Pareto front, as showed in Figure 3. Figure 3a is a continuous but incomplete approximate Pareto front; Figure 3b is a discontinuous and incomplete approximate Pareto front.

In the incomplete parts or the discontinuous parts of the approximate Pareto front, there may appear several endpoints (circled in black in Figure 3). In total, there are two types of endpoints: 1) endpoints without gaps (continuous but incomplete), as shown in Figure 3a, and 2) endpoints with gaps (discontinuous and incomplete), as shown in Figure 3b. To obtain a better approximate Pareto front that can be closer to the true PF, these kinds of endpoints need to be well used. If we can generate some new individuals around the endpoints in Figure 3a, we may obtain a more complete approximate Pareto front and can achieve better convergence. If we can generate some new individuals in the gaps of endpoints in Figure 3b, we may obtain a more complete approximate Pareto front. As above, an endpoint is identified according to its position in the objective space, and tries to induce generating new individuals in the decision space. Since the monotonicity of the objective functions cannot be guaranteed useful, the position of the newly generated individuals cannot be guaranteed to be useful in the objective space. Therefore, only an individual should be generated in the gaps at a time:

$$vunit_{new} = 0.5 \cdot p_1 \cdot (endpoint_i + endpoint_{i+1}), \quad i \in [1, n_{gaps}] \tag{13}$$

In equation 13, $vunit_{new}$ is a newly generated vascular unit. $endpoint_i$ and $endpoint_{i+1}$ are two endpoints with a gap. n_{gaps} is the total number of the endpoints with gaps. p_1 is a perturbation parameter to prevent redundant computation. The reason is that two endpoints may be repeatedly identified in different generations. In order to avoid generating a new individual in the same position, a small disturbance needs to be added. If the vascular unit $vunit_{new}$ is generated in the gaps, it can guide other individuals to move farther in the next generation. If not, the time lost encountered by this step can be ignored since there is only one individual produced; moreover, it is the first way to generate new individuals with endpoints, which is named ''gaps filling'' here.

To precisely select the endpoints in Figure 3b to finish the ''gaps filling'' scheme, we need a sorting scheme based on the difference of each objective function:

$$\begin{aligned} vesselsort^j &= qsort^j(vunit), j \in [1, N] && (14)\\ \Delta vesselsort_i^j &= vesselsort_{i+1}^j - vesselsort_i^j, \\ &\qquad i \in [1, n_v - 1], j \in [1, N] && (15)\\ endpoint &= \{vunit_{i,i+1} | \Delta vesselsort_i^j > T_{gaps}\} && (16) \end{aligned}$$

In equation 14, $vunit$ is the set of vascular units. $qsort^j$ is a Quicksort, which sorts the $vunit$ by their j^{th}-objective value. $vesselsort^j$ is the result after Quicksort $qsort_j$. N is the total number of objective functions. It is employed to sort the vascular units according to their j^{th}-objective values and finally obtain N arrays. In equation 15, $vesselsort_i^j$ is the i^{th} element in the array $vesselsort_j$. $\Delta vesselsort_i^j$ is the difference between the i^{th} vascular unit and the $(i+1)^{th}$ vascular unit in the array $vesselsort^j$. n_v is the number of vascular units. This step is to calculate the differences of sorted arrays $vesselsort^j$. In equation 16, *endpoint* is the endpoints with gaps that identified by a threshold T_{gaps}. $vunit_{i,i+1}$ is the i^{th} and the $(i+1)^{th}$ vascular units on both sides of $\Delta vesselsort_i^j$. If $\Delta vesselsort_i^j$ is grater than the predefined threshold T_{gaps}, it is identified as a gap, and the vascular units on both sides of the gaps will be added into the endpoints with gaps. The value of T_{gaps} is set in Section 5. By utilizing these formulas, the difference of each objective function will be calculated several times, and then the endpoints with larger gaps can be counted repeatedly to generate more. Moreover, equations 13-16 are independent of the space dimension, and then they can still be implemented with N-objective problems without any extra settings to determine the corresponding (N-1)-dimensional structure. After sorting and selection, endpoint 1 and endpoint 3 in Figure 3b can be first selected, and then endpoint 4 and endpoint 5, and so on. The reason for sorting the vascular units by their j^{th}-objective values instead of determining their neighbours is that the former can reduce computational redundancies. The way to select other endpoints will be discussed below. All of the endpoints can be utilized as the guidance of other tumor cells.

2) VASCULAR GUIDANCE

As mentioned above, the endpoints on the both sides of the approximate Pareto front are effective for searching since their search directions are wider and farther. They can also be utilized as a significant base for invasive cells to jump into the unsearched areas. We identified a vascular unit that is farthest from the center of vascular tissue that:

$$dis_i^c = \left| vunit_i - \frac{1}{n_v}\sum_{i=1}^{n_v} vunit_i, \right|, i \in [1, n_v] \quad (17)$$

$$endpoint_{new} = vunit_{idx(max(dis^c))} \quad (18)$$

In equation 17, $vunit_i$ is the i^{th} vascular unit. n_v is the number of vascular units. dis_i^c is the distance between the geometric center and $vunit_i$. In equation 18, $endpoint_{new}$ is the newly identified endpoint on one of the sides of the current approximate Pareto front. $idx(max(dis^c))$ is the index of the maximum dis^c.

Moreover, the individuals that far away from others should also be identified as endpoints. As shown in Fig 3, this kind of individuals can also be utilized to generate individuals to improve the convergence. All of them should be utilized as the guiders:

$$dis_i^v = min_{value|2}(|vunit_i - vunit|), i \in [1, n_v - 1] \quad (19)$$

$$endpoint_{new} = \{vunit_i | dis_i^v > T_{gaps}\} \quad (20)$$

$$guider = endpoint \quad (21)$$

In equation 17, dis_i^v is the Euclidean distance between the i^{th} vascular unit and $vunit$. $min_{value|2}$ (A) is a method to determine the second minimum elements in array A and return its value. Therefore, this step is to determine the individual that is closest to $vunit_i$ (the minimum distance is 0 of $vunit_i$ and itself). In equation 18, *endpoint* is the endpoints that far away from the vascular units. $vunit_i$ is the i^{th} vascular unit. T_{gaps} is a predefined threshold in Section 5. This step is to determine the individual that is farthest from the $vunit$. After applying these formulas, endpoint 2 in Figure 3b can be selected first since it is the farthest from the vascular units. These *endpoint* will be utilized as the *guider* of other tumor cells. More details will be discussed in the next subsections.

3) VASCULAR UNIT UPDATING

After the individuals' generation and merging, there may be some old and invalid vascular units that are located outside of the current approximate Pareto front. Therefore, they should be deleted by a dominance relationship to retain efficiency. The vascular updating approach to remove these invalid vascular units is defined as:

$$vunit_j \leftarrow \varnothing \ if\ \exists\ vunit_i \succcurlyeq vunit_j \quad (22)$$

$vunit_i$ and $vunit_j$ are two different vascular units such that $vunit_i$ dominates $vunit_j$. In this situation, $vunit_j$ should be deleted immediately. After this operation, the old and invalid vascular units could be removed to maintain a more effective approximate Pareto front for the next generation.

4) VASCULAR UNIT PRUNING

In the proposed algorithm, the size of the whole population is not fixed. Although invalid individuals can be deleted by dominance above, the size of the whole population is still inevitably increasing and lead to a massive density. Moreover, similar individuals that have similar search directions will cause excessive redundant calculations. Therefore, a parameter *delta* is introduced to construct a vascular units pruning scheme that trims out similar vascular units to indirectly control the size of the whole population and avoid a massive population density. The counterintuitive operation of the pruning scheme is that it is utilized to trim out similar valid individuals (similar to other individuals) instead of invalid individuals:

$$vunit_i \leftarrow \varnothing, if\ \left|vunit_i - vunit_j\right| < delta, \\ i \in [1, n_v], j \in [1, n_v] \quad (23)$$

In equation 23, $vunit_i$ and $vunit_j$ are both vascular units. n_v is the total number of vascular units. *delta* is the threshold that triggers the pruning scheme. If the number of the population increases, it means that the density of individuals is also increasing to a certain degree. Then parameter *delta* can play its role to make the VITGO algorithm maintain its effectiveness. The value of *delta* will be tested in section 5.

5) BOUNDARY DETECTION AND RANDOM WALK WITH CORRESPONDING DIMENSION

In the proposed VITGO algorithm, a partially random walk scheme was utilized to enhance the performance. If an individual touches the boundary, it will randomly move in the directions that exceed the boundary until other directions stay unchanged. Mathematically,

$$vunit_{i,m} = rand \cdot (ub_m - lb_m) + lb_m, m \in [1, M_{exceed}] \tag{24}$$

In equation 24, *rand* is a uniformly distributed random number. $vunit_i$ is the i^{th} vascular unit. *exceed* is the subscripts of the directions that exceed the boundary. *M* is the dimension of the decision space. *lb* is the lower boundary and *ub* is the upper boundary. Specifically, if an individual exceeds the upper boundary or the lower boundary, only the variables that exceed the boundary need a random walk while other variables unchanged. Comparing with the fully random walk, it is more stable.

The above methods constitute the entire vascular mechanism, which is the core of the whole VITGO algorithm. To better cooperate with the proposed vascular mechanism, we redesigned the growth, metastasis, and transformation rules of four types of tumor cells.

6) THE OPPOSITE SEARCH OF INVASIVE CELLS

The role of invasive cells is to identify new regions that have not been searched. Similar to ITGO, the invasive cells also apply the Levy flight to get out of the current location. However, different from the ITGO, invasive cells in the VITGO will utilized a second opposite step to identify a more effective search direction:

$$\begin{aligned} icell_{new} = Dominance\{&guider_{randj} \cdot (step + 1), \\ &guider_{randj} \cdot (-step + 1)\}, randj \in [1, n_e] \end{aligned} \tag{25}$$

The parameter *step* in equation 25 is the just the same as that in equation 9. As listed above, some randomly chosen guiding individuals $guider_{randj}$ will guide the generation and movement of invasive cells. Since the endpoints are located exactly in the approximate Pareto front, it will be the best guidance for invasion. To be more effective, for each endpoint, there are two new invasive cells generated by the *step* and the opposite $-step$, and the nondominated one will be retained for the next iteration. In general, one of these two opposite step sizes determines a better search direction than the other. If their performances are equal to each other, only one of them can remain. If the Levy flight is not satisfactory, a completely random walk will be utilized by invasive cells to search for new regions.

7) THE WIDER SEARCH OF GROWING CELLS

As listed in Figure 3a, the endpoints (guiders) located at the both sides of the approximate Pareto front will be more effective than others to search wider. Since these kinds of endpoints have become the guiders of tumor cells, the growing cells will be generated by:

$$gcell_{new} = p_2 \cdot guider_{randj}, randj \in [1, n_e] \tag{26}$$

$$guider_{new} = gcell_{new}, \; if \; gcell_{new} \succ guider_{randj} \tag{27}$$

In equation 26, $gcell_{new}$ is a newly produced growing cell. $guider_{randj}$ is a randomly chosen guidance individual. p_2 is a parameter to control the location of the newly produced growing cells. If $gcell_{new}$ dominates $guider_{randj}$, it will be added into the *guider* to guide the movement of dormant cells. After applying formulas above, the newly generated growing cell is actually located around its guider in the decision space. If the monotonic of the objective functions is smooth, the newly generated growing cell can actually locate around its guider in the objective space. If not, it will locate on some other position in the objective space according to the objective functions. In other words, this scheme will succeed in probability.

8) THE FARTHER SEARCH OF DORMANT CELLS

To make the population search farthest in each direction, the movement of dormant cells is designed as follows: a dormant cell only moves in 1 dimension of the decision space at a time. From a numerical point of view, only 1 element in the step-size vector of a dormant cell can be positive or negative, while other elements contain 0. In a real biological environment, the movement of a dormant cell is very small due to the lack of nutrients. The dormant cells in VITGO follow this behavior to make themselves work better. This scheme can help a dormant cell to search father in each direction and stay stable in other directions. To better solve the multi-objective optimization problems and cooperate with vascular tissues, dormant cells will be guided by vascular units instead of growing cells:

$$dorcell_{new} = p_3 \cdot guider_{randj}, randj \in [1, n_e], |p_3| \in (\frac{1}{t}, 1) \tag{28}$$

$$\begin{aligned} dorcell_{new,m} = p_4 \cdot dorcell_{new,m}, m \in [1, M], \\ |p_4| \in (1 - \frac{1}{t}, 1) \end{aligned} \tag{29}$$

In equation 28, $dorcell_{new}$ is a newly generated dormant cell. *guider* is the guiding individual. Obviously the generation and movement of dormant cells will be guided by both the endpoints and growing cells with high quality. The size of dormant cells is equal to n_e. p_3 is a uniformly distributed random number which has an absolute value between $1/t$ and 1. t is the t^{th} iteration. In equation 29, $dorcell_{new}$ is a newly generated dormant cell. *M* is the dimension of the decision space, and *m* is the m^{th} dimension of the decision space. p_4 is a uniformly distributed random number that has an absolute value between 1-1/t and 1.

Equation 28 indicates that the dormant cell is guided by a randomly selected guider and generates nearby. The absolute value of p_3 decreases as the number of iterations increases to ensure that when the dormant cells become closer to the true Pareto front, the algorithm performs a more refined search. The absolute value of p_4 increases as the parameter p_3 decreases to ensure the activity of the dormant cells to avoid redundant computation.

Equation 29 indicates that the dormant cells perform a farther search in each dimension of the decision space to ensure that no feasible search direction is missed. They can also ensure the search accuracy when the approximate Pareto front is very close to the true Pareto front. As shown in Figure 2, the newly formed dormant cells always generate a cell mass due to their small step size.

9) RULE FOR RELEASING RESOURCES

To reduce the waste of computational resources, once a newly produced dormant cell or its guider cannot perform well, it immediately turns into a dead cell and releases its resources. In contrast to the original ITGO, invasive cells cannot be produced here.

$$deacell_{new} = \neg Dominance\{dorcell_{new1}, dorcell_{new2}\} \tag{30}$$

$$deacell \leftarrow \varnothing \tag{31}$$

In equation 30, $dorcell_{new1}$ and $dorcell_{new2}$ are both newly generated dormant cells in equation 28-29. *Dominance* refers to the dominance relationship. If the newly generated dormant cell is dominated by others, it will be immediately transferred to a dead cell because its search direction is invalid. No more action is taken here without releasing the computational resources. As shown in Figure.2, some dead cells appear in the center of the dormant cell mass (the invalid search direction). They will be deleted immediately. The purpose of this step is to eliminate the invalid individuals before entering the vascular renewal process to avoid invalid calculations.

Table 1 lists the differences between VITGO and ITGO.

TABLE 1. Difference between ITGO and VITGO.

structure	ITGO	VITGO
invasive cell	single step	opposite steps
invasive cell	invasion from $gcell$	invasion from $vunit$
growing cell	guided by $icell$	guided by $vunit$
dormant cell	search in all dimensions	search in 1 dimension
dead cell	release at the end of iteration	release immediately
dead cell	remind to produce $icell$	-
vascular unit	-	record the Pareto front
vascular unit	-	defined the search direction
vascular unit	-	generate end points
vascular unit	-	partially guide other cells

C. FLOWCHART AND PSEUDOCODE

The flowchart of VITGO is shown in Fig.4. The parameter *cPareto* refers to the approximate Pareto front in the corresponding iteration. After the initialization, the vascular units, invasive cells, growing cells, dormant cells (and dead cells) generate according to corresponding rules; then, these cells/units merge together for pruning and domination until the termination condition is reached.

The pseudocode lists the same steps.

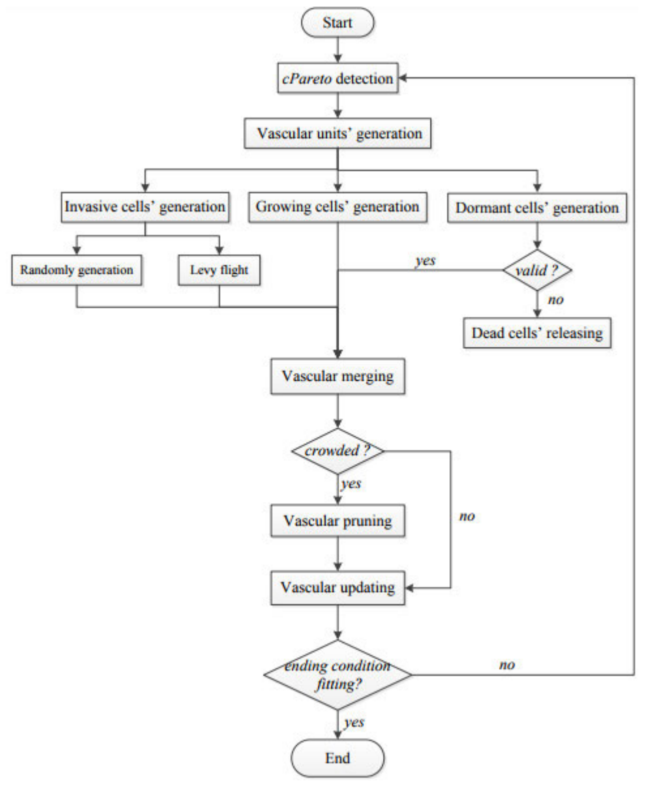


FIGURE 4. Flowchart of multi-objective vascular invasive tumor growth optimization.

D. TIME COMPLEXITY

According to the processes mentioned above, the time complexity of VTIGO can be analyzed separately. Let T denote the number of iterations, N denote the number of objective functions, n denote the size of the population, M denotes the number of dimension of the decision space (in section 4), n_v, n_i, n_g, n_d refer to the size of vascular units, invasive cells, growing cells, and dormant cells, respectively. According to the predefined formulas, we can concluded that: $n = n_v + n_i + n_g + n_d$, $n_i = n_g = n_e \approx n_d$, and n_e is determined by $vunit$. Therefore, the time complexity of VITGO can be divided as: 1) the growth of vascular units that requires $Nn_v logn_v$ computation for each iteration; and the time complexity for this part is O($TNn_v logn_v$); 2) the update of vascular units that requires n_v^2 for domination; and the time complexity for this part is O(Tn_v^2); 3) the pruning process for blood vessels that requires n_v computation for each iteration; and the time complexity for this part is O(Tn_v); 4) boundary detection that requires a constant computation and the time complexity in this part is also a constant; 5) the production and search process of invasive cells that requires n_i computation for each iteration; and the time complexity for this part is O(Tn_i); 6) the production and search process of growing cells that requires n_g computation for each iteration; and the time complexity for this part is

Algorithm 1 Vascular Invasive Tumor Growth Optimization Algorithm for Multi-Objective Optimization

```
Input: $problemIndex$
Output: $cPareto$
1  Initialization();
2  while t $< T$ do
3      call Vascular unit' generation;
4      call Invasive cell' generation;
5      call Growing cell' generation;
6      call Dormant cell' generation;
7      call Random generation;
8      $vunit \leftarrow vunit \cup icell \cup gcell \cup dorcell\ tmpicell$;
9      $i \leftarrow 1$;
10     while $i < n_v$ do
11         $dis \leftarrow |vunit_i - vunit|$;
12         if $min_{value|2}(dis) < delta$ then
13             $vunit_i \leftarrow \varnothing$;
14             $n_v \leftarrow n_v - 1$;
15         end
16     end
17     foreach $i,j$ in $vunit$ do
18         if $vunit_i \succcurlyeq vunit_j$ then
19             $vunit_j \leftarrow \varnothing$;
20         end
21     end
22     $t \leftarrow t + 1$;
23 end
24 $cPareto \leftarrow vunit$;
```

O(Tn_g); 7) the production and searching of dormant cells that requires n_d computation for each iteration; and the time complexity for this part is O(Tn_d). Since n_v, n_i, n_g, and n_d are part of the population size n, the total time complexity of VITGO is:

$$\begin{aligned} time\ complexity &= O(TNn_v logn_v + Tn_v^2 + Tn_v \\ &\quad + Tn_i + Tn_g + TMn_{dor}) \\ &= O(T(M + Nn)n) \end{aligned} \tag{32}$$

The time complexity of VITGO was similar to the time complexity of NSGAII, which is generally influenced by its fast non-dominated sorting algorithm, and requires O(Nn^2) time complexity in each iteration [16]. Therefore, the time complexity of the NSGAII algorithm is O(TNn^2). The only difference between VITGO and NSGAII is that the VITGO considers the influence of the spatial dimension M while the NSGAII does not.

Algorithm 2 Vascular Unit' Generation

```
Input: $vunit$, $Tgaps$, $p_1$
Output: $vunit$, $guider$
1  $endpoint \leftarrow \varnothing$;
2  $guider \leftarrow \varnothing$;
3  foreach $j$ in $N$ do
4      $vesselsort^j \leftarrow qsort^j(vunit)$;
5      foreach $i$ in $vunit$ do
6          $\Delta$vesselsort$_i^j \leftarrow$ vesselsort$_{i+1}^j$ - vesselsort$_i^j$;
7          if $\Delta vesselsort_i^j > Tgaps$ then
8              $vunit_{new} \leftarrow 0.5 \cdot p_1 \cdot (vesselsort_{i+1}^j +$
               $vesselsort_i^j)$;
9              $vunit \leftarrow vunit \cup vunit_{new}$;
10             $endpoint \leftarrow endpoint \cup vesselsort_i^j \cup$
               $vesselsort_{i+1}^j$;
11         end
12     end
13 end
14 $v_{center} \leftarrow \frac{1}{n_v}\sum_{i=1}^{n_v} vunit_i$;
15 foreach $i$ in $vunit$ do
16     $dis_i^c \leftarrow |vunit_i - v_{center}|$;
17 end
18 $idxMax \leftarrow Idxof(max(dis_i^c))$;
19 $endpoint \leftarrow endpoint \cup vunit_{idxMax}$;
20 foreach $i$ in $vunit$ do
21     $dis_i^v \leftarrow |vunit_i - vunit|$;
22     if $min_{value|2}(dis_i^v) > Tgaps$ then
23         $endpoint \leftarrow endpoint \cup vunit_i$;
24     end
25 end
26 $guider \leftarrow endpoint$;
```

Algorithm 3 Invasive Cell' Generation

```
Input: $guider$, $n_e$, $n_i$
Output: $icell$
1  $icell \leftarrow \varnothing$;
2  $i \leftarrow 0$;
3  while $i < n_i$ do
4      $randj \leftarrow ceil(n_e \cdot rand)$;
5      $step \leftarrow \alpha \cdot Levy$;
6      $icell_{new1} \leftarrow guider_{randj} \cdot (step+1)$;
7      $icell_{new2} \leftarrow guider_{randj} \cdot (-step+1)$;
8      if $icell_{new1} \succcurlyeq icell_{new2}$ then
9          $icell \leftarrow icell \cup icell_{new1}$;
10     else
11         $icell \leftarrow icell \cup icell_{new2}$;
12     end
13     $i \leftarrow i + 1$;
14 end
```

V. EXPERIMENTS

A. TEST PROBLEMS, COMPARING ALGORITHMS, AND RUNNING ENVIRONMENT

There are many multi-objective optimization problems that have been proposed for testing in recent years. The most generally utilized test problems are ZDTs, UFs, DTLZs and WFGs. They were constructed with different principles and contain different difficulties for an algorithm to address. The test set ZDT has a total of six problems; each of them

Algorithm 4 Growing Cell' Generation

```
Input: $guider, n_e, n_g, p_2$
Output: $guider, gcell$
1  $gcell \leftarrow \varnothing$;
2  $i \leftarrow 0$;
3  while $i < n_g$ do
4      $randj \leftarrow ceil(n_e \cdot rand)$;
5      $gcell_{new} \leftarrow p_2 \cdot guider_{randj}$;
6      $gcell \leftarrow gcell \cup gcell_{new}$;
7      if $gcell_{new} \nsucceq guider_{randj}$ then
8          $guider \leftarrow guider \cup gcell_{new}$;
9      end
10     $i \leftarrow i + 1$;
11 end
```

Algorithm 5 Dormant Cell' Generation

```
Input: $guider, n_e, n_d, p_3, p_4$
Output: $dorcell$
1  $dorcell \leftarrow \varnothing$;
2  $deacell \leftarrow \varnothing$;
3  $i \leftarrow 0$;
4  while $i < n_d$ do
5      $randj \leftarrow ceil(n_e \cdot rand)$;
6      $dorcell_{new1} \leftarrow p_3 \cdot guider_{randj}$;
7      if $guider_{randj} \nsucceq dorcell_{new}$ then
8          $deacell \leftarrow deacell \cup dorcell_{new1}$;
9      else
10         foreach $m$ dimension in $M$ do
11             $dorcell_{new2} \leftarrow dorcell_{new1}$;
12             $dorcell_{new2,m} \leftarrow p_4 \cdot dorcell_{new1,m}$;
13             if $dorcell_{new1} \nsucceq dorcell_{new2}$ then
14                 $deacell \leftarrow deacell \cup dorcell_{new2}$;
15             else
16                 $deacell \leftarrow deacell \cup dorcell_{new1}$;
17                 $dorcell_{new1} \leftarrow dorcell_{new2}$;
18             end
19         end
20     end
21     $deacell \leftarrow \varnothing$;
22     $i \leftarrow i + 1$;
23 end
```

Algorithm 6 Random Generation

```
Input: $n_{rand}$
Output: $tmpicells$
1  $tmpicells \leftarrow \varnothing$;
2  $i \leftarrow 0$;
3  while $i < n_{rand}$ do
4      $icell_{tmp1} \leftarrow rand \cdot boundary$;
5      $tmpicells \leftarrow tmpicells \cup icell_{tmp1}$;
6      $i \leftarrow i + 1$;
7  end
```

involves a difficulty that can be evaluated, and an algorithm may encounter these difficulties during evolutions [39]. Each test problem in ZDT set has two objective functions. Since ZDT5 of this test set is rarely used, in the experiment, we only use the other five multi-objective optimization problems for testing. The test set DTLZ is totally different from the test set ZDT in that the number of its objective functions is extensible and can be arbitrarily specified [40]. The most frequently utilized number of the objective functions in DTLZs is three, and the most frequently utilized test problems are DTLZ1-DTLZ4. The test set UF is also totally different from that of the ZDT and DTLZ in that it is constructed by a series of unconstrained problems and a series of generally constrained problems [41]. There are ten test problems in total in the UF set: UF1-UF10. UF1-UF7 are the two-objective optimization problems, and UF8-UF10 are the three-objective optimization problems. All ten of these problems will be utilized in the experiment. The test set WFG includes nine test problems: WFG1-WFG9. Similar to DTLZ, the number of objective functions in WFG is extensible, and the number of objective functions is set to three in all the experiments. The parameter k of these problems is set to 2, and and the parameter l is set to 2. Other parameters are the same as which suggested in the corresponding paper. There are in total 28 test problems in the experiment: ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, UF1, UF2, UF3, UF4, UF5, UF6, UF7, UF8, UF9, UF10, DTLZ1, DTLZ2, DTLZ3, DTLZ4, WFG1, WFG2, WFG3, WFG4, WFG5, WFG6, WFG7, WFG8, and WFG9. The first 12 problems are two-objective problems, and the others are three-objective problems.

In the experiment, we will utilize five multi-objective optimization algorithms to compare with VITGO: NSGA-III [17], MODA [30], HEIA [27], MOEA/D-AGR [36] and MOEA/D-CDE [35]. These algorithms have been briefly introduced in Section 2. The running environment of these algorithms is R2014a. Each algorithm will be tested with the 28 test problems with 30 runs, and the results will be recorded in the following tables.

B. PERFORMANCE METRICS

To evaluate the performance of multi-objective algorithms in different aspects, we choose four performance metrics in total: generational distance (GD) [42], inverse generational distance (IGD) [43], Spread (SP) [44] and hypervolume (HV) [45]. Generational distance (GD) is to calculate the root of the sum of the squares of the minimum distances from each individuals of the approximate Pareto front to its nearest neighbor of the true Pareto front [42]. It takes a similar idea to the Euclidean distance and can therefore be utilized to estimate how far it is from the approximate Pareto front to the true Pareto front.

$$GD = \frac{\sqrt{\sum_{i=1}^{|\boldsymbol{cR}|} distance(\boldsymbol{cr},\boldsymbol{p})}}{|\boldsymbol{cR}|} \tag{33}$$

Here ***cR*** is the approximate Pareto front (current results of a multi-objective optimization algorithm). $|\boldsymbol{cR}|$ is the size of the approximate Pareto front. ***cr*** is an individual on the approximate Pareto front ***cR***. ***p*** is an individual on the true Pareto front ***PF*** and it is the nearest individual of ***cr***. The function *distance*() is to calculate the Euclidean distance. This formula is to calculate the Euclidean distance of each individual ***cr*** in ***cR*** and its nearest neighbor ***p*** in ***PF*** to estimate how far it is from the approximate Pareto front ***cR*** to the true Pareto front ***PF***. It can identify the convergence of the approximate Pareto front ***cR*** and the performance of the corresponding multi-objective optimization algorithm.

Inverse generational distance (IGD) is to calculate the minimum distance from each individual of the true Pareto front ***PF*** to its nearest neighbor in the approximate Pareto front ***cR***. It takes a similar idea to GD but an inverse technique to both identify the convergence and diversity of a multi-objective algorithm.

$$IGD = \frac{\sum_{i=1}^{|\boldsymbol{PF}|} distance(\boldsymbol{p},\boldsymbol{cr})}{|\boldsymbol{PF}|} \tag{34}$$

Here, ***PF*** is the true Pareto front of a multi-objective optimization problem. $|\boldsymbol{PF}|$ is the size of the true Pareto front. ***p*** is an individual on the true Pareto front ***PF***. ***cr*** is a point on the approximate Pareto front ***cR*** and it is the nearest individual from ***p***. The function *distance*() is to calculate the Euclidean distance. This formula is to calculate the (average) Euclidean distance between an individual ***p*** in ***PF*** and its nearest neighbor ***cr*** in ***cR*** to estimate how far it is from the true Pareto front ***PF*** to the approximate Pareto front ***cR***. Therefore, it can both identify the convergence and diversity of the approximate Pareto front ***cR*** and the performance of the corresponding multi-objective optimization algorithm.

Spread (SP) is a formula to calculate the square root of the approximate average of the square of mean difference of the Manhattan distance between each individual of the approximate Pareto front and its nearest neighbor. It takes a similar idea to the standard deviation and can therefore be utilized to estimate the identify the density and diversity of the approximate Pareto front.

$$SP = \sqrt{\frac{1}{|\boldsymbol{cR}|-1}\sum_{i=1}^{|\boldsymbol{cR}|}(\bar{d}-d_i)^2} \tag{35}$$

$$d_i = min_j \sum_{m=1}^{M}(|f_{i,m} - f_{j,m}|),\ i, j \in \boldsymbol{cR} \tag{36}$$

Here ***cR*** is the approximate Pareto front. $|\boldsymbol{cR}|$ is the size of the approximate Pareto front. $f_{i,m}$ is the m^{th} fitness value of the i^{th} individual in ***cR***. d_i is the Manhattan distance of the i^{th} individual in ***cR*** and its nearest neighbor j. $\bar{d}$ is the mean value of d_i. This formula can identify the degree of dispersion of the approximate Pareto front and be utilized to estimate the identify the density and diversity to a certain extent.

Hypervolume (HV) is to calculate the hypervolume of each individual of the approximate Pareto front, which can be utilized to identify the dominance.

$$HV(\boldsymbol{cR}, \boldsymbol{r}) = volume(\bigcup_{f \in \boldsymbol{cR}} [f_1, r_1] \times ...[f_M, r_M]) \tag{37}$$

Here ***cR*** is the approximate Pareto front. ***r*** is the predefined reference individual. f_i is the i^{th} objective value of an individual. This formula is to test the dominance of the approximate Pareto front to the reference ***r***. In the experiments, we choose ***r*** to be $1.1 * max(\boldsymbol{PF})$ for each test problems. For example, the reference individual ***r*** for ZDT1 is [1.1 1.1]. The parameter N for HV is set to 100000.

C. PARAMETER SETTINGS

To better analyze the proposed VITGO algorithm, we should firstly identify the influence of each parameter in VITGO and select proper values for them. The test problem for the parameter selection is UF1 since it is not easy to achieve the true Pareto front. Therefore, the nondominated solutions can be more discriminating for different values of parameters and then the effect of each parameter can be clearly showed.

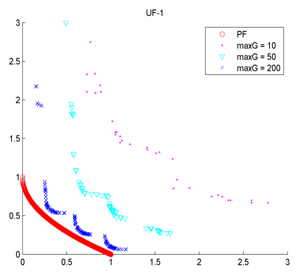


FIGURE 5. MaxGTest. Tgaps = 0.01, delta = 0.001, popsize = 20.

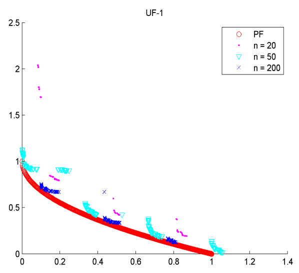


FIGURE 6. PopsizeTest. Tgaps = 0.01, delta = 0.001, maxG = 200.

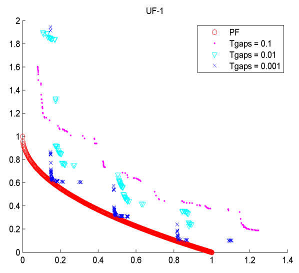


FIGURE 7. TgapsTest1. maxG = 200, delta = 0.001, popsize = 20.

Figure 5 - Figure 12 is the results of parameter testing. Figure 5 shows the influence of the parameter *maxG*, which is the maximum generation of the VITGO algorithm. To clearly showed the influence of the parameter *maxG*,

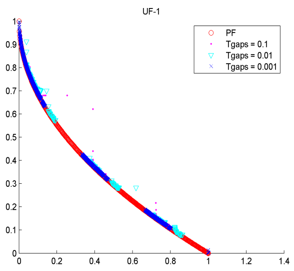


FIGURE 8. TgapsTest2. maxG = 1000, delta = 0.001, popsize = 20.

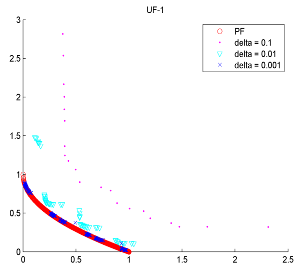


FIGURE 9. DeltaTest1. maxG = 200, Tgaps = 0.01, popsize = 20.

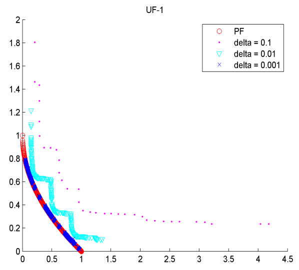


FIGURE 10. DeltaTest2. maxG = 1000, Tgaps = 0.01, popsize = 20.

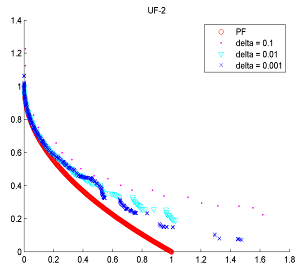


FIGURE 11. DeltaTest3. maxG = 200, Tgaps = 0.01, popsize = 20.

we have selected three small values for it: 10, 50, and 200. According to Figure 5, it can be concluded that as the number of iterations increases, the converge of the VITGO algorithm becomes better. Larger maximum generation will help for convergence. In the experiments the parameter *maxG* will be set to 1000.

Figure 6 shows the influence of the parameter n, which is the initial number of the whole population. The value of this parameter is set to 20, 50, and 200. The parameter *maxG* in Figure 6 was set to 200 to magnify the discrimination of the

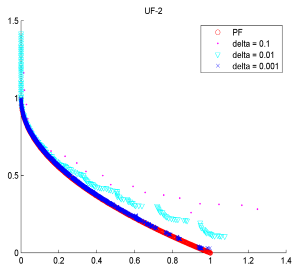


FIGURE 12. DeltaTest4. maxG = 1000, Tgaps = 0.01, popsize = 20.

effect of the parameter n. It can be concluded from Figure 6 that within a certain range, as the initial number of population increases, the convergence of the VITGO algorithm becomes better. However, after the initial number of the population increase to a certain value, the effect of this parameter will gradually weaken. It is because the size of the population of the VITGO algorithm is not fixed, instead it is indirectly controlled by the parameter δ. Therefore, in the experiment the parameter n will be set to 50.

Figure 7 and Figure 8 show the influence of the parameter *Tgaps*, which is a parameter to identify the endpoints on the approximate Pareto front. The value of this parameter is set to 0.1, 0.01, and 0.001. The parameter n was set to 20 to magnify the discrimination. In Figure 7, the parameter *maxG* was set to 200 and in Figure 8, the parameter *maxG* was set to 1000. It can be concluded from Figure 7 that as the value of *Tgaps* decreases, the convergence of the VITGO algorithm becomes better. However, as the value of *maxG* increases, the effect of this parameter will gradually weaken (see Figure 8). Therefore, in the experiment the parameter *Tgaps* will be set to 0.01.

Figure 9 - Figure 12 show the influence of the parameter δ, which is a parameter to trigger the pruning scheme and then indirectly control the size of the whole population. The value of this parameter is set to 0.1, 0.01, and 0.001. The parameter n was set to 20 to magnify the discrimination. In Figure 9, the parameter *maxG* was set to 200 and in Figure 10, the parameter *maxG* was set to 1000. It can be concluded from Figure 9 that as the value of δ decreases, the convergence of the VITGO algorithm becomes better. Figure 10 supports this conclusion. It can also be concluded from Figure 10 that as the value of δ decreases, the diversity of the VITGO algorithm becomes better. The reason is that δ is to indirectly control the density of the approximate Pareto front and the size of the whole population. Larger δ will results in smaller density of the approximate Pareto front and then the guidance of the vascular units will weaken. To comprehensively demonstrate the effect of the parameter δ, we utilized another test problem UF2 to identify the influence of this parameter. The parameter settings in UF2 were the same as those in UF1. Figure 11 - Figure 12 showed the results and they also support these conclusions. Therefore, in the experiment the parameter δ will be set to 0.001.

Other parameters was just the same as those in the original ITGO, as shown in Table 2. The letter "M" in Table 2 means

TABLE 2. Parameters for VITGO.

Parameters	Value/Value Range	Description
n	50	popsize
ω	1.5	levy flight
β	1	step-size
δ	0.001	pruning
$Tgaps$	0.01	endpoints
$maxG$	1000	max generation
p_1	$(-1,-0.95) \cup (0.95,1)$	perturbation(M)
p_2	$(-1,-0.9) \cup (0.9,1)$	perturbation(M)
p_3	$(-1,-\frac{1}{t}) \cup (\frac{1}{t},1)$	perturbation(M)
p_4	$(-1,-(1-\frac{1}{t})) \cup (1-\frac{1}{t},1)$	perturbation(1)

that the length of the parameter is M and the letter "1" means that the length of the parameter is 1. p_1, p_2, p_3 and p_4 are the perturbations of the VITGO algorithm. They should take some tiny values to avoid the array (element) to equal to another one. Moreover, the values of p_1, p_2, p_3 and p_4 is no fixed. They require a relatively small range of values to ensure that they assist the selected individual to search around. In this experiment we will take small values for each of them. As listed in Table 2.

Besides the VITGO algorithm, other five algorithms also apply the same values of n and $maxG$ as those in VITGO. The *nArchives* of NSGAIII and other algorithms is set to 1000 to maintain the diversity as much as possible. Moreover, in the last iteration they will not apply the deletion based on the *nArchives* for the same goal. Other parameters of these algorithms are just the same as the corresponding papers [17], [27], [30], [35], [36].

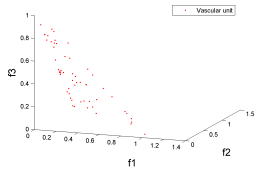


FIGURE 13. Part-1 vascular unit generation.

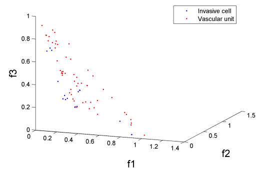


FIGURE 14. Part-2 invasive cells generation.

D. MAIN EXPERIMENTS

The production of different types of individuals are listed in Figure 13 - Figure 16. It is an immediate results of DTLZ1. Figure 13 is the beginning of a generation: vascular units identification. The read dots in Figure 13 are the vascular units in the current generation and they are also the current approximate Pareto front. Figure 14 is the second step: invasive cells generation. The blue dots are generated by the guidance of vascular units and some of them are located at un-searched areas. Figure 15 is the generation of growing cells. The green dots are also generated by the guidance of vascular units and they will search around. Figure 16 is the generation of dormant cells and the identification of dead cells. After the generation and movement of dormant cells, all of the individuals will merge together to get a new approximate Pareto front, and lead the next generation.

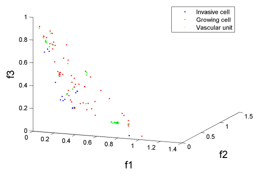


FIGURE 15. Part-3 growing cells generation.

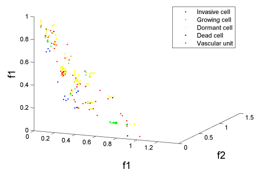


FIGURE 16. Part-4 dormant cells and dead cells generation.

Table 3 - Table 7 are the results of the experiments. M-CDE and M-AGR respectively referred to MOEA/D-CDE and MOEA/D-AGR.

Table 3 is the generational distance of the approximate Pareto front determined by the algorithms in 30 runs. It can be concluded from Table 3 that the proposed VITGO outperforms other five algorithms in most cases, especially the test problems ZDTs and WFGs. Half of them are two-objective problems and the others are three-objective problems. The test problems that cannot be completely solved by VITGO also include two-objective problems and three-objective problems. Therefore, the performance of the proposed VITGO algorithm is independent of the number of the conflicting objective functions of a multi-objective problem. Since generational distance is mostly utilized to identify the convergence, it can be concluded that the convergence of the proposed VITGO is satisfactory in most cases. In those test problems where VITGO does not perform well, its performance is not unacceptable.

Table 4 is the inverse generational distance of the approximate Pareto front determined by the algorithms in 30 runs. In more than half of the cases, the proposed VITGO algorithm can outperform other five algorithms. Similar to Table 3, the proposed VITGO algorithm can almost solve the test problems in ZDT and WFG, and it can also solve some of the test problems in UF and DTLZ. Since IGD can both identify the convergence and diversity of an approximate Pareto front, it can be concluded that both the convergence and diversity of the proposed VITGO is acceptable in more than half of the cases. Even for the test problems that it cannot solve completely, the performance of the VITGO is

TABLE 3. GDs of algorithms.

Problem	-	VITGO	NSGA-III	HEIA	M-CDE	M-AGR	MODA
ZDT1	Avg.	**6.5080e-04**	1.8650e-03	2.8966e-03	7.1908e-03	1.3246e-02	1.6480e-01
	Std.	**1.8610e-05**	1.8227e-04	1.6131e-04	4.4792e-04	1.3572e-04	2.0721e-04
ZDT2	Avg.	**7.3419e-05**	3.5196e-03	1.9742e-03	2.5424e-03	1.1673e-02	1.3894e-02
	Std.	**1.0133e-05**	4.9148e-04	2.1822e-04	1.7676e-04	1.5022e-03	2.8713e-03
ZDT3	Avg.	**9.6785e-04**	1.8513e-03	7.2203e-04	2.6350e-03	2.2862e-02	7.2410e-03
	Std.	**1.9489e-05**	3.4817e-04	1.0920e-04	5.3321e-04	1.2921e-04	9.3550e-04
ZDT4	Avg.	**9.9122e-05**	3.7925e-04	1.9307e-04	4.2055e-04	6.8135e-04	1.1026e-03
	Std.	**6.4582e-07**	6.6218e-05	1.3414e-05	3.2273e-05	1.7714e-05	7.2194e-04
ZDT6	Avg.	**8.8851e-05**	1.5283e-04	1.2242e-04	2.2489e-04	2.8818e-04	3.9150e-04
	Std.	1.4831e-05	1.0537e-05	**1.0528e-05**	2.7399e-05	5.2312e-05	4.9515e-05
UF1	Avg.	**1.3457e-04**	1.9214e-03	9.7005e-04	1.1426e-03	2.3928e-03	2.2654e-03
	Std.	4.2619e-05	7.7912e-04	**1.5815e-05**	5.7279e-04	6.1591e-04	6.4549e-04
UF2	Avg.	**1.0705e-04**	5.3199e-03	1.5653e-03	7.4126e-03	1.4791e-03	4.6798e-03
	Std.	**1.6066e-05**	4.4598e-04	4.1924e-04	1.5135e-04	5.2983e-04	8.1094e-04
UF3	Avg.	1.9427e-02	6.7501e-03	8.3604e-03	**5.5428e-03**	8.5037e-03	2.3102e-02
	Std.	7.8557e-04	5.5028e-04	6.1694e-04	8.3092e-04	**3.5172e-04**	2.1385e-04
UF4	Avg.	**8.4040e-04**	5.6119e-03	2.7972e-02	7.4709e-03	1.7674e-02	9.9127e-02
	Std.	**1.5362e-04**	1.8727e-04	2.5182e-03	7.1984e-04	3.3291e-04	1.5277e-03
UF5	Avg.	**1.7104e-02**	1.0364e-01	8.9471e-02	6.4629e-02	8.9901e-02	1.2283e-01
	Std.	**6.4897e-03**	4.9621e-03	7.7695e-03	1.0703e-02	3.4502e-02	4.1811e-02
UF6	Avg.	2.8002e-02	**1.3764e-03**	4.5372e-03	2.1697e-03	9.2758e-03	1.2185e-02
	Std.	1.2647e-03	1.5239e-04	1.6884e-04	**1.1262e-04**	1.0792e-03	1.3386e-03
UF7	Avg.	**5.3661e-05**	1.3539e-04	2.4287e-03	2.2471e-03	4.3438e-03	1.3851e-02
	Std.	**3.5874e-06**	5.8093e-05	1.9229e-04	1.4678e-04	2.1254e-04	2.5957e-04
UF8	Avg.	**5.8971e-04**	1.5812e-03	7.1430e-04	1.2267e-03	3.0608e-03	5.4724e-03
	Std.	4.4451e-04	1.6506e-04	2.4571e-04	**3.1511e-05**	3.8189e-04	2.6574e-04
UF9	Avg.	3.8587e-04	3.4207e-04	1.4459e-03	**3.4117e-04**	4.1820e-04	2.1209e-03
	Std.	2.1572e-04	1.0364e-04	3.0957e-04	**8.6085e-05**	9.5608e-05	2.9961e-04
UF10	Avg.	4.1709e-03	2.3238e-03	8.6431e-03	**1.2955e-03**	3.2568e-03	8.4077e-03
	Std.	3.3152e-03	1.7829e-04	1.4272e-04	1.5883e-04	9.8493e-04	1.2253e-04
DTLZ1	Avg.	4.4658e-04	**1.2977e-04**	4.0561e-04	3.5582e-04	1.2037e-03	2.1775e-03
	Std.	1.1813e-04	**1.2713e-05**	9.0305e-05	2.5227e-05	3.3924e-04	2.5007e-03
DTLZ2	Avg.	1.7616e-04	**1.2386e-04**	6.1767e-04	6.6045e-04	9.1974e-04	8.5201e-04
	Std.	5.6307e-05	1.6135e-05	**1.3381e-05**	1.7834e-05	2.9198e-05	8.8615e-05
DTLZ3	Avg.	6.9270e-04	**1.4471e-04**	1.3629e-03	8.4118e-04	1.6978e-03	9.7030e-04
	Std.	2.8837e-04	2.7384e-05	1.6957e-04	**1.3016e-05**	6.7958e-04	6.3702e-04
DTLZ4	Avg.	5.4121e-04	**1.5475e-04**	3.1071e-04	3.0371e-04	1.2197e-03	1.4728e-03
	Std.	5.4296e-05	**1.2351e-05**	1.9380e-05	3.1319e-05	9.7301e-05	2.5574e-04
WFG1	Avg.	**3.2141e-04**	5.4944e-04	4.2083e-04	8.4707e-04	1.0010e-03	1.1139e-03
	Std.	**3.4823e-06**	1.4299e-05	2.1192e-05	5.9328e-06	3.3989e-05	3.2103e-05
WFG2	Avg.	6.6502e-04	1.5970e-03	**6.0387e-04**	1.1172e-03	1.5273e-03	1.1984e-03
	Std.	**1.4019e-05**	1.5314e-04	6.0981e-05	1.2019e-04	6.6774e-04	9.4010e-04
WFG3	Avg.	**2.1047e-04**	4.2390e-04	3.8725e-04	4.0613e-04	7.2880e-04	7.2407e-04
	Std.	**7.9576e-07**	1.6291e-05	2.8308e-05	4.7798e-06	1.5401e-05	1.8092e-05
WFG4	Avg.	7.8177e-04	**4.3648e-04**	6.3190e-04	4.8208e-04	1.1785e-03	6.7036e-04
	Std.	**6.3457e-06**	1.0040e-05	8.5198e-05	1.8402e-05	1.1610e-05	2.2392e-05
WFG5	Avg.	**3.5479e-04**	4.2407e-05	4.1194e-04	3.9487e-04	9.4408e-04	1.2854e-03
	Std.	3.5716e-06	2.6101e-06	5.8592e-06	**2.1765e-06**	4.5214e-05	8.4573e-05
WFG6	Avg.	**2.7025e-04**	5.0157e-04	3.3471e-04	3.9082e-04	7.9247e-04	9.0180e-04
	Std.	**6.1233e-07**	3.6289e-06	2.8497e-06	4.4501e-06	1.9385e-06	7.1190e-06
WFG7	Avg.	2.4618e-04	2.8380e-04	**2.3726e-04**	4.0982e-04	8.8192e-04	1.0165e-03
	Std.	**1.5398e-06**	1.6037e-06	4.6193e-06	2.4298e-06	5.8001e-06	7.5474e-05
WFG8	Avg.	**2.7165e-04**	4.7503e-04	3.2719e-04	4.4003e-04	5.0478e-04	5.0310e-04
	Std.	**1.9597e-06**	2.9331e-06	9.7037e-06	7.3982e-06	5.1183e-06	4.7572e-06
WFG9	Avg.	**2.4686e-04**	4.1789e-04	3.0213e-04	3.4501e-04	7.9264e-04	8.9195e-04
	Std.	5.9399e-06	2.1963e-06	**1.4751e-06**	7.5959e-06	3.2392e-06	3.2936e-06

TABLE 4. IGDs of algorithms.

Problem	-	VITGO	NSGA-III	HEIA	M-CDE	M-AGR	MODA
ZDT1	Avg.	**4.2182e-03**	7.7491e-03	6.5914e-03	1.3386e-02	3.8237e-02	6.8303e-02
	Std.	6.6739e-04	**3.4732e-04**	4.2912e-04	4.0009e-04	5.8999e-03	8.4927e-03
ZDT2	Avg.	**2.1471e-03**	4.7851e-03	4.5379e-03	9.0011e-03	1.9628e-02	2.2637e-02
	Std.	**4.0247e-04**	4.3607e-04	1.0913e-03	1.6921e-03	2.9051e-03	2.5877e-03
ZDT3	Avg.	**2.4291e-03**	2.3215e-02	8.4408e-03	3.1901e-02	3.4118e-02	8.1951e-02
	Std.	1.7535e-04	1.3172e-04	**1.2102e-04**	1.7808e-04	1.0241e-04	1.1160e-03
ZDT4	Avg.	**5.4370e-04**	5.3084e-03	2.7040e-03	5.6020e-03	9.6075e-03	1.5880e-02
	Std.	**5.3006e-05**	5.1871e-04	2.5924e-04	5.8177e-04	4.1521e-04	8.6812e-04
ZDT6	Avg.	**3.5271e-03**	8.4028e-03	7.1550e-03	1.2182e-02	1.5574e-02	2.2894e-02
	Std.	4.6458e-04	4.9889e-04	**2.1187e-04**	2.5128e-03	5.5832e-03	8.0920e-03
UF1	Avg.	2.0923e-02	3.7892e-02	4.2073e-02	**1.9812e-02**	4.0347e-02	5.2268e-02
	Std.	9.5183e-03	9.1219e-03	5.8237e-03	**1.8582e-03**	5.1214e-03	3.0752e-03
UF2	Avg.	**1.3417e-02**	1.3785e-02	2.2649e-02	1.4301e-02	2.7307e-02	9.1752e-02
	Std.	3.2501e-03	3.4910e-03	3.9683e-03	5.6982e-03	4.8715e-03	**2.3706e-03**
UF3	Avg.	2.1270e-01	1.8574e-01	2.1925e-01	**1.7061e-01**	2.1240e-01	3.1550e-01
	Std.	1.2989e-01	2.4534e-02	**1.3227e-02**	2.5282e-02	2.1224e-02	8.1110e-02
UF4	Avg.	2.3989e-02	2.6743e-02	8.5746e-02	**2.3165e-02**	4.9107e-02	2.2344e-01
	Std.	5.4981e-03	1.8810e-02	2.0217e-02	**2.0024e-03**	2.6957e-03	1.4338e-02
UF5	Avg.	**1.5502e-01**	1.6573e-01	2.4961e-01	2.1057e-01	2.9471e-01	4.0810e-01
	Std.	1.9382e-02	**1.7779e-02**	6.6371e-02	1.0072e-02	2.9343e-02	2.2491e-01
UF6	Avg.	1.5771e-01	**2.6073e-02**	9.9289e-02	5.0819e-02	1.9354e-01	2.2738e-01
	Std.	6.0104e-02	5.0287e-03	7.7411e-03	**3.7049e-03**	8.9068e-02	2.2124e-01
UF7	Avg.	**1.2025e-02**	1.7908e-02	7.2752e-02	5.7491e-02	9.5982e-02	2.1094e-01
	Std.	3.2581e-03	2.9016e-03	2.0007e-03	2.0326e-03	3.2785e-03	4.8971e-02
UF8	Avg.	8.1675e-02	**5.4406e-02**	6.4300e-02	8.5391e-02	1.0684e-01	1.6102e-01
	Std.	1.5417e-02	**7.6034e-03**	2.2190e-02	1.0627e-02	6.0343e-03	2.2985e-02
UF9	Avg.	**1.0752e-02**	1.1361e-02	1.3649e-01	1.0873e-02	1.6372e-02	1.5551e-01
	Std.	**5.5306e-04**	1.2025e-03	2.4908e-02	1.2487e-03	1.1214e-03	5.0592e-02
UF10	Avg.	1.1457e-01	6.2750e-02	3.1334e-01	**5.2040e-02**	1.2671e-01	2.4835e-01
	Std.	1.9015e-02	4.4118e-03	1.3224e-02	1.2978e-02	**2.2025e-03**	1.0090e-02
DTLZ1	Avg.	**5.3501e-03**	5.7685e-03	2.0887e-02	1.5603e-02	2.6052e-02	1.7285e-02
	Std.	1.6382e-03	**1.2741e-03**	3.4725e-03	7.6573e-03	2.0780e-03	7.0992e-03
DTLZ2	Avg.	**8.7281e-03**	9.5239e-03	4.1389e-02	5.6484e-02	8.9743e-02	1.2782e-01
	Std.	**1.1290e-04**	2.2907e-03	5.3918e-03	9.5502e-03	8.5871e-03	2.2425e-02
DTLZ3	Avg.	3.0285e-02	**1.8047e-02**	1.6039e-01	6.3853e-02	1.1758e-01	9.1739e-02
	Std.	5.9827e-03	**5.1532e-03**	2.5637e-02	1.1745e-02	1.9283e-02	9.7800e-03
DTLZ4	Avg.	1.4502e-02	**8.2576e-03**	2.2649e-02	3.5432e-02	9.3954e-02	1.1121e-01
	Std.	1.2891e-03	**8.8605e-04**	1.1710e-03	9.2396e-03	2.7539e-03	7.1023e-03
WFG1	Avg.	**1.5018e-02**	3.1329e-02	2.6710e-02	5.0980e-02	5.9608e-02	6.4581e-02
	Std.	**5.4121e-04**	1.5851e-03	2.8792e-03	1.1680e-03	3.5291e-03	5.9579e-03
WFG2	Avg.	**1.7852e-02**	3.6370e-02	2.3601e-02	3.6825e-02	6.0282e-02	7.4392e-02
	Std.	**5.3089e-04**	1.0531e-03	7.3692e-03	3.1425e-03	3.6173e-03	4.3914e-03
WFG3	Avg.	**1.6975e-02**	2.4478e-02	2.0194e-02	3.4270e-02	4.1860e-02	4.2659e-02
	Std.	**1.5734e-04**	1.7691e-04	5.1576e-04	2.4576e-04	9.1138e-04	4.6572e-04
WFG4	Avg.	**2.1375e-02**	2.3507e-02	3.6706e-02	3.2492e-02	6.0518e-02	3.9857e-02
	Std.	1.9427e-03	2.9761e-03	3.2192e-03	1.8927e-03	**1.8103e-03**	6.8275e-03
WFG5	Avg.	**1.8705e-02**	2.1602e-02	2.2647e-02	3.9722e-02	5.3603e-02	5.6487e-02
	Std.	3.1759e-03	3.2392e-03	**2.3197e-03**	4.7805e-03	7.6284e-03	5.9213e-03
WFG6	Avg.	**1.7312e-02**	2.8623e-02	1.8927e-02	2.3417e-02	5.2157e-02	5.5264e-02
	Std.	3.1380e-04	3.0781e-04	1.1209e-04	**1.0854e-04**	3.8255e-04	2.9060e-04
WFG7	Avg.	2.4715e-02	**1.7841e-02**	1.9574e-02	3.0119e-02	5.9317e-02	6.5471e-02
	Std.	**1.1229e-04**	4.0381e-04	3.7172e-04	3.8273e-04	3.2759e-04	4.8020e-04
WFG8	Avg.	**2.2074e-02**	2.9706e-02	2.2140e-02	2.5178e-02	3.0272e-02	4.1619e-02
	Std.	**3.7322e-05**	4.6501e-04	7.5082e-04	3.4183e-04	2.6024e-04	6.0683e-04
WFG9	Avg.	2.2375e-02	2.6479e-02	2.0917e-03	**2.0842e-02**	4.8693e-02	5.6510e-02
	Std.	**1.6350e-04**	9.7537e-04	2.1784e-04	6.6375e-04	1.0854e-03	1.0776e-03

not unacceptable. It can also be concluded from Table 4 that the performance of the VITGO algorithm is independent of the number of the conflicting objective functions of a multi-objective problem.

Table 5 is the Spread of the approximate Pareto front determined by the algorithms in 30 runs. It can be utilized to identify the nearest neighbor of each individual in the approximate Pareto front. To a certain extent, it can be utilized to identify the diversity of an algorithm. If the distribution of the approximate Pareto front is not uniform, the performance metric SP may fail. For example, Figure 23 shows the final approximate Pareto front of UF2, which is actually not complete (the diversity of it is not so much good). However, the SP of UF2 in Table 5 is even better than that of ZDT3, in which diversity of the approximate Pareto front is obviously better than that of UF2. The reason is that the performance metric SP is utilized to identify the nearest neighbor of each individual, and the nearest neighbor of each individual in UF2 is closer than that of ZDT3. Therefore, to a certain extent SP can be utilized to identify the diversity of an approximate Pareto front, but not in all of the cases. According to Table 5, it can be concluded that the performance of our proposed VITGO is also acceptable.

Table 6 is the hypervolume of the approximate Pareto front determined by the algorithms in 30 runs. It can identify the dominance of an approximate Pareto front. According to

TABLE 5. SPs of algorithms.

Problem	-	VITGO	NSGA-III	HEIA	M-CDE	M-AGR	MODA
ZDT1	Avg.	**6.0352e-04**	2.0418e-03	1.4957e-03	1.1379e-02	1.7782e-02	2.8548e-02
	Std.	**2.9218e-04**	8.9192e-04	1.0291e-04	1.9138e-04	1.2654e-04	2.5038e-04
ZDT2	Avg.	**5.3038e-04**	1.7714e-03	7.1785e-03	3.7713e-03	4.0972e-03	3.8719e-03
	Std.	**5.9171e-05**	1.3739e-04	4.7102e-04	1.2330e-04	8.5198e-04	5.3762e-04
ZDT3	Avg.	**2.3992e-03**	7.3098e-03	9.3542e-03	4.8571e-03	1.0245e-02	1.3249e-02
	Std.	**2.2291e-04**	1.8918e-03	1.6192e-03	5.9574e-04	1.8423e-03	1.2546e-03
ZDT4	Avg.	**4.7331e-04**	9.5236e-04	1.2973e-03	2.4602e-03	2.6198e-03	6.0012e-03
	Std.	**4.2522e-05**	1.6143e-04	2.2112e-04	8.0982e-04	2.4417e-04	2.5983e-04
ZDT6	Avg.	**5.8055e-04**	2.8213e-03	1.4998e-03	2.0012e-03	7.5508e-03	4.9542e-03
	Std.	**1.1534e-04**	4.3332e-03	1.9527e-03	7.4752e-03	3.0556e-04	9.0433e-03
UF1	Avg.	**6.3293e-03**	2.5019e-02	1.5323e-02	2.9120e-02	2.2098e-02	1.7923e-02
	Std.	1.3482e-02	**1.2910e-03**	1.2417e-02	7.5298e-02	1.9119e-04	2.0668e-02
UF2	Avg.	**8.4588e-04**	1.7052e-03	1.2294e-03	3.1173e-03	2.9425e-03	3.9520e-03
	Std.	6.5318e-04	8.3387e-04	8.4780e-04	**5.4183e-04**	7.1329e-04	5.4863e-04
UF3	Avg.	2.0192e-03	**1.8503e-03**	3.0750e-03	2.9282e-03	1.2437e-02	6.4602e-02
	Std.	7.5395e-04	**2.4279e-04**	3.1661e-04	7.1208e-04	1.6683e-03	3.2097e-03
UF4	Avg.	4.5932e-03	**4.3577e-03**	6.5329e-03	2.1924e-02	2.8447e-02	2.2835e-02
	Std.	2.2001e-03	1.7592e-03	1.9228e-03	2.7183e-03	1.9539e-03	2.9277e-03
UF5	Avg.	6.0957e-02	**1.3720e-02**	6.8800e-02	3.9571e-02	4.8711e-02	8.2820e-02
	Std.	7.3154e-02	1.9142e-02	**1.2115e-02**	3.7817e-02	2.2301e-02	3.9337e-02
UF6	Avg.	2.5035e-01	7.0641e-02	1.0364e-01	**2.7419e-02**	7.3705e-02	3.4072e-02
	Std.	2.3381e-01	**1.2778e-03**	1.5323e-02	2.9265e-03	1.9518e-03	4.1796e-03
UF7	Avg.	**1.3293e-03**	1.5172e-03	1.9322e-03	1.3728e-03	2.2350e-03	3.1517e-03
	Std.	**1.7372e-04**	2.5298e-04	1.1165e-04	1.7293e-04	2.1900e-03	1.9962e-03
UF8	Avg.	3.6217e-02	5.9614e-02	2.4532e-02	**6.5847e-03**	6.3771e-02	2.8336e-02
	Std.	4.8973e-03	6.4675e-04	8.0429e-03	**1.5127e-04**	2.2779e-03	3.9250e-03
UF9	Avg.	6.4158e-03	6.5602e-03	7.1197e-03	**6.5019e-03**	1.0882e-02	1.3594e-02
	Std.	2.5812e-03	**2.3097e-04**	3.0886e-03	2.4100e-03	3.5983e-03	5.3452e-03
UF10	Avg.	3.7129e-02	**3.5147e-02**	4.4715e-02	6.8710e-02	7.8651e-02	7.3656e-02
	Std.	4.7385e-03	1.6350e-03	1.6584e-03	**1.6011e-03**	4.0650e-03	4.4854e-03
DTLZ1	Avg.	4.5041e-03	**3.7811e-03**	5.5237e-03	3.7400e-03	4.1781e-03	1.6643e-02
	Std.	6.7182e-03	**4.2917e-04**	1.6605e-03	4.0981e-03	6.4550e-03	9.0182e-03
DTLZ2	Avg.	9.0013e-03	**7.1561e-03**	1.4369e-02	1.5942e-02	3.74e-02	5.459e-02
	Std.	1.4579e-03	**1.2257e-03**	1.2810e-03	2.1781e-03	1.5932e-03	4.9418e-02
DTLZ3	Avg.	1.7592e-02	3.2887e-02	6.9030e-02	**1.2101e-02**	1.8675e-02	1.0227e-01
	Std.	1.6018e-03	1.4785e-03	4.8007e-03	**1.3512e-03**	1.6627e-03	7.7895e-03
DTLZ4	Avg.	1.3582e-02	**8.7827e-03**	4.8517e-02	5.1501e-02	5.4025e-02	1.5138e-01
	Std.	2.5019e-03	**1.7802e-03**	3.4197e-02	5.0610e-02	1.1139e-02	4.9029e-02
WFG1	Avg.	**1.3982e-02**	2.2345e-02	3.5143e-02	2.1638e-02	2.6571e-02	1.5034e-02
	Std.	**2.5401e-04**	5.1393e-04	4.2810e-03	1.5070e-03	7.1274e-03	7.8092e-03
WFG2	Avg.	1.8572e-02	2.0197e-02	**1.4605e-02**	2.7421e-02	2.8767e-02	2.8140e-02
	Std.	2.8192e-03	3.8720e-03	3.0518e-03	**1.5072e-03**	2.8513e-03	6.0895e-03
WFG3	Avg.	**1.4974e-02**	2.5343e-02	2.6580e-02	2.4107e-02	2.4792e-02	2.4417e-02
	Std.	**1.3114e-04**	1.8317e-04	3.5892e-04	2.1453e-04	1.8040e-04	2.2719e-04
WFG4	Avg.	**2.1292e-02**	2.8720e-02	2.052e-02	3.3124e-02	3.0798e-02	1.5901e-02
	Std.	4.1845e-03	3.6474e-03	**3.3129e-03**	5.2373e-03	3.5832e-03	5.4895e-03
WFG5	Avg.	**1.9517e-02**	2.7677e-02	2.5084e-02	3.1739e-02	3.3209e-02	2.5752e-02
	Std.	5.4897e-04	4.0518e-04	**3.8692e-04**	4.0758e-04	7.6570e-04	3.8415e-03
WFG6	Avg.	**1.8219e-02**	1.8658e-02	2.2247e-02	3.3001e-02	4.1309e-02	1.4172e-02
	Std.	4.6330e-04	4.7174e-04	**2.0371e-04**	5.5187e-04	3.3479e-04	2.7351e-04
WFG7	Avg.	1.9082e-02	**1.8831e-02**	2.3934e-02	3.3192e-02	3.5048e-02	2.1507e-02
	Std.	1.9145e-03	**1.6751e-03**	2.0473e-03	4.1337e-03	6.1983e-03	5.9417e-03
WFG8	Avg.	**1.8087e-02**	2.8762e-02	2.6715e-02	2.0050e-02	3.0392e-02	4.4911e-02
	Std.	2.5974e-03	1.3012e-03	3.7854e-03	**1.2109e-03**	1.7869e-03	3.5631e-03
WFG9	Avg.	2.1795e-02	2.6482e-02	2.1083e-02	**2.0932e-02**	3.4721e-02	3.9880e-02
	Std.	4.4569e-03	2.1037e-03	8.9714e-04	**7.0050e-04**	1.8291e-03	1.5417e-03

TABLE 6. HVs of algorithms.

Problem	-	VITGO	NSGA-III	HEIA	M-CDE	M-AGR	MODA
ZDT1	Avg.	**0.7237**	0.7146	0.7175	0.7050	0.6703	0.6355
	Std.	4.013e-03	**1.6281e-03**	6.4420e-03	4.4177e-03	7.4984e-02	1.2113e-02
ZDT2	Avg.	**0.4541**	0.4408	0.4426	0.4361	0.4077	0.3875
	Std.	5.3991e-03	**1.0662e-03**	7.3832e-03	6.4117e-03	1.8205e-02	4.8834e-02
ZDT3	Avg.	**0.8311**	0.8095	0.8255	0.8022	0.7963	0.7471
	Std.	4.4921e-03	2.6683e-03	**2.1912e-03**	3.1569e-03	1.3921e-02	7.28739e-02
ZDT4	Avg.	**0.7235**	0.7163	0.7174	0.7149	0.7105	0.6963
	Std.	**2.9317e-03**	5.3121e-03	4.8293e-03	7.2481e-03	1.8483e-02	3.6192e-03
ZDT6	Avg.	**0.4216**	0.4075	0.4092	0.4072	0.4002	0.3946
	Std.	6.071e-03	1.2429e-03	**1.5275e-04**	2.8298e-03	5.0713e-03	1.2038e-02
UF1	Avg.	0.6988	0.5797	0.5614	**0.7035**	0.5690	0.4234
	Std.	**1.3810e-02**	2.7449e-02	6.2127e-02	1.5074e-02	6.1692e-02	3.7073e-02
UF2	Avg.	0.7112	**0.7177**	0.5417	0.7101	0.5649	0.4247
	Std.	7.1198e-03	8.9013e-03	5.3082e-03	6.7147e-03	**2.9109e-03**	1.2895e-02
UF3	Avg.	0.3724	0.4366	0.3820	**0.4613**	0.4092	0.2603
	Std.	4.8249e-02	**1.7407e-02**	7.9429e-02	8.2468e-02	4.1332e-02	5.3573e-02
UF4	Avg.	0.4068	0.4320	0.3227	**0.4347**	0.3965	0.1570
	Std.	1.0410e-02	1.9299e-02	1.3417e-02	**7.6152e-03**	3.6619e-02	8.5287e-02
UF5	Avg.	**0.3690**	0.3502	0.2785	0.3264	0.2157	0.1487
	Std.	**1.6518e-02**	1.9910e-02	7.2129e-02	4.4380e-02	3.2518e-02	2.3294e-02
UF6	Avg.	0.2931	**0.5016**	0.4057	0.4631	0.2972	0.2274
	Std.	7.8324e-02	**1.4369e-02**	7.4824e-02	2.3317e-02	4.0742e-02	1.1728e-02
UF7	Avg.	**0.5710**	0.5625	0.4885	0.5220	0.4375	0.3249
	Std.	6.3501e-03	6.5298e-03	8.0104e-03	**5.3190e-03**	5.2379e-02	7.2243e-02
UF8	Avg.	0.4527	**0.4865**	0.4702	0.4582	0.4313	0.3216
	Std.	5.6721e-03	1.9018e-03	6.8582e-03	**1.7401e-03**	3.7429e-03	4.6738e-03
UF9	Avg.	0.7810	**0.7831**	0.6282	0.7824	0.7345	0.6110
	Std.	4.1192e-03	4.6470e-03	**3.4293e-03**	5.3922e-03	9.9013e-03	7.0112e-03
UF10	Avg.	0.3171	0.4386	0.2181	**0.4527**	0.3233	0.2375
	Std.	7.5398e-03	**1.2684e-03**	4.9287e-02	6.8192e-03	1.3511e-02	3.7062e-02
DTLZ1	Avg.	0.8471	**0.8493**	0.8051	0.8197	0.7897	0.8198
	Std.	4.7645e-03	**1.4918e-03**	1.1912e-02	5.5678e-03	1.7854e-02	5.0781e-02
DTLZ2	Avg.	**0.5962**	0.5937	0.5378	0.4983	0.4380	0.3587
	Std.	6.0513e-03	**1.0960e-03**	1.6885e-02	1.4431e-02	9.2918e-03	1.8481e-02
DTLZ3	Avg.	0.5606	**0.5780**	0.3223	0.4936	0.3820	0.4088
	Std.	1.3854e-02	9.8517e-03	**1.2320e-03**	1.8682e-02	7.3219e-03	1.3058e-02
DTLZ4	Avg.	0.5887	**0.5924**	0.5572	0.5399	0.4103	0.3908
	Std.	4.5872e-03	**1.2719e-03**	5.7110e-02	1.3129e-02	1.9487e-02	6.4658e-02
WFG1	Avg.	**0.9631**	0.9398	0.9483	0.9360	0.9350	0.9328
	Std.	**2.4079e-03**	1.5118e-02	8.6931e-03	2.8792e-03	1.4110e-02	3.9785e-02
WFG2	Avg.	**0.9498**	0.9273	0.9362	0.9298	0.9238	0.9193
	Std.	**8.5049e-04**	2.8083e-02	5.8192e-03	7.8997e-03	6.2024e-03	1.5379e-02
WFG3	Avg.	**0.8701**	0.8576	0.8679	0.8515	0.8509	0.8468
	Std.	**1.4832e-03**	3.9815e-03	4.0192e-03	3.1817e-03	2.8428e-02	4.3574e-02
WFG4	Avg.	**0.5794**	0.5715	0.5644	0.5709	0.5582	0.5667
	Std.	2.5139e-03	1.7834e-03	**1.1693e-03**	3.2291e-03	3.6184e-02	4.3109e-02
WFG5	Avg.	**0.5720**	0.5665	0.5637	0.5542	0.5457	0.5421
	Std.	**8.5440e-04**	1.0819e-03	1.0528e-03	2.2242e-03	1.4672e-02	2.4908e-02
WFG6	Avg.	**0.6014**	0.5779	0.5865	0.5853	0.5683	0.5650
	Std.	2.3192e-03	**2.1507e-03**	8.5389e-03	7.8921e-03	7.2293e-03	5.8672e-03
WFG7	Avg.	0.5894	0.5906	**0.5934**	0.5868	0.5471	0.5256
	Std.	1.2192e-03	**1.0968e-03**	1.2237e-03	2.0382e-03	1.9019e-03	2.8973e-03
WFG8	Avg.	**0.5947**	0.5841	0.5867	0.5835	0.5818	0.5754
	Std.	3.6192e-03	1.2317e-03	1.6752e-03	**1.0073e-03**	3.6187e-03	5.5302e-03
WFG9	Avg.	0.5855	0.5785	0.5861	**0.5896**	0.5679	0.5548
	Std.	9.0185e-03	6.158e-04	8.8517e-04	**3.0920e-04**	4.1573e-03	9.7291e-03

Table 6, it can be concluded that the dominance of the approximate Pareto front determined by the proposed VITGO is satisfactory in most cases. For the test problems that it cannot solve completely, the performance is also acceptable. It can also be concluded that the performance of the VITGO algorithm is independent of the number of the conflicting objective functions of a multi-objective problem. Therefore, the performance of the proposed VITGO algorithm is better than others in more than half of the cases.

Table 7 is the average time cost of the six algorithms. It can be concluded from Table 7 that the time cost of the proposed VITGO algorithm is worse than some of the other algorithms.

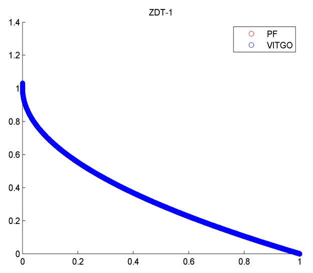


FIGURE 17. ZDT1.

The reason is that the farthest search and expanding population size will cause a lot of computations which cannot be omitted. However, it is not necessarily unacceptable.

TABLE 7. Time cost of algorithms (s).

Problem	VITGO	NSGA-III	HEIA	M-CDE	M-AGR	MODA
ZDT1	7.2874e+02	5.0598e+02	**4.5729e+02**	1.0312e+03	9.6029e+02	1.0708e+03
ZDT2	6.7402e+02	5.0921e+02	**4.9044e+02**	9.6522e+02	9.5218e+02	1.8278e+03
ZDT3	7.7580e+02	5.1383e+02	**5.8885e+02**	9.3339e+02	9.1715e+02	1.0805e+03
ZDT4	3.5762e+02	**3.2065e+02**	4.0234e+02	9.3291e+02	8.6118e+02	9.8921e+02
ZDT6	2.9474e+02	**2.4514e+02**	5.6693e+02	9.0716e+02	8.0731e+02	9.7157e+02
UF1	2.4892e+02	1.8826e+02	**1.4847e+02**	1.3722e+03	9.3242e+02	8.0703e+02
UF2	3.1043e+02	**1.4792e+02**	2.0495e+02	1.7135e+03	8.3730e+02	8.1967e+02
UF3	2.8852e+02	**1.8787e+02**	2.0272e+02	1.3141e+03	8.3968e+02	7.0655e+02
UF4	5.5364e+02	2.6924e+02	**2.6091e+02**	1.2015e+03	9.0863e+02	7.3921e+02
UF5	3.8370e+02	**3.0692e+02**	3.1089e+02	1.0044e+03	5.0177e+02	6.5209e+02
UF6	3.6491e+02	**2.9070e+02**	3.1407e+02	1.0555e+03	8.4309e+02	6.5921e+02
UF7	3.6197e+02	**2.9967e+02**	3.2059e+02	1.1910e+03	6.0482e+02	6.7366e+02
UF8	1.4125e+03	8.7294e+02	**7.6473e+02**	1.2473e+03	1.6198e+03	7.6799e+02
UF9	1.5477e+03	6.6786e+02	**4.1891e+02**	1.1470e+03	1.6456e+03	8.1725e+02
UF10	8.5122e+02	8.9066e+02	**4.5121e+02**	1.1344e+03	8.8526e+02	7.9121e+02
DTLZ1	1.3053e+03	7.9100e+02	**7.6150e+02**	1.2309e+03	1.5422e+03	7.6333e+02
DTLZ2	1.7093e+03	8.4932e+02	**8.1522e+02**	1.8566e+03	1.7698e+03	1.0003e+03
DTLZ3	1.6142e+03	**7.8021e+02**	8.4151e+02	1.7108e+03	1.3114e+03	6.8089e+02
DTLZ4	8.5530e+02	1.2002e+03	**6.8156e+02**	1.9700e+03	1.7489e+03	8.6841e+02
WFG1	1.7588e+03	1.0727e+03	**9.9167e+02**	1.0163e+03	1.1294e+03	1.2705e+03
WFG2	1.8720e+03	1.0730e+03	**8.5428e+02**	9.8943e+02	1.1148e+03	1.0520e+03
WFG3	1.6857e+03	1.7864e+03	**8.9445e+02**	1.0224e+03	1.1275e+03	1.2626e+03
WFG4	1.6340e+03	1.2395e+03	**9.0320e+02**	9.8627e+02	1.1143e+03	1.2508e+03
WFG5	1.6979e+03	1.3656e+03	**9.0832e+02**	1.0468e+03	1.1226e+03	1.2977e+03
WFG6	1.7962e+03	1.3755e+03	8.9479e+02	**8.9393e+02**	1.1399e+03	1.2788e+03
WFG7	2.0154e+03	1.7666e+03	9.0530e+02	**8.8495e+02**	1.1220e+03	1.2342e+03
WFG8	1.9195e+03	1.7742e+03	**9.0197e+02**	1.0119e+03	1.1502e+03	1.3313e+03
WFG9	2.6512e+03	1.8393e+03	**9.1567e+02**	1.0527e+03	1.1873e+03	1.3578e+03

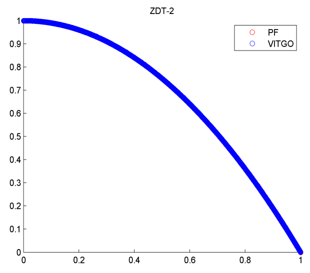


FIGURE 18. ZDT2.

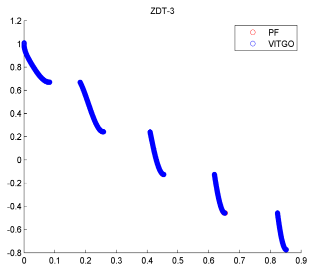


FIGURE 19. ZDT3.

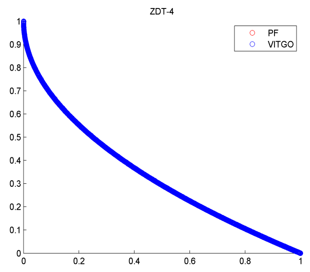


FIGURE 20. ZDT4.

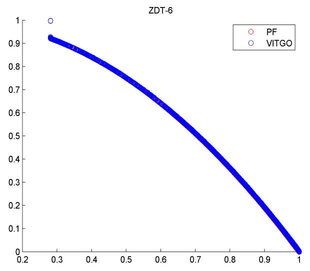


FIGURE 21. ZDT6.

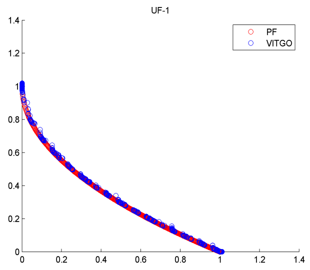


FIGURE 22. UF1.

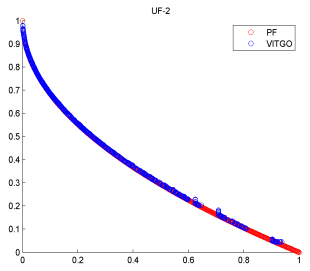


FIGURE 23. UF2.

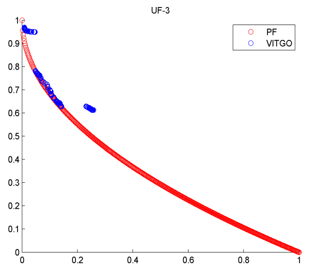


FIGURE 24. UF3.

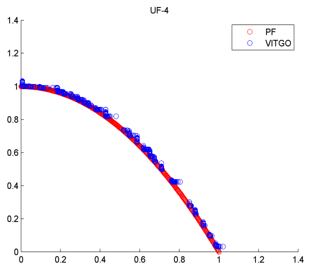


FIGURE 25. UF4.

To analyze the performance of the VITGO algorithm in depth, its graphical results are also provided in Fig 17 - Fig 44. The red dots in these figures refer to the true Pareto front and the blue dots refer to the approximate Pareto front gotten by VITGO. From these figures we can intuitively identify the performance of VITGO and mutually verify its performance with the performance metrics.

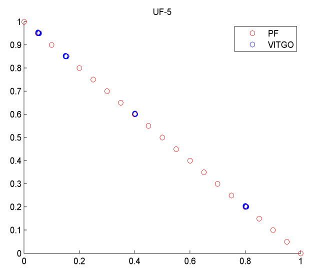


FIGURE 26. UF5.

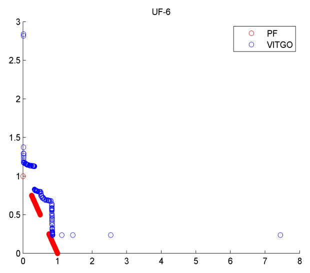


FIGURE 27. UF6.

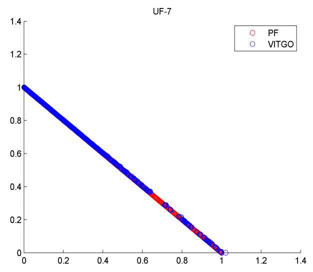


FIGURE 28. UF7.

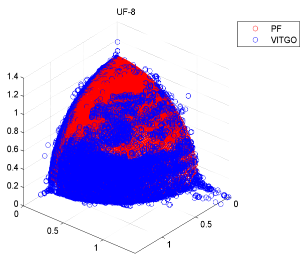


FIGURE 29. UF8.

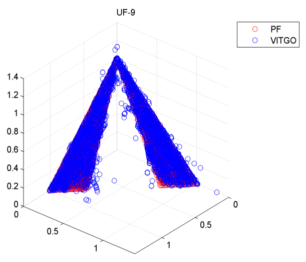


FIGURE 30. UF9.

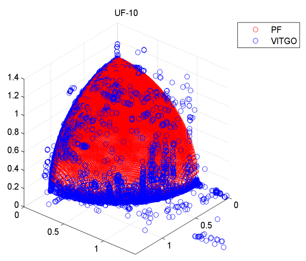


FIGURE 31. UF10.

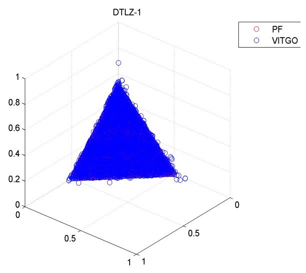


FIGURE 32. DTLZ1.

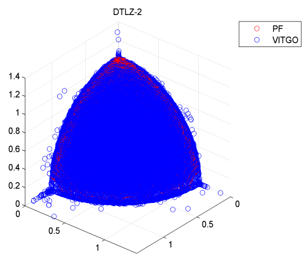


FIGURE 33. DTLZ2.

The graphical results of ZDTs are provided in Fig 17 - Fig 21. It can be concluded from these figures that the proposed VITGO algorithm can almost solve these problems completely. Table 3 - Table 6 support this conclusion. According to paper [39], ZDT1 has a convex Pareto front and its solutions are uniformly distributed; ZDT2 has a Pareto front which is the non-convex counterpart to that of ZDT1; ZDT3 has a Pareto front which consists of several non-continuous convex parts and its search space is unbiased; ZDT4 is multi-model; and ZDT6 is not uniformly distributed, especially near its Pareto front [39]. This level of bias, unbalance or multi-model in ZDTs seems nothing to the proposed VITGO. The test problems can be successfully solved. Moreover, we can probably draw a conclusion that properties of true Pareto front have no effect on the VITGO algorithm since the VITGO algorithm searches in the decision space and the Pareto front displays in the objective space. Instead, the properties of the Pareto solutions are more likely to affect the performance of VITGO. We will continue to discuss these properties in the following subsubsections.

The graphical results of UFs are provided in Fig 22 - Fig 31. It can be concluded from these figures that the true Pareto front of UFs have all been reached. However, the approximate Pareto front obtained by VITGO is not

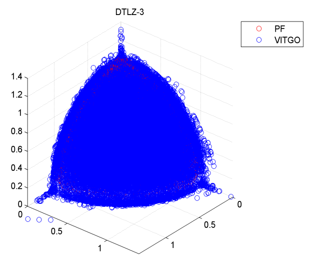


FIGURE 34. DTLZ3.

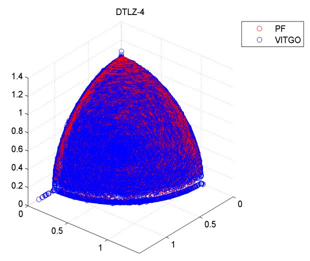


FIGURE 35. DTLZ4.

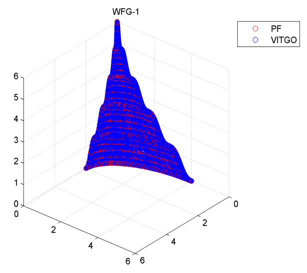


FIGURE 36. WFG1.

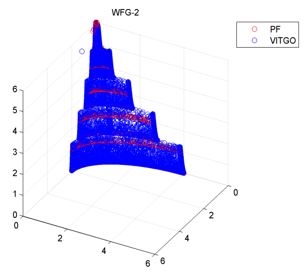


FIGURE 37. WFG2.

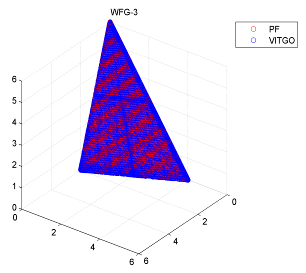


FIGURE 38. WFG3.

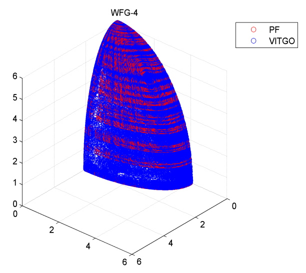


FIGURE 39. WFG4.

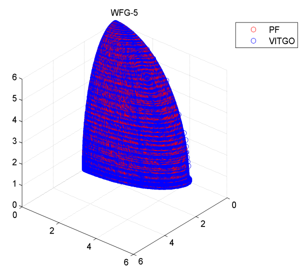


FIGURE 40. WFG5.

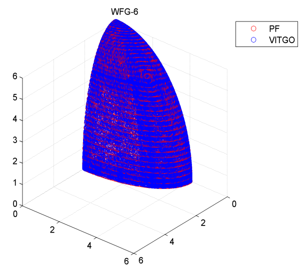


FIGURE 41. WFG6.

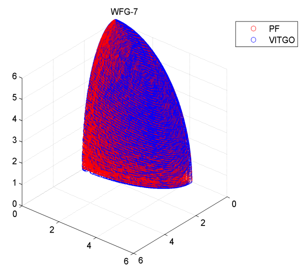


FIGURE 42. WFG7.

always complete or not all of the individuals in the approximate Pareto front have reached the true Pareto front. The two-objective problems in UF can be farther divided into two parts: i) test problems that can almost be completely solved, including UF1, UF2, and UF7; and ii) test problems that cannot be completely solved, including UF3, UF4, UF5, and UF6. According to paper [41], objective functions in the test set UF are composed of trigonometric functions, squares, sets and a large number of products and summations. The concavity and monotonicity of these objective functions are very complicated and this kind of properties are really different from that of the corresponding Pareto front.

According to the conclusions obtained above, the properties of true Pareto front may not affect the performance of VITGO and the properties of Pareto solutions

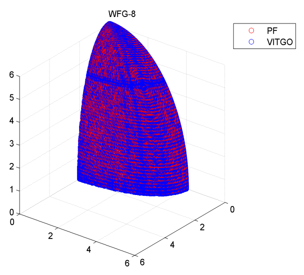


FIGURE 43. **WFG8.**

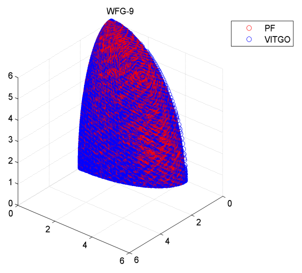


FIGURE 44. **WFG9.**

could do. Therefore, the properties of the Pareto solutions should also be taken into consideration. According to paper [41], Pareto solutions of UFs can be continuous (UF1, UF2, UF3, UF4, UF7) or discrete (UF5, UF6). Continuous Pareto solutions represent a situation that they can be determined by searching around their neighbors in the decision space, which is exactly what we did in the vascular unit generation, growing cell generation and dormant cell generation in VITGO. Discrete Pareto solutions are more likely to be determined by the Levy flight, the random walk or the improved boundary detection scheme (in VITGO). In other words, the latter operation is based on the probability, and the former one is based on stably logical searching. This is why VITGO cannot perform well in UF5 and UF6, but instead it can perform well in UF1, UF2 and UF7. If the objective functions drastically change their values, the properties of objective functions can still affect the performance of VITGO. For example, if an objective function dramatically changes its monotonicity in a small range (such as 0.000001), logical search of VITGO may fail and only the random surrounding search could succeed. In this situation, the performance of VITGO is definitely affected, such as in UF3. However, this kind of influence is smaller than that of the distributions of Pareto solutions since ZDTs can be successfully solved by VITGO while ignoring their difficulties. Three-objective test problems in UF show similar characteristics. Fig.29 - Fig.31 indicates that the VITGO could almost achieve the complete Pareto front of UF9, but not UF8 and UF10. However, due to the difficulties of UF9 it still can not be solved completely. Some individuals in the approximate Pareto front of UF9 is far from the true Pareto front compared with other ones. Table 3 - Table 6 could also verified this conclusion.

The graphical results of DLTZs are provided in Fig.32 - Fig.35. It can be concluded from these graphical results that the true Pareto front of DTLZ1-DTLZ4 have already been reached, but some individuals in the approximate Pareto front were not good enough. According to paper [40], the objective functions of DTLZs are composed of trigonometric functions, exponential functions, products and summations, but the complexity of them is less than that of UFs [40]. In other words, these functions can be solved more easily and the Pareto solutions of DLTZs can be more easier to determined. Similar conclusion can be drawn from Table 3 - Table 6 that the proposed VITGO algorithm can perform better in DTLZs than UFs.

The graphical results of WFGs are provided in Fig.36 - Fig.44. It can be concluded from these graphical results that WFG1-WFG9 have already been solved by VITGO, both considering the convergence and diversity. According to paper [46], the objective functions of WFGs are composed of polynomial, multi-model functions and linear functions, which to be transformed by bias, shift, or reduction to cause different difficulties. Different from the objective functions in UFs, this kind of transformations in WFGs is simple and gentle, and then does not cause very dramatic changes in objective functions. Therefore, the schemes in VITGO such as endpoint generation, vascular guidance and movement, invasion, cell generation, and even the improved boundary detection scheme can play the their biggest roles in determining the approximate Pareto front. Finally, as shown in Fig.36 - Fig.44, the approximate Pareto front of WFGs are almost closed to the corresponding true Pareto front, just as those in ZDTs. The results in Table 3 - Table 6 support the conclusion that VITGO can almost solve the WFGs.

In conclusion, the proposed VITGO can outperform other five multi-objective algorithms in more than half of the 28 test problems. Even in the test problems where VITGO does not perform well, its performance is still acceptable. More than half of the 28 test problems can be solved completely by the proposed VITGO algorithm. The effect of the VITGO is mostly based on the distributions of Pareto solutions, and partly based on the monotonicity and the dispersion of the objective functions. The properties of the true Pareto front cannot directly affect the performance of VITGO. According to the "no free lunch" in paper [47], the performance of VITGO is satisfactory to a certain extent. In the future we will make it more targeted to solve a real-world multi-objective problem rather than solve MOPs generally.

E. VITGO WITH 20-OBJECTIVE PROBLEMS

To comprehensively evaluate the performance of VITGO, an additional experiment based on 20-objective problems is provided. Scalable test problems DTLZ1-DTLZ7 were chosen to identify the performance of VITGO. The number of objective functions of DTLZs is set to 20. A parallel coordinates plot was provided to display the approximate Pareto front determined by VITGO. In order to evaluate the results more accurately, the parameter *maxG* was further

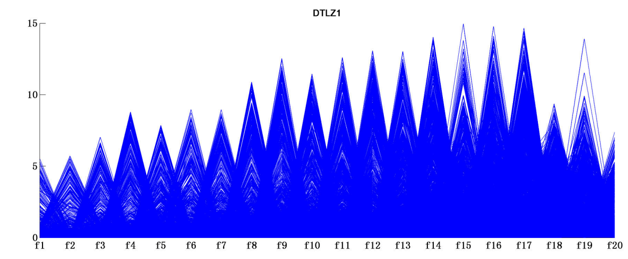


FIGURE 45. DTLZ1-20(maxG=1000).

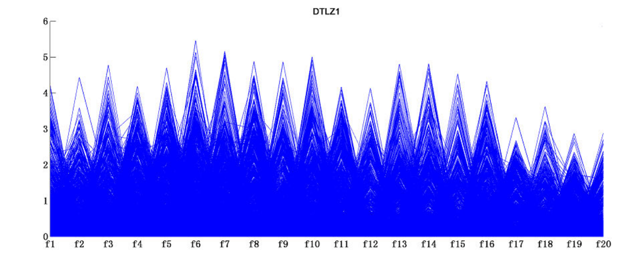


FIGURE 46. DTLZ1-20(maxG=5000).

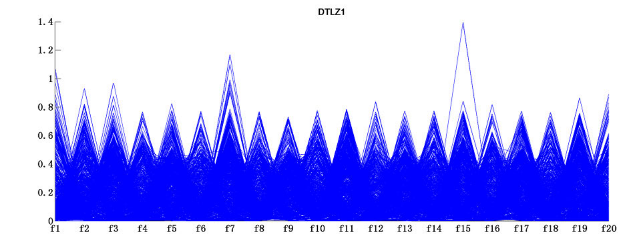


FIGURE 47. DTLZ1-20(maxG=20000).

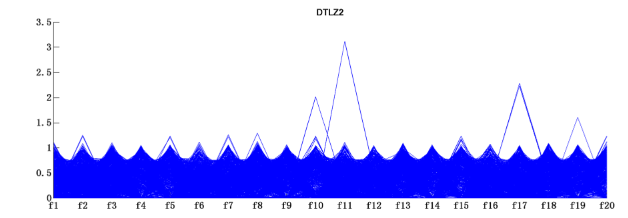


FIGURE 48. DTLZ2-20(maxG=20000).

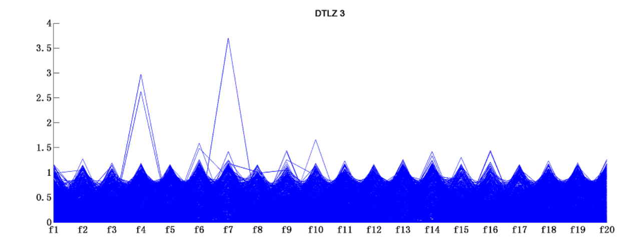


FIGURE 49. DTLZ3-20(maxG = 20000).

adjusted with larger values while leaving other parameters unchanged. Figure 45 - Figure 47 listed the results of the experiments. It can be concluded from these figures that as the parameter *maxG* increases, the algorithm will get close to

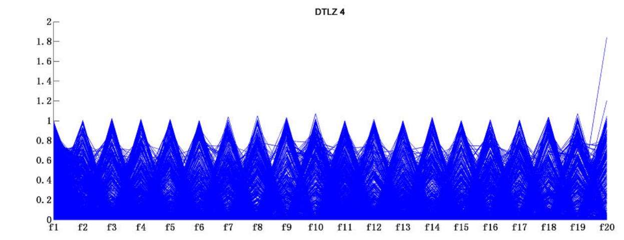


FIGURE 50. DTLZ4-20(maxG = 20000).

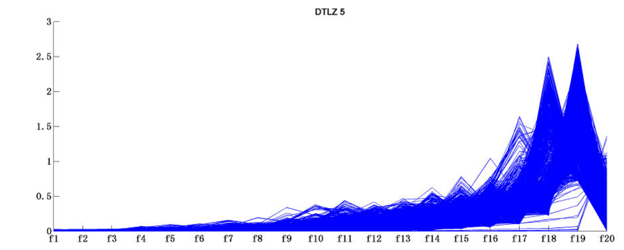


FIGURE 51. DTLZ5-20(maxG = 20000).

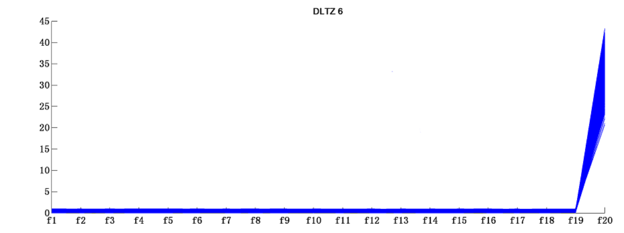


FIGURE 52. DTLZ6-20(maxG = 20000).

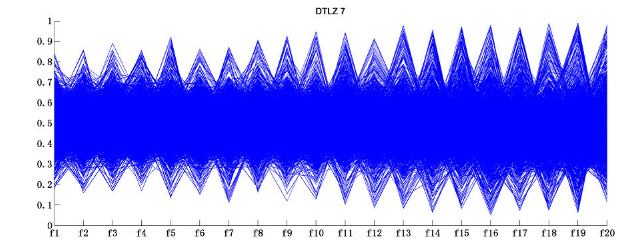


FIGURE 53. DTLZ7-20(maxG = 20000).

a better result. We chose $maxG = 20000$ for DTLZs in final. Figure 48 - Figure 53 shows the results. The performance of VITGO was also acceptable.

VI. CONCLUSION

This paper proposed a vascular invasive tumor growth optimization algorithm for multi-objective optimization called VITGO, which is implemented to search for a significant Pareto front of different MOPs. It extended the cell model of the invasive tumor growth optimization algorithm to a cell-vessel model, including the generation of vascular units, the guidance of vascular units, the update and pruning of vascular units, the generation and movement of different tumor cells, and the interactions between different individuals. A series of search schemes were creatively proposed to enable determinations elicited by the VITGO, including

1) endpoints detection and generation, which tries to identify the incomplete parts and discontinuous parts of the approximate Pareto front and generate new individuals based on the endpoints; 2) vascular guidance, which guides the movement of growing cells and dormant cells by the vascular units since the vascular units are located exactly on the current approximate Pareto front; 3) a pruning scheme, which trims out similar vascular units periodically to maintain convergence and diversity in a tiny loop; 4) unknown region searching, which inherits the invasive cells' behavior of ITGO and improves it with an opposite search and a random walk to achieve better convergence; 5) a wider search, which generates growing cells on both sides of the incomplete approximate Pareto front (guided by vascular units) to attain better diversity; 6) farthest search, which searches in just one dimension at a time to acquire better diversity and stability; and 7) a random walk with the corresponding variable that exceeds the boundary while the other variables stay the same, ensuring the diversity and stability. These schemes can indeed help to search and solve different types of MOPs successfully. The experiments and analysis showed that the proposed VITGO can outperform the other five state-of-the-art algorithms since it became closer to the true Pareto front in most cases, and the approximate Pareto front identified by the VITGO algorithm was more complete in most cases. In the future, we will expand the application of VITGO to deal with more problems encountered in real-world settings.

REFERENCES

[1] H. Seada and K. Deb, "U-NSGA-III: A unified evolutionary optimization procedure for single, multiple, and many objectives: Proof-of-principle results," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, Guimaraes, Portugal, Mar. 2015, pp. 34–49.

[2] H. Seada, M. Abouhawwash, and K. Deb, "Multiphase balance of diversity and convergence in multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 503–513, Jun. 2019.

[3] P. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 174–188, Apr. 2003.

[4] D. Tang, S. Dong, Y. Jiang, H. Li, and Y. Huang, "ITGO: Invasive tumor growth optimization algorithm," *Appl. Soft Comput.*, vol. 36, pp. 670–698, Nov. 2015.

[5] C. L. Hwang and A. Masud, *Multiple Objective Decision Making ął Methods Applications*. Berlin, Germany: Springer, 1994.

[6] J. Branke, "Finding knees in multi-objective optimization," in *Parallel Problem Solving from Nature-PPSN VIII*. Berlin, Germany: Springer, Sep. 2004, pp. 722–731.

[7] S. Iredi, D. Merkle, and M. Middendorf, "Bi-criterion optimization with multi colony ant algorithms," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, Zurich, Switzerland, Mar. 2001, pp. 359–372.

[8] J. Cheng, "Adaptive CCR-ELM with variable-length brain storm optimization algorithm for class-imbalance learning," in *Natural Computing*. Berlin, Germany: Springer, 2019.

[9] Y.-N. Guo, X. Zhang, D.-W. Gong, Z. Zhang, and J.-J. Yang, "Novel interactive preference-based multi-objective evolutionary optimization for bolt supporting networks," *IEEE Trans. Evol. Comput.*, to be published.

[10] W. Qiu, Z. Qian, and S. Lu, "Multi-objective virtual machine consolidation," in *Proc. IEEE 10th Int. Conf. Cloud Computing (CLOUD)*, Jun. 2017, pp. 270–277.

[11] D. Z. Zhu, P. L. Werner, and D. H. Werner, "Design and optimization of 3-D frequency-selective surfaces based on a multiobjective lazy ant colony optimization algorithm," *IEEE Trans. Antennas Propag.*, vol. 65, no. 12, pp. 7137–7149, Dec. 2017.

[12] Q. Kang, S. Feng, M. Zhou, A. C. Ammari, and K. Sedraoui, "Optimal load scheduling of plug-in hybrid electric vehicles via weight-aggregation multi-objective evolutionary algorithms," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2557–2568, Sep. 2017.

[13] S. Wang, S. Ali, T. Yue, and M. Liaaen, "Integrating weight assignment strategies with NSGA-II for supporting user preference multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 3, pp. 378–393, Jun. 2018.

[14] J. Schaffer, "Some experiments in machine learning using vector evaluated genetic algorithms," Ph.D. dissertation, Vanderbilt University, Nashville, TN, USA, 1984.

[15] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, Sep. 1994.

[16] K. Deb, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *Proc. Int. Conf. Parallel Problem Solving Nature*, Berlin, Heidelberg, Sep. 2000, pp. 849–858.

[17] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[18] T. Ray and K. Liew, "A swarm metaphor for multiobjective design optimization," *Eng. Optim.*, vol. 34, no. 2, pp. 141–153, Jan. 2002.

[19] C. Coello, G. Pulido, and M. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.

[20] P. Pawar, R. Rao, and R. Shankar, "Multi-objective optimization of electro-chemical machining process parameters using an artificial bee colony (abc) algorithm," *Mechtron. Intell. Manuf.*, vol. 1, pp. 165–180, Aug. 2012.

[21] H. Nasiraghdam and S. Jadid, "Optimal hybrid PV/WT/FC sizing and distribution system reconfiguration using multi-objective artificial bee colony (MOABC) algorithm," *Solar Energy*, vol. 86, no. 10, pp. 3057–3071, Oct. 2012.

[22] H. R. Hassanzadeh and M. Rouhani, "A Multi-objective Gravitational Search Algorithm," in *Proc. 2nd Int. Conf. Comput. Intell., Commun. Syst. Netw.*, Liverpool, U.K., Jul. 2010, pp. 7–12.

[23] M. Nayak, C. Nayak, and P. Rout, "Application of multi-objective teaching learning based optimization algorithm to optimal power flow problem," *Procedia Technol.*, vol. 6, no. 4, pp. 255–264, 2012.

[24] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz, and N. Pavón, "MOSFLA-MRPP: Multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning," *Eng. Appl. Artif. Intell.*, vol. 44, pp. 123–136, Sep. 2015.

[25] B. Niu, H. Wang, J. Wang, and L. Tan, "Multi-objective bacterial foraging optimization," *Neurocomputing*, vol. 116, pp. 336–345, Sep. 2013.

[26] G.-C. Luh, C.-H. Chueh, and W.-W. Liu, "MOIA: Multi-objective immune algorithm," *Eng. Optim.*, vol. 35, no. 2, pp. 143–164, Apr. 2003.

[27] Q. Lin, J. Chen, Z.-H. Zhan, W.-N. Chen, C. Coello Coello, Y. Yin, C.-M. Lin, and J. Zhang, "A hybrid evolutionary immune algorithm for multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 711–729, Oct. 2016.

[28] M. A. Tawhid and V. Savsani, "Multi-objective sine-cosine algorithm (MO-SCA) for multi-objective engineering design problems," *Neural Comput. Appl.*, vol. 31, no. S2, pp. 915–929, Feb. 2019.

[29] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. D. S. Coelho, "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Syst. Appl.*, vol. 47, pp. 106–119, Apr. 2016.

[30] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[31] Y.-N. Guo, P. Zhang, J. Cheng, C. Wang, and D. Gong, "Interval multi-objective quantum-inspired cultural algorithms," *Neural Comput. Appl.*, vol. 30, no. 3, pp. 709–722, Aug. 2018.

[32] M. A. Tawhid and V. Savsani, "A novel multi-objective optimization algorithm based on artificial algae for multi-objective engineering design problems," *Appl. Intell.*, vol. 48, no. 10, pp. 3762–3781, Oct. 2018.

[33] X. S. Yang, "Bat algorithm for multi-objective optimization," *Int. J. Bio-Inspired Comput.*, vol. 3, no. 5, pp. 267–274, 2011.

[34] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[35] Q. Lin *et al.*, "Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm," *Inf. Sci.*, vol. 339, no. 20, pp. 332–352, Apr. 2016.

[36] Z. Wang, Q. Zhang, A. Zhou, M. Gong, and L. Jiao, "Adaptive replacement strategies for MOEA/D," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 474–486, Feb. 2016.

[37] H. Nobahari and A. Bighashdel, "MOCSA: A multi-objective crow search algorithm for multi-objective optimization," in *Proc. 2nd Conf. Swarm Intell. Evol. Comput. (CSIEC)*, Kerman, Iran, Mar. 2017, pp. 60–65.

[38] Y. Guo, H. Yang, M. Chen, J. Cheng, and D. Gong, "Ensemble prediction-based dynamic robust multi-objective optimization methods," *Swarm Evol. Comput.*, vol. 48, pp. 156–171, Aug. 2019.

[39] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000.

[40] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. Congr. Evol. Comput. (CEC)*, Honolulu, HI, USA, Jun. 2003, pp. 825–830.

[41] Q. Zhang, "Multiobjective optimization test instances for the CEC 2009 special session and competition," School Comput. Sci. Electron. Eng., University of Essex, Colchester, U.K., Tech. Rep. CES-487, 2009.

[42] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Dept. Elec. Comput. Eng., Graduate School Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH, USA, Tech. Rep. TR-98-03, 1998.

[43] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

[44] J. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," M.S. thesis, Univ. Cambridge, Cambridge, MA, USA, 1995.

[45] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[46] S. Huband, "A scalable multi-objective test problem toolkit," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, Guanajuato, Mexico, 2005, pp. 280–295.

[47] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

JING ZHOU received the B.S. degree from the Department of Computer Science and Technology, South China University of Technology, in 2014, where she is currently pursuing the Ph.D. degree in computer science and technology. Her research interests include evolutionary computation, cloud computing, and optimization problems.

SHOUBIN DONG received the Ph.D. degree in electronic engineering from the University of Science and Technology of China (USTC), Hefei, China, in 1994. She was a Visiting Scholar with the School of Computer Science, Language Technology Institute, Carnegie Mellon University (CMU), Pittsburgh, USA, from 2001 to 2002. She is currently a Professor with the School of Computer Science and Engineering, South China University of Technology (SCUT), Guangzhou, China. Her research interests include high-performance computing, cloud computing, and natural language processing.

DEYU TANG received the Ph.D. degree from the School of Computer Science and Technology, South China University of Technology, China, in 2015. He is currently an Associate Professor with the School of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou, China. He has published many articles in different journals including *Information Sciences*, *Applied Soft Computing*, and *Neural Computing and Applications*. His research interests include the areas of swarm intelligence, memetic algorithms, machine learning, and bioinformatics.

XIAOFEI WU received the B.S. degree from the Department of Mathematics, South China University of Technology, in 2016, where she is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology. Her research interests include high performance computing, cloud computing, and swarm intelligence algorithms.