Journal of Optimization

Metaheuristic Optimization: Algorithmic Design and Applications

Guest Editors: Gexiang Zhang, Linqiang Pan, Ferrante Neri, Maoguo Gong, and Alberto Leporati

Metaheuristic Optimization: Algorithmic Design and Applications

Journal of Optimization

Metaheuristic Optimization: Algorithmic Design and Applications

Guest Editors: Kazuo Toda, Jorge L. Zeredo, Sae Uchida, and Vitaly Napadow

Editorial Board

Contents

Hindawi
Journal of Optimization
Volume 2017, Article ID 1053145, 2 pages
https://doi.org/10.1155/2017/1053145

Editorial

Metaheuristic Optimization: Algorithmic Design and Applications

Gexiang Zhang,[1] Linqiang Pan,[2] Ferrante Neri,[3] Maoguo Gong,[4] and Alberto Leporati[5]

[1] *School of Electrical Engineering, Southwest Jiaotong University, Chengdu, China*
[2] *School of Automation, Huazhong University of Science and Technology, Wuhan, China*
[3] *Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester, UK*
[4] *Key Laboratory of Intelligent Perception and Image Understanding, Xidian University, Xi'an, China*
[5] *Department of Informatics, Systems and Communication, Università degli Studi di Milano-Bicocca, Milano, Italy*

Correspondence should be addressed to Gexiang Zhang; gexiangzhang@gmail.com

Received 15 June 2017; Accepted 15 June 2017; Published 5 September 2017

This special issue focuses on algorithmic design and applications of metaheuristic optimization algorithms. More specifically, we aim at offering some examples of informed design where the problem features are taken into account during the design of the metaheuristic algorithm. Two categories of approaches are here identified:

(i) Algorithms that perform changes during the run on the basis of the success of the components, thus adapting to the problem

(ii) Algorithms that are designed after a thorough problem examination

Some of the papers contained in this special issue mainly focus on algorithmic aspects by modelling and conceptualizing some problem features and/or design approaches. Other papers propose domain specific implementations. However, due to the nature of the subject, the two categories are not clearly separated and, naturally, research in the field tends to lay on both areas.

The paper by E. B. de Moraes Barbosa and E. L. Franca Senne makes use of statistical and artificial intelligence methods to fine-tune the parameters of algorithms. The proposed method, namely, Heuristic Oriented Racing Algorithm (HORA), explores the parameter space to search for a good configuration and thus adapts the algorithm to each problem under consideration.

The paper by S. Alharbi and I. Venkat proposes an evolutionary approach for a problem of chess puzzles, that is, the Minimum Dominating Set of Queens Problem. To tackle this problem, S. Alharbi and I. Venkat propose a simplistic Genetic Algorithm with an encoding and basic algebra especially suited to the problem.

The paper by Y. Lei and J. Shi focuses on a specific class of combinatorial problems, namely, examination timetabling problems. In this paper, the problem is stated as a two-objective optimization problem and solved by a tailored memetic algorithm. The proposed algorithm uses a previously designed special crossover operator and two local search algorithms combined with a diversity-keeping strategy used to ensure that the nondominated set contains enough solutions.

The paper by Y. Li et al. proposes a novel sophisticated metaheuristic belonging to the class of particle swarm optimizers. The proposed framework is thought to address complex highly multivariate problems and makes use of the MapReduce programming model to decompose the algorithmic operations and allow a natural parallelization.

The paper by K. Li et al. focuses on an image processing problem with a specific reference to agriculture. The proposed system is composed of an evolutionary metaheuristic and two image processing algorithms. The goal of the proposed system is to detect spots on plants due to maize diseases and insect pests.

The paper by N. Elkhani and R. C. Muniyandi proposes a novel hybridization within multiobjective particle swarm optimization frameworks. This hybridization makes use of a parallelized version of membrane computing to perform the selection within the metaheuristic. The multiobjective particle swarm optimizer is used to identify marker genes for cancer classification.

The last two papers focus on tailored algorithmic design for military applications. The paper by G. Peng et al. proposes a hybrid/memetic discrete particle swarm optimizer to solve a complex real-world military problem. The problem consisting of assigning the targets to the weapons available to maximise the efficiency of an air attack leads to a combinatorial problem which is at the same time multiobjective, dynamic, and constrained. The proposed algorithm is composed of multiple parts including a Cauchy moving operator and local search as well as a repairing algorithm to solve situations with too few feasible solutions.

The paper by K. Zhang et al. studies the unmanned combat aerial vehicle path planning and proposes a new metaheuristic approach which integrates within its structure a domain specific graph theory model. The obtained trajectory of the unmanned aerial vehicle designed by the proposed approach smoothly avoids those areas controlled by radars.

Gexiang Zhang
Linqiang Pan
Ferrante Neri
Maoguo Gong
Alberto Leporati

Hindawi
Journal of Optimization
Volume 2017, Article ID 9710719, 11 pages
https://doi.org/10.1155/2017/9710719

Research Article

Robust Circle Detection Using Harmony Search

Jaco Fourie

Lincoln Agritech Ltd., Lincoln, New Zealand

Correspondence should be addressed to Jaco Fourie; jacofourie@gmail.com

Received 10 January 2017; Accepted 24 May 2017; Published 2 August 2017

Academic Editor: Ferrante Neri

Automatic circle detection is an important element of many image processing algorithms. Traditionally the Hough transform has been used to find circular objects in images but more modern approaches that make use of heuristic optimisation techniques have been developed. These are often used in large complex images where the presence of noise or limited computational resources make the Hough transform impractical. Previous research on the use of the Harmony Search (HS) in circle detection showed that HS is an attractive alternative to many of the modern circle detectors based on heuristic optimisers like genetic algorithms and simulated annealing. We propose improvements to this work that enables our algorithm to robustly find multiple circles in larger data sets and still work on realistic images that are heavily corrupted by noisy edges.

1. Introduction

Circle detection is a key element in the larger field of automatic extraction and identification of geometric shapes from digital images. The ability to identify and extract shapes from images has many applications in industry and agriculture. This is because simple geometric shapes are common in man-made environments and are often used in symbols that only have meaning when properly identified.

Circle and ellipse are particularly common shapes to identify due to their application in biological cell tracking, automated mechanical parts inspection, and biometrics (iris detection). Circle detection is traditionally done using the circle Hough transform (CHT) [1, 2]. The CHT algorithm has been used for circle detection for over 30 years and much research has been done to improve the original algorithm. Most of the research focused on the main limitations of CHT, namely, high computational and storage requirements. Proposed improvements include probabilistic, randomised, and fuzzy adaptations of the original algorithm [3–5]. These improvements all aim to decrease the computational complexity of CHT and some also address the accuracy of the algorithm in the presence of noise [6].

With the dramatic increase in computational power, more researchers have begun investigating metaheuristic optimisation algorithms as possible approaches to more robust and accurate circle detection. These algorithms are often biologically inspired like the genetic algorithm and the particle swarm optimisation algorithm. They are usually computationally expensive but have the advantage of not making any assumptions about the objective function or the amount and type of noise that may be present. This usually implies more robust and accurate performance on noisy or ambiguous data.

Both genetic algorithms and particle swarm optimisation algorithms have been successfully used in circle detection along with other metaheuristics like simulated annealing and bacterial foraging optimisation algorithm [7–10]. We propose the use of another biologically inspired metaheuristic optimiser, namely, the Harmony Search (HS) algorithm [11].

Harmony Search is a musical improvisation-inspired metaheuristic optimisation method originally developed for the design of pipeline network systems and published in 2001 [12]. Since its publication, it has been used in many engineering and scientific fields, including computer science, as a robust optimisation technique [13].

2. Theory

In this section, we introduce circle detection in digital images and give an overview of the circle Hough transform (CHT).

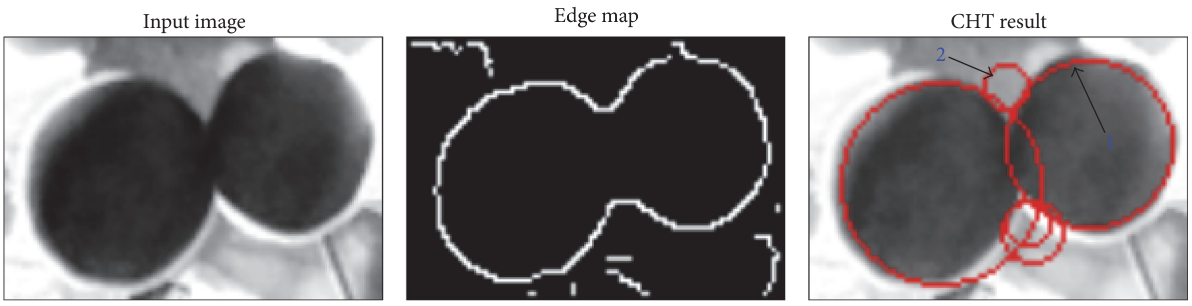


Figure 1: An example of the CHT detecting multiple circles in an image. The largest and second-largest maxima in the accumulator matrix is labelled as 1 and 2, respectively.

Being the most popular method of circle detection, the CHT gives us a starting point to investigate circle detection methods with specific focus on their limitations and accuracy under challenging conditions.

We pay specific attention to circle detection methods that interpret the problem as a fitness function that has to be optimised and that make use of metaheuristics to solve the optimisation problem. We then introduce Harmony Search (HS) as a heuristic optimiser and mention some of the variants of the original algorithm that attempt to address the limitations of traditional HS.

2.1. Classical Hough Transform Based Circle Detection. As previously mentioned, the classical way to do circle detection is using the circular Hough transform. This method assumes that the edge pixels of the image have already been identified using one of the many edge detection methods, for example, the Canny edge detector [14]. The collection of edge pixels, called the edge map, is then processed to identify which of the pixels is part of a circle by transforming the edge pixels into a 3D parameter matrix called the Hough parameter space. The maximum point in the parameter space corresponds to the parameters of a circle that has the most edge pixels on its perimeter.

Consider a circle in 2D space represented by the equation

$$(x-a)^2 + (y-b)^2 = r^2, \quad (1)$$

where (a, b) represents the centre of the circle and r is the radius. From this equation, we can map each pixel (x, y) to a conic surface in the 3D parameters space (a, b, r) that represents all possible circles that the pixel can form a part of. By discretising and limiting the parameter space based on the original image dimensions, a 3D accumulator matrix is formed. The location of the maxima in this matrix represents the circles detected by the CHT.

Though this simple approach is somewhat robust to noise and occlusion, the main limitation is in the computational and space requirements. As the 3D accumulator matrix grows cubically with the size of the image, the memory required to store this matrix and the time spent creating it quickly become prohibitive. This is especially true when circle location accuracy demands that the parameter space be quantised into accumulator cells that are as small as possible or when the space needs to be enlarged to beyond the size of the image to allow for partial circle detection that may have centres that fall outside the image bounds. In Figure 1, we show a common result of the CHT applied to a noisy image. In this example, we did not apply a set threshold to determine which maxima in the accumulator should be selected as valid circles but instead picked the top 5 largest values. In this way, we highlight the false circles that are often detected in real image examples. We label the strongest circle in the accumulator as 1 and the second strongest as 2 to show that simply picking the strongest circles in the accumulator will not solve the false circle problem even when the exact number of circles to be detected is previously known. If we only picked the strongest two circles in this example, the large circle on the left would not be detected while the small circle in the middle is chosen instead.

2.2. Circle Detection Using Metaheuristics. In order to address some of the limitations of the CHT, researchers have extended and adapted the CHT in various ways to improve the accuracy but mainly to improve the performance and improve the large memory requirements that classical CHT has. Some examples of CHT extensions include the randomised Hough transform [5] and the fuzzy Hough transform [16]. Both these approaches attempt to improve performance by either decreasing the memory and computational load, as with the randomised Hough transform, or by generalising the algorithm to improve robustness as with the fuzzy Hough transform.

More recently researchers have also begun to apply evolutionary algorithms to circle detection. Examples include genetic algorithm based approaches [8], differential evolution based approaches [17], and bacterial foraging algorithm based approaches [10]. These methods all have the common feature that they turn the circle detection problem into a mathematical optimisation problem. It is important to note however that no assumptions are being made regarding the amount or type of noise in the search space of solutions for this optimisation problem. The shape of the search space is also not assumed and can be discrete and highly multimodal and can contain discontinuities that would make it difficult for traditional gradient-based optimisation techniques to

```
input: A fitness function $f$ for scoring the solutions along with
       an upper bound (UB) and lower bound (LB) for each
       dimension of the $M$-dimensional solution. A function,
       $U(l, u)$, for sampling random numbers between $l$ and $u$
       from a uniform distribution is also required.
output: The best solution $x_{\text{best}} = [x_0, \dots, x_M]$
(1)  Initialise the HM with random entries
(2)  foreach $n \in [1, \text{NI}]$ do
(3)      foreach $i \in [1, M]$ do
(4)          if $U(0,1) > \text{HMCR}$ then
(5)              $x_i' \longleftarrow \text{LB}_i + r \times (\text{UB}_i - \text{LB}_i)$
(6)          else
(7)              $x_i' \longleftarrow x_i^j$
(8)              if $U(0,1) \le \text{PAR}$ then
(9)                  $x_i' \longleftarrow x_i' + r \times \text{FW}$
(10)             end
(11)         end
(12)         $x_{\text{new}}[i] \longleftarrow x_i'$
(13)     end
(14)     select the lowest scoring solution $x_{\text{worst}}$ from the HM
(15)     if $f(x_{\text{new}}) > f(x_{\text{worst}})$ then
(16)         remove $x_{\text{worst}}$ from HM and replace with $x_{\text{new}}$
(17) end
(18) select the highest scoring solution $x_{\text{best}}$ from the HM and return it
```

Algorithm 1: The Harmony Search algorithm.

successfully traverse and find a solution. This is why heuristic optimisers, like the family of evolutionary algorithms, are used since they typically make no assumptions about the fitness function that describe the search space.

2.3. Harmony Search Optimisation. The (HS) algorithm was inspired by the way musicians collectively and cooperatively improve harmonies in certain genres of music, especially Jazz [11, 12]. Like evolutionary algorithms, it generates a population of candidate solutions and then iteratively improves on the candidate population by adding and removing individual candidates. Unlike most evolutionary algorithms, it does not update the entire solution population at every iteration but only changes one individual at a time. This maintains critical diversity in the population for more iterations making the early convergence to local optima much less likely.

Consider the analogy of a Jazz trio during an improvisation session. Each member of the trio plays a note on their instrument selected from a limited range of notes which may be different for each member. Together the three notes form a harmony that may or may not be pleasing to the audience (or the trio itself). For the sake of this analogy, the aim of the trio is to find the 3-note harmony that is most pleasing to the audience.

In our analogy, the 3-note harmony is the solution to an optimisation problem of finding the most pleasing harmony as judged by the audience. The search space of possible solutions is 3-dimensional in this case and is bounded by the notes playable by each musician. For example, if each musician could only choose from 5 different notes, the search of all possible solutions would only contain $5 \times 5 \times 5 = 125$ candidates. The fitness function that evaluates each candidate harmony is represented by the audience.

Improvisation of new harmonies is an iterative process involving each member of the trio choosing a note to play based on what resulted in pleasing harmonies during previous iterations. In this way, the improvisation process uses a memory of good solutions that is constantly updated and improved with new entries as better harmonies are discovered. The improvisation process continues until either a harmony of sufficiently high quality is produced or the maximum number of iterations has been reached.

This HS analogy is summarised as follows. The performance of HS is controlled by four parameters: the harmony memory size (HMS), the harmony memory consideration rate (HMCR), the fret width (FW), and the maximum iterations (NI). The HMS is the number of *good solutions* that are stored at any one time in the harmony memory (HM) and used for future iterations. The HMCR determines how likely the memory is considered as opposed to randomly selecting a value during improvisation and the FW determines the amount a value from the memory may be adjusted during improvisation. The NI puts a limit on how long the process may continue by limiting the number of improvisation attempts. HS can be described in pseudo code as shown in Algorithm 1.

Algorithm 1 describes the simplest form of HS and while this form has been used in many practical applications, much research has gone into improving HS to be more robust, accurate, and computationally efficient. Over the last

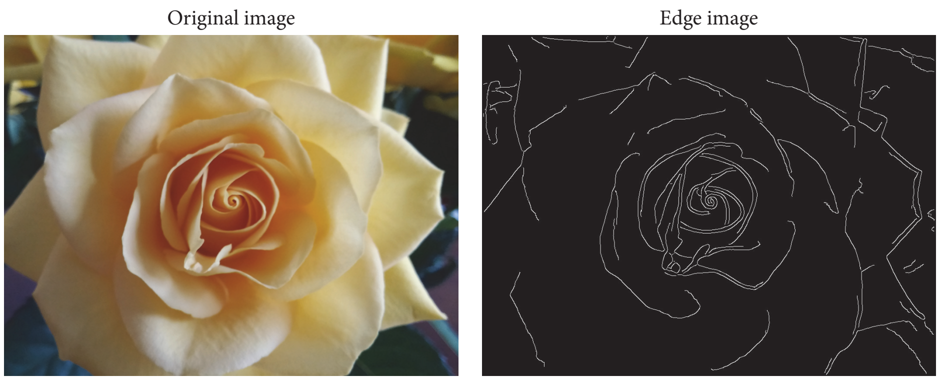


Figure 2: The Canny edge detector is used to extract the edge map.

decade, many new variations of HS have been developed. The majority aim to address a limitation of HS arising from the choice of the HMS, HMCR, PAR, and FW parameters. The optimal choice of these parameters depends on the shape of the search space which in most optimisation problems is unknown. Sensible default values for each of these parameters can be found in the first published results [12] and will result in good performance in most cases but can usually be improved considerably. Most of the modern HS variants remove one or more of these parameters from the algorithm completely or replace them with other parameters that are less sensitive to the shape of the search space or are easier to choose. These improved HS variants have been used in many diverse applications and the interested reader is referred to [18–23], for examples, many HS variants that are used in computer vision, vehicle routing, music composition, solving Sudoku, and various engineering disciplines.

3. Circle Detection as an Optimisation Problem

The main contribution of this work is based on applying a variation of HS to the circle detection problem. This approach was pioneered by Cuevas et al. and their work forms the basis of our proposed algorithm [24]. This section starts with a review of their work followed by a description of our contribution and how it relates to the original pioneering work.

In their work, Cuevas et al. describe circles in an image by the well-known second-degree equation introduced in Section 2.1 as (1). Images are preprocessed to extract object outlines or edges in the image by using the Canny edge detector [14]. This identifies the pixels in the image that should be tested as potentially being part of circles in the image. In Figure 2, an image pair representing the original image and its edges is shown to illustrate the way the Canny edge detector is used to identify circular edge candidates.

Detection of circles in the image is interpreted as an optimisation problem with each edge pixel in the edge map considered to be part of the search space of possible solutions. A candidate solution, that is, a circle in the image, is represented as three distinct edge points since 3 points on a plane are required to define a unique circle. Candidate solutions are scored using a fitness function defined as the number of pixels in the edge map that support the circle described by the 3 points in the candidate solution. The more edge pixels that lie on the circle described by the candidate solution, the higher its fitness.

3.1. Circle Detection Using Harmony Search

3.1. Circle Detection Using Harmony Search. Once the circle detection problem is defined as optimisation of a finite search space and an appropriate fitness function is provided to compare candidate solutions, HS can be used to find the best solution in the search space. Since the search space is simply a vector of all the edge pixels, candidate solutions can be coded in the HM as three indices into the vector of edge points. Let $P = \{p_1, p_2, p_3, \ldots, p_E\}$ be the set of E edge points, that is, the search space. Then a candidate solution in the HM is a 3-dimensional vector $x_c = [p_i, p_j, p_k]$, where $0 \leq i, j, k < E$. This means that we now have a standard 3-dimensional optimisation problem with a search space limited in all three dimensions by the number of edge pixels.

All that remains to be defined is the fitness function. Let $p_i = [x_i, y_i]$, $p_j = [x_j, y_j]$, and $p_k = [x_k, y_k]$ be 3 edge pixels that together define a circle and a solution vector. The radius, r, and the centre, x_0, y_0 of this circle can be calculated as follows. Let

$$X_{ji} = x_j - x_i,$$

$$X_{ki} = x_k - x_i,$$

$$Y_{ji} = y_j - y_i,$$

$$Y_{ki} = y_k - y_i,$$

$$A_0 = x_j^2 + y_j^2 - x_i^2 - y_i^2,$$

$$A_1 = x_k^2 + y_k^2 - x_i^2 - y_i^2,$$

$$A = \begin{bmatrix} A_0 & 2Y_{ji} \\ A_1 & 2Y_{ki} \end{bmatrix},$$

$$B = \begin{bmatrix} 2X_{ji} & A_0 \\ 2X_{ki} & A_1 \end{bmatrix},$$

$$D = 4\left(X_{ji}Y_{ki} - X_{ki}Y_{ji}\right); \tag{2}$$

then,

$$x_0 = \frac{\det(A)}{D},$$
$$y_0 = \frac{\det(B)}{D}, \tag{3}$$
$$r = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2},$$

where det denotes the matrix determinant.

With the circle centre and radius calculated, we can generate a list of pixels that lie on the perimeter of that circle $S = \{s_0, s_1, \ldots, s_C\}$ by using the midpoint circle algorithm (MCA) [25]. The fitness function f is then defined as follows. Let

$$E(P, s) = \begin{cases} 1 & \text{if } s \text{ is one of the edge points in } P \\ 0 & \text{if } s \text{ is not an edge point in } P \end{cases} \tag{4}$$

$$f(S) = \left[\frac{\sum_{i=0}^{C} E(P, s_i)}{C}\right]. \tag{5}$$

In other words, $f(S)$ simply counts the number of points on the perimeter of S that are also edge pixels from the set of all edge pixels P.

In this implementation, the detector will only find one circle but Cuevas et al. expand this idea to allow for multiple circle detection by removing any circles found from the edge map and then iteratively rerunning the HS optimiser until no more circles can be found of sufficient quality or until some threshold has been reached. For further details as well as experimental results from this circle detector, the reader is referred to the original publication by Cuevas et al. [24].

Before describing the proposed algorithm that forms the main contribution of this work, it is valuable to investigate the size and shape of the search space that HS optimises over. HS was designed to be robust enough to optimise over almost any kind of search space even those that are highly discontinuous, multimodal, and not differentiable. However, like most heuristic optimisers, HS relies to some extent on the existence of a *basin of attraction* around local optima in the search space, even if it is small compared to the size of the search space [26]. This means that solutions close to each other (based on some distance measure) in the search space are expected to have similar fitness values. In situations where this is not the case, HS may still be able to find a good solution but its performance will suffer greatly and could even degrade into a randomised search. As an example, refer to Figure 3 that shows a 3-dimensional representation of the Ackley function [27] which is often used to benchmark optimisation algorithms. Notice that it is highly multimodal

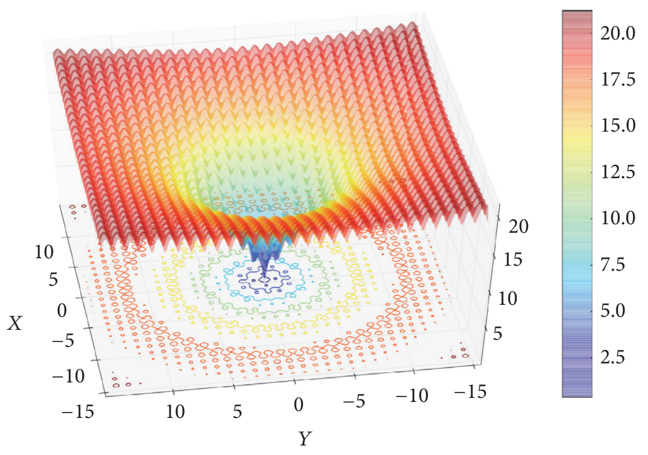


FIGURE 3: Illustrating local minima and basins of attraction in the Ackley function.

and has many local minima but only one global minimum. Also notice that even though they may be small, each local minimum is surrounded by a small neighbourhood of points that have similar function values and represents the basin of attraction around that local minimum.

Even on a small 1-megapixel image, the number of edge points can easily be in in the order of 10,000. Since any three of these points can potentially form a circle, the size of the search space that HS optimises over is $10{,}000^3 = 10^{12}$ potential solutions. This large search space is typically highly multimodal since a dense edge map will have many ways in which pixels from different areas can be combined into circular or approximately circular shapes. These characteristics on their own already make for a challenging optimisation problem but a greater problem lies in the shape of the search space.

Consider the way that a solution vector is defined from the list of edge pixels. Instead of storing the pixel location in, for example, Cartesian coordinates, a solution vector is three indices into the list of edge pixels which inherently discards the relative distance information between pixels. This decision was made for practical reasons since a more complicated encoding of the solution would mean that the HS improvisation step would need to be adapted so that improvisation based on previous good solutions would still likely result in better ones. However, one result of this encoding is that edge pixels that are far from each other (large difference between index values) in the list of edge pixels may be close together on the image plane in a Euclidean sense. This means that edge pixels that are neighbours and therefore likely belong to the same circle are not neighbours in the search space. Consider the example of Figure 4.

Part of the edge list from an example image is displayed as an edge map in Figure 4. One can easily identify two separate circular objects where each contains multiple edge pixels which could be considered as belonging to a circle. However, if one encodes this edge map by starting from the top row of the image and recording the edge pixel locations row-by-row, then edge pixels that are in the same row will be neighbours in the edge list as well as those which are

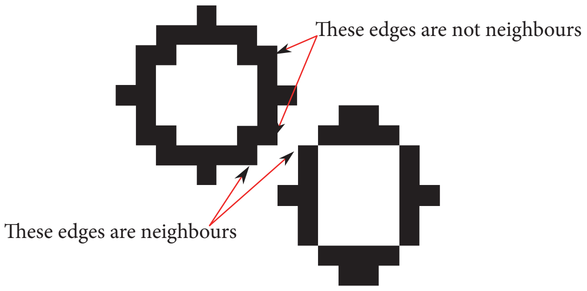


Figure 4: Edge pixels can be close together on the image plane and belong to the same circle but may be far apart in the list of edge pixels that define the search space.

separated by rows containing only nonedge pixels. However, pixels on different rows are not neighbours in the search space even when they are close together on the image plane. Figure 4 highlights edge pixels that are close together on the image plane and belong to the same circle but will not be in the same neighbourhood in the search space. Similarly, it also shows two pixels that would be right next to each other in the search space but belong to completely different objects. Note that as long as there are no nonedge pixels between those two pixels, they will always be next to each other in the search space no matter how large the distance is between them on the image plane.

This is a serious problem for the HS algorithm as it means that a basin of attraction in the search space will not necessarily contain the edge pixels required to find the circle that it may represent. Also, small changes to a solution in the HM (pitch adjustment in HS) may cause edge pixels from a completely different part of the image to be included in a new improvisation that should be based on an already discovered circle.

In small images (few edge pixels), this limitation in the encoding of the HM may simply cause slow convergence while still resulting in correct identified circle. However, in our experiments, this approach fails on large complex images that contain many thousands of edge pixels and potentially dozens of circles that we aim to identify. The edge map in Figure 5 is an example of a 1.4 megapixel image that contains 68,094 edge pixels. This means the search space of possible solutions contains 3.15×10^{14} distinct candidates. This large search space and the previously mentioned limitations are the two primary reasons why HS failed in our experiments with larger, more complex images.

3.2. The Proposed Algorithm. We propose two main changes to the approach discussed in the previous section. Firstly, the size of the search space needs to be limited and it needs to be organised so that basins of attraction surround most local optima and that neighbouring solutions are likely to belong to the same circle. Secondly, instead of using the standard HS algorithm, we use one of the modern variants (see Section 2.3) that is designed to be more robust in discontinuous, highly multimodal search spaces.

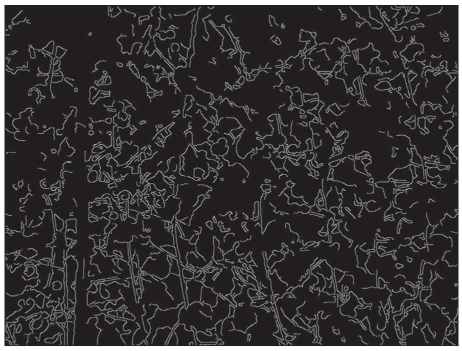


Figure 5: This complex edge map was constructed from a 1386 × 1039 image and contains 68,094 edge pixels.

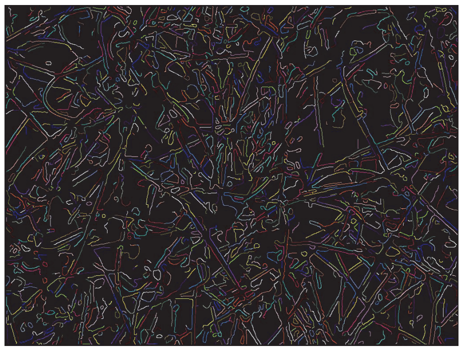


Figure 6: The edge map shown here has been organised into connected neighbourhoods called ribbons.

In order to both limit the size of the search space and improve the shape, we sort the list of edge pixels into connected components and focus our search around a single neighbourhood of edge pixels that are 8-connected neighbours on the image plane. Pixels are 8-connected when they touch horizontally, vertically, or diagonally on the image plane. We call these connected collections of edge pixels *ribbons*. They are combined to form the ribbon list that replaces the edge list to describe the search space. In Figure 6, the edge list is colour coded to illustrate how it may be sorted into multiple ribbons that can be individually considered.

Because we use the Canny edge detector to build our edge map, the edges that are connected are already grouped during the hysteresis thresholding step (see [14]). Therefore, we do not have to separately sort the edge list into separate ribbons. Additionally, the search space is further reduced by removing all the ribbons that are considered too small to form a significant part of any circle. Note that even though these ribbons are removed from the search space for the improvisation of new solutions, the edge pixels inside them are still considered when scoring other solutions. All edge pixels, even those from other ribbons, affect the fitness score of any potential solution. This allows for circles that span multiple ribbons to be included as viable candidates.

We also split long ribbons into smaller ones so that multiple circles may be found on long connected edges. Given that our search space now only consists of short simple ribbons, we can simplify the search space further by defining the circle as only two points and implying the location of the third. Given two points on a single ribbon, we make the assumption that these points represent the start and end points of a circle arc that forms part of the ribbon. The third point required to identify the circle is implied as being half way between the start and end point on the ribbon. This significantly reduces the size of the search space and simplifies the optimisation problem by changing the 3-dimensional problem into a 2-dimensional one.

In addition to the previous modification of the search space, we optionally normalise the fitness function slightly to favour larger circles. Normally the fitness function is normalised by dividing the number of edge pixels that fall on the candidate circle by the circle circumference as defined in (5). This tends to favour smaller circles since fewer pixels are required to define a given ratio of the total circle circumference. In some images, this results in larger circles misidentified as small ones. In these cases, it is better to normalise with a constant factor based on the circumference of the largest circle one expects to find in the image.

The minimum and maximum circles sizes are user parameters that allow the algorithm to focus only on a specific range of circle sizes and help to filter out circles that could be considered as false positives. In our alternate definition of the fitness function, we normalise by the circumference of the maximum circle size so that $f(S)$ is defined as

$$f(S) = \left[\frac{\sum_{i=0}^{C} E(P, s_i)}{C_{\max}} \right], \tag{6}$$

where $C_{\max}$ is circumference of the maximum circle size expected. This is an optional modification of the fitness function and is only used when the application would benefit from a detector that favours larger circles.

In addition to simplifying the search space, we also propose the use of a modern variant of HS that is specifically designed to be more robust when the search space is discrete and discontinuous, for example, labelling problems and integer optimisation. The CHS algorithm is based on HS and was designed to solve the labelling problem associated with the blind deconvolution problem of binary images [28]. Our problem is similar in that our search space is also discrete and discontinuous but we do not need the special improvisation step used in CHS called the *cadenza* operation as this is specific to image deconvolution. Instead we use the standard HS improvisation method. However, we also split the HM into separate islands with special border members like CHS in order to maintain as much diversity in the HM as possible. For more details on CHS and optimising discrete and discontinuous search spaces, see the original CHS paper [28].

Our proposed detection method can be summarised with the following steps:

(1) Use the Canny edge detector to extract the edge map from the source image and sort the edges into connected components called ribbons.

(2) Filter out ribbons that are too short and split ribbons that are too long to both limit the search space and ensure that multiple circles on long continuous edges are not missed.

(3) Use the adapted CHS algorithm to find the best candidate circle on each ribbon in the filtered list.

4. Experimental Results

We tested our method using both synthetic images and images captured using normal consumer grade cameras of real objects. Using synthetic images, we can generate scenes with an arbitrary amount of noise and can partially occlude any matches by an arbitrary amount. This allows us to test the robustness and accuracy of the algorithm in a more quantitative way rather than using images captured from real scenes. Conversely, the nonsynthetic images are more realistic and allow us to show potential results in practical applications.

4.1. Synthetic Images. In Figure 7, we show a synthetic image designed to test the robustness of the circle detector. The image contains circle segments of various sizes representing various ratios of a complete circle. Some of the segments are corrupted with rectangular overlays that obscure some edges in the edge map with conflicting edges that do not belong to any circle. Other edges are corrupted with a random pixel offset in both x and y directions causing a wavy arc that has few pixels actually landing on the circle solution. This is made clear in Figure 8 which shows the edge map and highlights how the corruption of the circle arcs results in edges that offer little evidence that they are part of a circle.

In Figure 9, we add Gaussian noise to the test image and redetect the circles using the same algorithm and the same parameter settings. As more noise is added, the edge map becomes more corrupted with edges that do not belong to any circle and overlaps with edges that are part of a circle. Furthermore, edges that were easily identified as part of a circle often get broken into smaller edges which cannot easily be identified by the addition of the noise. We see this effect in Figure 9 particularly in the 20% and 40% noise images.

It is worth noting that, even with 40% Gaussian noise added, most of the circles are still identified correctly. Part of the reason for this robustness is the relatively low quality threshold which is indicative of found circles (see (5)). However, a low quality threshold poses a risk that some edges may be incorrectly identified as circles. We see this happening several times in the noisy images of Figure 9 when small circles are detected on the edges of what should be identified as a much larger circle. This situation seems to indicate that (6) may be a better alternative to the original fitness function since it will suppress the small circles, but this was not the case. In images with greater variation in the expected size of the circles, (5) fails to accurately describe what sufficient

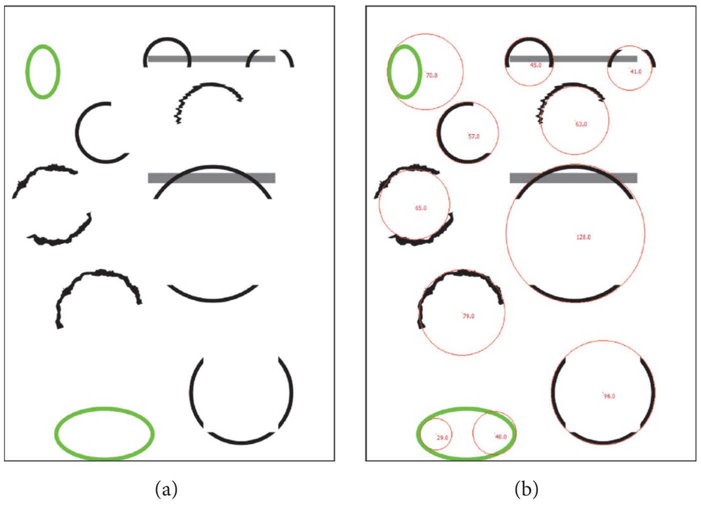


Figure 7: We use this synthetic test image to test the robustness of the proposed circle detector under various conditions. The red circles in (b) indicate the solutions found by the detector.

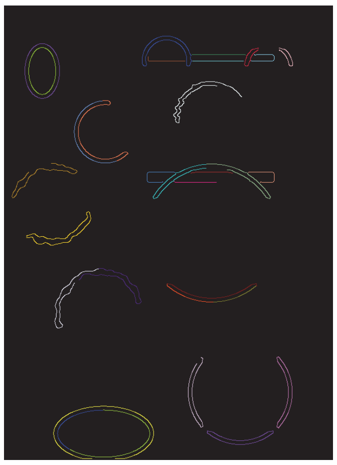


Figure 8: This is the edge map of the synthetic image of Figure 7.

evidence for a circle should be. This is especially true when there is a high degree of noise in the image.

4.2. Real Images. We used a selection of images from the Caltech Computer Vision Group archive [15] to test our detector on real images that contain circles or objects that are roughly circular or elliptical. In Figure 10, we show four images that each highlight different features of the algorithm.

Figure 10(a) is a good example of using the detector to find only circles of a specific size. We restricted the detector to circles with a minimum radius of 240 and a maximum of 250 pixels. The detector found all the coins with the smallest radius at 244 and the largest at 247. Figure 10(b) is an example of using the detector to match elliptical objects. Due to the perspective of the image, the bowl is a large ellipse in the edge map but with an appropriate quality threshold the circle detector can approximate the bowl's shape with a circle. In Figure 10(c), we used a low quality threshold and a large range of circle sizes. This allowed at least one noncircular edge to be identified as a circle. In a similar example on Figure 10(d), we used the same circle sizes but a much stricter quality threshold resulting in fewer circles detected but also no false positives. Note that in all the results from these images, we filtered out the detected circles that overlap with other circles that have a higher fitness score.

Our original intention with this work was to address some limitations of the original work done by Cuevas et al. [24] and show that a HS based circle detector can be used in practical applications to detect multiple circles in complex images. One such practical example is automated fruit counting as shown in Figures 11 and 12. In both these examples, the majority of the fruit was correctly identified making it useful as part of a yield estimation system.

In another practical example, the aim is to count cells or other microscopic particles in an image captured from a microscope. In this example (Figure 13), the detector was able to find 3140 circular particles in one image. This is also a good example of where the approach used by Cuevas et al. [24] will likely give poor results since most of the circles share edges since almost all of them touch. In their original work, as mentioned in Section 3.1, Cuevas et al. can find multiple circles by removing the edges of a circle already detected from the edge map before searching for the next one. In situations like those seen in Figure 13, this will likely cause edges from neighbouring circles to be removed as well as making it likely that many circles will not be successfully detected since most of their edges were already removed from the edge map in previous iterations.

5. Conclusion

Based on the original research by Cuevas et al. [24], we propose an alternative approach to using the Harmon Search algorithm for circle detection. Our approach differs in the way that the optimisation problem is structured so that we

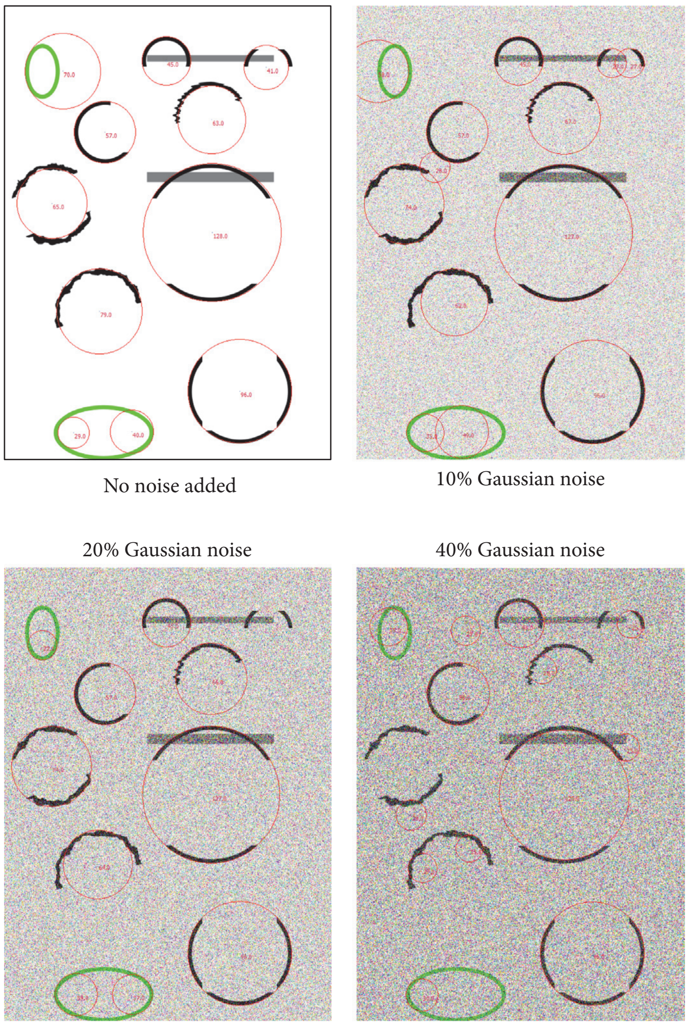


Figure 9: Gaussian noise is added to the synthetic test images.

sort the edges into connected components called ribbons which becomes input to the detector in isolation of the other edges in the edge map. This both improves the shape of the search space making it easier to optimise and allows one to accurately find multiple circles in a single image. Instead of using the original Harmony Search optimiser we use a variant based on one of the modern improvements to Harmony Search that is designed for search spaces similar to what we find in the circle detection problem.

We showed through various examples that our approach leads to accurate results that are robust to noise as well as incomplete or imperfect circles. We also highlighted some practical applications that demonstrate our detector's ability to find thousands of circles in images that will be challenging to find multiple circles using the approach used by Cuevas et al. [24]. However, it should be noted that our approach does not improve on the results of Cuevas et al. [24] in all examples. Instead it should be thought of as an alternative that could be more effective in specific applications.

5.1. Future Work. The main aim of this paper is to build on the research done by Cuevas et al. [24] and give an alternate approach to circle detection using Harmony Search. However, there are several ways to potentially improve what we started. One example is to improve the robustness to imperfect circles by expanding the algorithm to detect ellipses instead of circles. The same set of connected and sorted edges can be used but instead of 3 points, 5 points are now needed to define a unique ellipse. This would turn the optimisation problem into a 5-dimensional problem that can be optimised using the same Harmony Search variant used in this paper.

Another less complex improvement involves regularising the fitness function of (5) by adding various penalty terms. For example, the presence of extraneous edges inside a circle candidate or edges that cross the circle's perimeter could be made into a penalty term that discourages the detector to find circles in noisy areas of the image making it less likely to be confused by a dense edge map.

The fitness function could also be improved by taking into account the gradient of the edges when deciding which edges

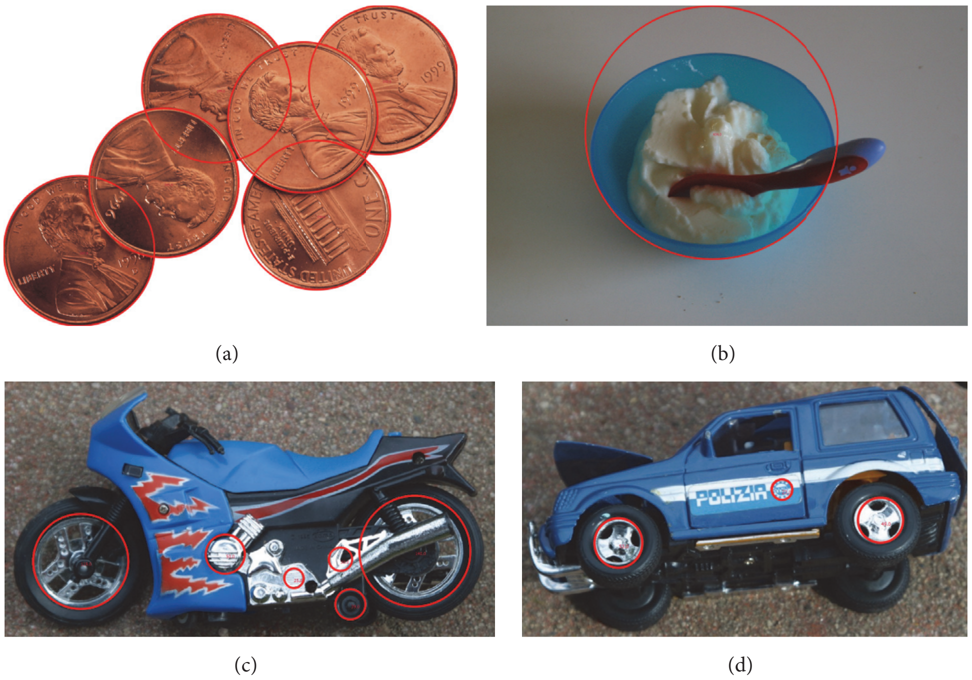


Figure 10: Various test images that highlight different aspects of the circle detector. Figures (b), (c), and (d) come from the image archive of the Computer Vision Group at Caltech [15] and (a) comes from the free clip art repository at http://pngimg.com.

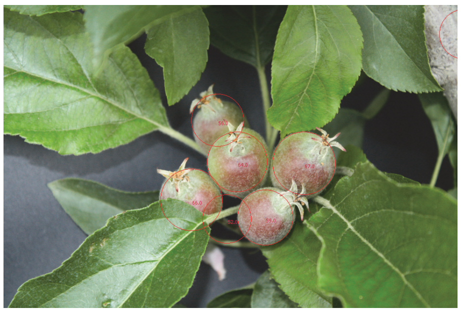


Figure 11: In this example, the circle detector is used to count fruit in order to estimate yield.

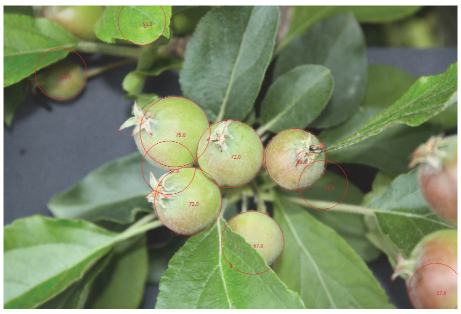


Figure 12: Another example of our circle detector used to count fruit.

would be considered evidence for a particular candidate circle. One would expect that an edge that forms part of a real circle would have a similar gradient as the candidate circle at

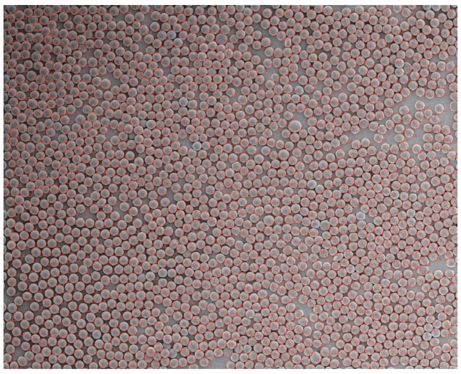


Figure 13: In this example, the circle detector is used to count cells in an image captured from a microscope.

the point of overlap and if the gradient differs by too much the edge can safely be rejected as evidence of a circle.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially funded through a grant from the New Zealand Ministry of Business Innovation and Employment under the research grant, Optimum N Nitrogen Sensing and Management (Contract no. CONT29854BITR_LVL).

References

[1] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[2] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer Vision, Graphics and Image Processing*, vol. 44, no. 1, pp. 87–116, 1988.

[3] J. Han, L. Koczy, and T. Poston, "Fuzzy hough transform," in *Proceedings of the 2nd International Conference on Fuzzy Systems*, pp. 803–808, IEEE, San Francisco, Calif, USA.

[4] D. Shaked, O. Yaron, and N. Kiryati, "Deriving stopping rules for the probabilistic hough transform by sequential analysis," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 512–526, 1996.

[5] L. Xu, E. Oja, and P. Kultanen, "A new curve detection method: randomized Hough transform (RHT)," *Pattern Recognition Letters*, vol. 11, no. 5, pp. 331–338, 1990.

[6] J. Iivarinen, M. Peura, J. Srel, and A. Visa, "Comparison of combined shape descriptors for irregular objects," in *Proceedings of the 8th British Machine Vision Conference*, pp. 430–439, University of Essex, London, UK, 1997.

[7] I. Habibie, A. Bowolaksono, R. Rahmatullah et al., "Automatic detection of embryo using particle swarm optimization based hough transform," in *Proceedings of the International Symposium on Micro-NanoMechatronics and Human Science, MHS '13*, IEEE, Nagoya, Japan, November 2013.

[8] V. Ayala-Ramirez, C. H. Garcia-Capulin, A. Perez-Garcia, and R. E. Sanchez-Yanez, "Circle detection on images using genetic algorithms," *Pattern Recognition Letters*, vol. 27, no. 6, pp. 652–657, 2006.

[9] K.-Y. Huang and Y.-L. Chou, "Simulated annealing for hierarchical pattern detection and seismic applications," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN '08*, pp. 1257–1264, IEEE, Hong Kong, China, June 2008.

[10] S. Dasgupta, A. Biswas, S. Das, and A. Abraham, "Automatic circle detection on images with an adaptive bacterial foraging algorithm," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO '08*, p. 1695, ACM, Atlanta, Ga, USA, July 2008.

[11] Z. W. Geem, *Music-Inspired Harmony Search Algorithm: Theory and Applications*, vol. 191, Springer Publishing Company, Berlin, Heidelberg, 1st edition, 2009.

[12] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[13] X. Z. Gao, V. Govindasamy, H. Xu, X. Wang, and K. Zenger, "Harmony search method: theory and applications," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 258491, 10 pages, 2015.

[14] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[15] Caltech Computer Vision Group Image Archive, "Electronic image archive," 2005, http://www.vision.caltech.edu/html-files/archive.html.

[16] K. P. Philip, E. L. Dove, K. B. Chandran, D. D. McPherson, N. L. Gotteiner, and W. Stanford, "The fuzzy hough transform-feature extraction in medical images," *IEEE Transactions on Medical Imaging*, vol. 13, no. 2, pp. 235–240, 1994.

[17] E. Cuevas, D. Zaldivar, M. P, and M. Ram, "Circle detection using discrete differential evolution optimization," *Pattern Analysis and Applications*, vol. 14, no. 1, pp. 93–107, 2011.

[18] J. Fourie, S. Mills, and R. Green, "Harmony filter: a robust visual tracking system using the improved harmony search algorithm," *Image and Vision Computing*, vol. 28, no. 12, pp. 1702–1716, 2010.

[19] Z. W. Geem, Ed., *Recent Advances In Harmony Search Algorithm*, vol. 270 of *Studies in Computational Intelligence*, Springer, Berlin, Germany, 1st edition, 2010.

[20] O. M. Alia, R. Mandava, D. Ramachandram, and M. E. Aziz, "Dynamic fuzzy clustering using harmony search with application to image segmentation," in *Proceedings of the 9th IEEE International Symposium on Signal Processing and Information Technology, ISSPIT '09*, pp. 538–543, IEEE, Ajman, UAE, December 2009.

[21] Z. Geem, "Harmony search algorithm for solving sudoku," in *Knowledge-Based Intelligent Information and Engineering Systems*, B. Apolloni, R. J. Howlett, and L. Jain, Eds., vol. 4692, pp. 371–378, Springer, Berlin, Germany, 2007.

[22] Z. W. Geem, K. S. Lee, and Y. Park, "Application of harmony search to vehicle routing," *American Journal of Applied Sciences*, vol. 2, no. 12, pp. 1552–1557, 2005.

[23] Z. W. Geem and J.-Y. Choi, "Music composition using harmony search algorithm," in *Proceedings of the 2007 EvoWorkshops*, pp. 593–600, Springer-Verlag, Berlin, Heidelberg, Germany, 2007.

[24] E. Cuevas, N. Ortega-S, D. Zaldivar, and M. Prez-Cisneros, "Circle detection by harmony search optimization," *Journal of Intelligent & Robotic Systems*, vol. 66, no. 3, pp. 359–376, 2012.

[25] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems Journal*, vol. 4, no. 1, pp. 25–30, 1965.

[26] V. Tirronen and F. Neri, "Differential evolution with fitness diversity self-adaption, studies in computational intelligence," in *Nature-Inspired Algorithms for Optimisation*, vol. 193, pp. 199–234, Springer, Berlin, Heidelberg, Germany, 2009.

[27] D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publishers, Norwell, Mass, USA, 1987.

[28] J. Fourie, R. Green, and S. Mills, "Counterpoint harmony search: an accurate algorithm for the blind deconvolution of binary images," in *Proceedings of the International Conference on Audio, Language and Image Processing, ICALIP '10*, pp. 1117–1122, IEEE, Shanghai, China, November 2010.

Hindawi
Journal of Optimization
Volume 2017, Article ID 8042436, 7 pages
https://doi.org/10.1155/2017/8042436

Research Article

Improving the Fine-Tuning of Metaheuristics: An Approach Combining Design of Experiments and Racing Algorithms

Eduardo Batista de Moraes Barbosa[1] **and Edson Luiz França Senne**[2]

[1] *Brazilian Institute for Space Research (INPE), Cachoeira Paulista, SP, Brazil*
[2] *Universidade Estadual Paulista (UNESP), Guaratinguetá, SP, Brazil*

Correspondence should be addressed to Eduardo Batista de Moraes Barbosa; eduardo.barbosa@inpe.br

Received 10 February 2017; Accepted 10 April 2017; Published 7 June 2017

Academic Editor: Ferrante Neri

Usually, metaheuristic algorithms are adapted to a large set of problems by applying few modifications on parameters for each specific case. However, this flexibility demands a huge effort to correctly tune such parameters. Therefore, the tuning of metaheuristics arises as one of the most important challenges in the context of research of these algorithms. Thus, this paper aims to present a methodology combining Statistical and Artificial Intelligence methods in the fine-tuning of metaheuristics. The key idea is a heuristic method, called Heuristic Oriented Racing Algorithm (HORA), which explores a search space of parameters looking for candidate configurations close to a promising alternative. To confirm the validity of this approach, we present a case study for fine-tuning two distinct metaheuristics: Simulated Annealing (SA) and Genetic Algorithm (GA), in order to solve the classical traveling salesman problem. The results are compared considering the same metaheuristics tuned through a racing method. Broadly, the proposed approach proved to be effective in terms of the overall time of the tuning process. Our results reveal that metaheuristics tuned by means of HORA achieve, with much less computational effort, similar results compared to the case when they are tuned by the other fine-tuning approach.

1. Introduction

Usually, metaheuristic algorithms are adapted to a large set of problems with few modifications on parameters for each specific case. However, this adaptation in some situations demands a huge effort to correctly tune its parameters. Sometimes, due to time restrictions for the tuning process, the algorithms eventually may lead to solutions of poor quality.

The flexibility of metaheuristics allows these algorithms to find acceptable solutions to a wide range of problems, but at the same time it is difficult to obtain good (or, sometimes, optimal) solutions, due to the difficulty of fine-tuning of parameters. Therefore, the tuning of metaheuristics arises as one of the most important research challenges in the context of the design and application of these algorithms.

These challenges are usually of interest to different research communities. In the contemporary literature, researches on statistical techniques stand out and are supported by efficient methods, in order to aid the process of understanding and also to reach effective settings (e.g., [1–7] and many others).

This paper aims to contribute to the literature by presenting a methodology combining Statistical and Artificial Intelligence methods in the fine-tuning of metaheuristics, such as Design of Experiments (DOE) [8] and the concept of Racing [9, 10]. Broadly, our approach employs DOE as a tool to define a parameter search space. Then, we apply the Racing concept to explore the previously defined search space looking for alternatives close to promising candidate configurations, in order to consistently find the good ones. The search process is focused on dynamically created alternatives in an iterative process, which evaluates and discards some of them based on statistical evidences.

The proposed approach brings together the characteristics of different strategies from the literature, such as

CALIBRA [11], I/F-Race [10, 12], and ParamILS [13], in a single heuristic method with the ability to define the search space and efficiently concentrate on searches for the candidate configurations within this search space. This approach will be illustrated by means of a simple case study, where a set of parameters of two distinct metaheuristics (Simulated Annealing, SA, and Genetic Algorithm, GA) will be tuned to solve a classical optimization problem, such as the traveling salesman problem (TSP). The quality of the proposed settings will be compared with a racing tuning approach.

The rest of the paper is structured as follows: Section 2 presents the problem of tuning metaheuristics and our approach combining Statistical and Artificial Intelligence methods to address this problem. In Section 3 there is an overview about the considered problem, as well as the metaheuristics used in a case study. The proposed approach is applied in a case study (Section 4) to fine-tune the metaheuristics SA and GA. Section 4 also presents the case study results and its analyses. Our final considerations are in Section 5.

2. An Approach to the Problem of Tuning Metaheuristics

In the contemporary literature it is possible to find formal definitions related to the problem of fine-tuning the parameters of algorithms [10, 14, 15].

Let M be a metaheuristic with a set of parameters (e.g., $\alpha, \beta, \ldots, \xi$) that must be tuned to solve a class P of problems. The parameters of M can assume a finite set of values and its cardinality can also vary extensively according to M and P. If Θ is a set of candidate configurations, such that θ is any setting of M, then the problem of tuning metaheuristics can be formalized as a state-space:

$$S = (\Theta, P) . \tag{1}$$

Broadly, this problem consists of determining the best setting $\theta \in \Theta$ present in S to solve problems P. However, its determination is not always simple and, in the worst hypothesis, it may require a full search in the state-space S.

2.1. The Dynamic of State-Space. In this study we propose an automatic approach to avoid a full search in the state-space (1) and still find a good setting of M to solve P. The idea is to consider the existence of an *agent*, whose *actions* modify the *state* of (1) [16]. In our approach this agent is a heuristic method, which combines robust statistical methods in an iterative process to create candidate configurations and find the good ones, based on statistical evaluations of a wide range of problems. An action (e.g., the creation of alternatives) is valid if it respects some constraints (e.g., the state-space bounds). Thus, given an initial state (e.g., the best known candidate configuration) we apply the heuristic method to explore (1) through the creation of candidate configurations at the neighborhood of a promising alternative.

The actions yielded by the heuristic method modify the state of (1) by creating candidate configurations on demand at the neighborhood of some best known alternative, as a sequence of sets of candidate configurations:

$$\Theta_0 \Longrightarrow \Theta_1 \Longrightarrow \Theta_2 \Longrightarrow \cdots . \tag{2}$$

Such actions can be thought as a directed graph in the state-space, where the nodes are the states and the arcs are the actions (Figure 1).

From step k to $k+1$ the set of candidate configurations is built possibly discarding some alternatives considered statistically inferior. Given that some candidate configurations persist in this set, they are evaluated on more instances. Therefore, finding a solution is equivalent to finding a path in (1); that from an initial state reaches the final state, that is, a good setting for M.

To illustrate this process (Figure 1), let us consider any state-space, where at each iteration $k, m = 3$ candidate configurations are created. At the end of iteration, all alternatives in the set Θ of candidate configurations are evaluated and those with inferior quality are discarded. Therefore, the set Θ is dynamic; that is, its size can increase or decrease. The process continues pursuing the alternatives in the search space until a stopping criterion (e.g., number of alternatives in Θ, runtime limit) is met.

2.2. Automatic Tuning of Metaheuristics. The tuning process begins with an arbitrary selection of n instances ($n > 1$) from a class of optimization problems and follows by defining ranges for the parameters of metaheuristic. The previously selected instances are treated as a training set, on which experimental studies are performed with the Response Surface Methodology (RSM) to define the best parameters settings for each instance. Therefore, at the end of the experimental phase there will be n different settings for each parameter, each one being related to an instance.

The settings identified in the training set ensure diversity for the parameters, and they are used to define the bounds of each parameter, that is, a search space of parameters limited by the maximum and minimum values of each parameter in the training set. Then, the goal is to pursue alternatives dynamically created at the neighborhood of some best known candidate configuration, with respect to the previously defined bounds of the search space. For each of the alternatives, the target algorithm is run in an expanded set of instances, larger than the previous one.

This process (Figure 2) is called Heuristic Oriented Racing Algorithm (HORA), due to the means of exploring the alternatives in the search space, that is, using a heuristic method, and by its evaluation process through a racing method.

3. Considered Problem and Metaheuristics: Overview

Optimization problems are common in many areas (e.g., science, engineering, management, and business) and different domains. Essentially, they involve finding an extreme value (maximum or minimum) called optimal, from a function with numerous local extremes, called objective function.

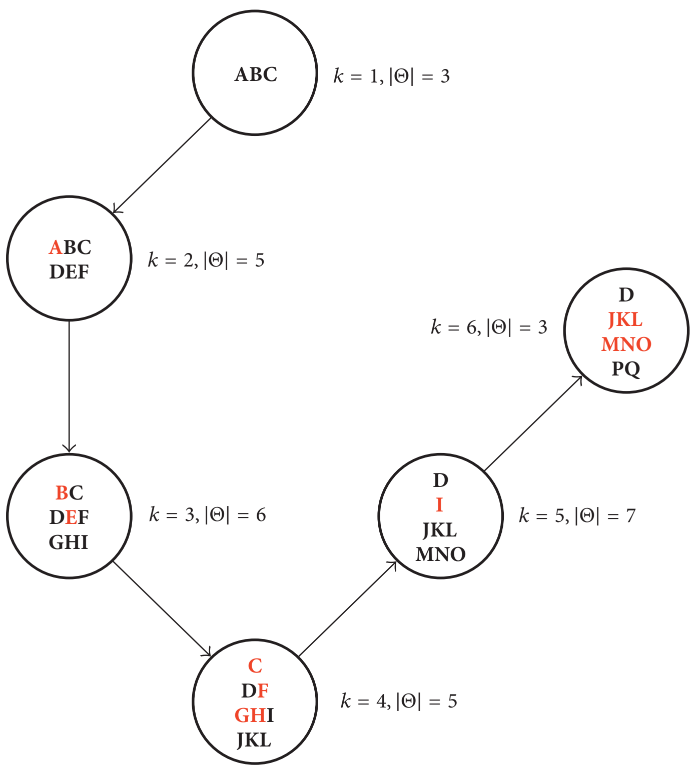


Figure 1: Illustrative process of creating (black) and excluding (red) alternatives in the state-space.

Some of those problems are classical in Operations Research area, such as the traveling salesman problem, and involve a significant number of publications in the specialized literature [17–20].

The traveling salesman problem (TSP) is a classical optimization problem, whose idea is to find the shortest route between a set of given cities, starting and ending at the same city, such that each city is visited exactly once. A TSP consists of a set C of cities ($c = 1, 2, \ldots, n$) and the corresponding distance d_{ij} of each pair of cities $i, j \in C$, such that $i \neq j$. The problem is classified as symmetric if $d_{ij} = d_{ji}$, $\forall i, j$, or asymmetric if $d_{ij} \neq d_{ji}$, $\forall i, j$ [20].

This problem is known to be NP-hard [21]; thus, in order to get the optimal solution, a significant computational effort is required and demands the use of efficient algorithms. Metaheuristics are one of the best known approaches to solving problems for which there is no specific efficient algorithm. Many metaheuristics are inspired by metaphors from different knowledge areas, like physics (particle swarm optimization and Simulated Annealing) and biology (Genetic Algorithms and neural networks). Usually, these algorithms differ from each other in terms of searching pattern but offer accurate and balanced methods for diversification (search space exploration) and intensification (exploitation of a promising region) and share features, such as the use of stochastic components (involving randomness of variables), and have a variety of parameters that must be set according to the problem under study.

Simulated Annealing (SA) is a probabilistic method proposed by Kirkpatrick et al. [22] and Černý [23] in order to find the global minimum of an objective function with numerous local minimums. Widely applied to solve optimization problems, SA simulates a physical process from which a solid is cooled slowly, so that the final product becomes a homogeneous mass to achieve a minimum energy configuration [24]. On the other hand, Genetic Algorithm (GA) is a population-based method invented by Holland [25] inspired on principles of survival from Darwin's evolution theory. GA simulates an evolution process in which the fitness of individuals (parents) is crucial to generating new individuals (children).

The main difference between these algorithms is the searching methods that are implemented. SA performs constant movements between one solution (s) and another (s') according to some predefined neighborhood structure, and it uses a probabilistic test to accept new solutions, which sometimes allows low-quality solutions to be considered in the search process. On the other hand, GA operates on a population of solutions, whose new generations (offspring) are generated from the fittest individuals of previous generations (parents). This feature (survival principle) guarantees an increase in the quality of solutions as new generations are created.

Both SA and GA algorithms have a wide range of parameters (e.g., initial temperature and its rate of decrease, number of interactions, for SA, and rates of crossover and mutation, population size, for GA) that must be tuned before starting

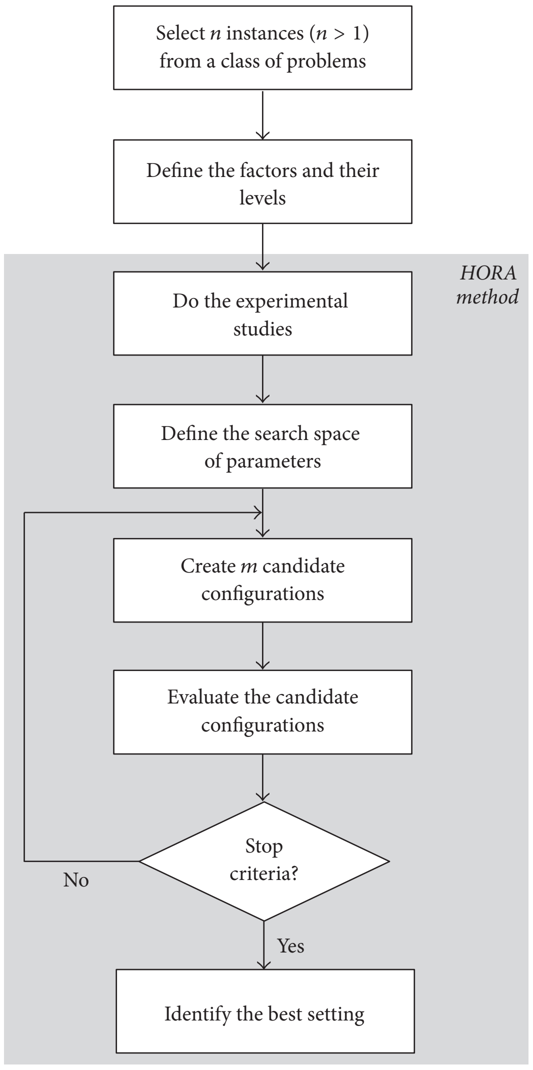

Select n instances ($n > 1$) from a class of problems
Define the factors and their levels
HORA method
Do the experimental studies
Define the search space of parameters
Create m candidate configurations
Evaluate the candidate configurations
Stop criteria?
No
Yes
Identify the best setting

Figure 2: Schema of the proposed approach.

to solve a problem. Since the metaheuristics are extremely dependent on the values assigned to those parameters, they must be carefully studied during the process of fine-tuning, since they can define the success of the algorithm.

4. A Case Study

For our study, we selected a set of parameters of each metaheuristic, which are the most frequently used in the literature and seem to influence the performance of both SA and GA, regardless of the studied problem. The considered parameters for SA are value of the initial temperature (T_0), number of iterations on one temperature stage ($\mathrm{SA}_{\max}$), and temperature cooling rate (α), while the chosen parameters for GA are mutation rate (p_m), crossover rate (p_c), population size (μ), and number of generations (g). The parameters (Table 1) were chosen within the region of operability of parameters (their real limits), in order to promote diversity in the search space, as well as differences between each particular parameter setting.

Table 1: Parameters settings for GA and SA.

SA	Low	High
T_0	1.00e4	1.50e6
$\mathrm{SA}_{\max}$	500	1500
α	0.900	0.980
GA	Low	High
p_m	0.001	0.025
p_c	0.400	0.900
μ	10	100
g	100	1000

The tuning process begins from a training set with $n = 4$ instances arbitrarily selected from the TSP benchmark from the TSPLIB [26]. The studies with DOE were conducted by means of a circumscribed Central Composite Design (CCD) [8], in which some points overcome the previously set limits to ensure an appropriate estimation for the parameters.

Thus, after the experimental studies we have four different results for each parameter, each one being related to an instance. From those results, we defined the search space of parameters, whose limits are the maximum and minimum values of the parameters in the training set. Accordingly, the SA search space is

(i) T_0: $[1.35e5, 1.45e5]$;
(ii) $\mathrm{SA}_{\max}$: $[1466, 1859]$;
(iii) α: $[0.946, 0.947]$.

In the case of GA, we have the following search space:

(i) p_m: $[0.012, 0.027]$
(ii) p_c: $[0.292, 0.774]$
(iii) μ: $[55, 122]$
(iv) g: $[494, 1474]$

Observing the results is noteworthy, where some parameter values are outside of the limits initially defined (Table 1). However, as pointed out before, this occurs due to the chosen CCD, whose axial points establish new limits for a region of interest.

From there, the exploration of the state-space is conducted by means of the HORA heuristic creating the alternatives at the neighborhood of some best known candidate configuration. For each of the alternatives we run the target metaheuristics (e.g., SA and GA) during 15 seconds on an expanded set of instances (e.g., for this study, the expanded set matches 40 instances from the benchmark TSP). This process was repeated 10 times and the results of fine-tuning of the metaheuristics by means of HORA are presented in terms of mean and standard deviation ($\mu \pm \sigma$) in Table 2. This table also presents the total time (in seconds) of the tuning process.

For comparisons, we considered the previously defined search space of parameters and one other fine-tuning approach, such as a racing algorithm based on F-Race

Table 2: Fine-tuning of metaheuristics (HORA).

SA	Settings
T_0	$1.40e5 \pm 2445$
SA_{max}	1584 ± 106
α	0.947 ± 0.000
—	—
t	969
GA	Settings
p_m	0.020 ± 0.005
p_c	0.665 ± 0.123
μ	81 ± 12
g	760 ± 202
t	896

Table 3: Fine-tuning of metaheuristics (*Racing*).

SA	Settings
T_0	$1.39e5 \pm 2411$
SA_{max}	1672 ± 112
α	0.947 ± 0.000
—	—
t	8338
GA	Settings
p_m	0.020 ± 0.007
p_c	0.549 ± 0.164
μ	84 ± 14
g	690 ± 217
t	16740

method [10, 12, 14], from hereon called *Racing*. The settings used for SA are the following: $T_0 = \{1.35e5, 1.37e5, 1.38e5, 1.40e5, 1.42e5, 1.43e5, 1.45e5\}$, $SA_{max} = \{1466, 1564, 1662, 1761, 1859\}$, and $\alpha = \{0.946, 0.947\}$; and for GA we have $p_m = \{0.012, 0.020, 0.027\}$, $p_c = \{0.292, 0.453, 0.613, 0.774\}$, $\mu = \{55, 77, 100, 122\}$, and $g = \{494, 821, 1147, 1474\}$.

These settings were defined in a discrete interval limited by the previously defined search space, with a number of levels that seem to be enough for the algorithms leading to a good result. Thus, each possible combination of parameters leads to a different algorithm setting, such that the search space is composed of 70 and 192 different parameter settings for SA and GA algorithms, respectively. The idea is to use the *Racing* to select a possible good configuration from a number of options, after running the target algorithms for 15 seconds on the same set of instances previously used by HORA. This process was also repeated 10 times and the fine-tuning results of the studied metaheuristics (Table 3) are presented in terms of mean and standard deviation ($\mu \pm \sigma$). The Table also presents the total time (t, in seconds) of this tuning process.

From these results (Table 3) it is possible to note the similarities between HORA and *Racing* in terms of parameter settings, since both approaches employ the same evaluation method for the candidate configurations. But it is possible to highlight that the tuning process using HORA spends about 11.6% and 5.3% of the time required in the fine-tuning process using *Racing*, for SA and GA, respectively.

Table 4: Statistics of the fine-tuning approaches for the metaheuristic SA.

Inst.	gap_H	t_H	gap_R	t_R
Berlin52	0.00	98	0.00	60
St70	1.48	190	1.63	178
Rat99	2.73	260	5.53	243
Rd100	1.78	263	3.67	245
Lin105	0.83	266	1.96	256
Pr124	1.66	285	1.84	271
Bier127	3.12	293	3.40	292
Ch130	2.39	297	4.66	299
KroA200	7.91	300	8.49	299
Ts225	3.23	300	5.32	299
μ	*2.51*	*255*	*3.65*	*244*
σ	*2.04*	*61*	*2.33*	*71*

4.1. Analysis of Results. Although it is not the main purpose of this paper, it is interesting to verify whether the metaheuristics tuned by HORA and *Racing* achieve good results for the TSP. Note that the quality of the results yielded by an algorithm configured by *Racing* can depend heavily on the number of candidate configurations initially established, since this approach performs an exhaustive analysis of the search space of previously defined candidate configurations. Thus, the smaller the candidate configurations space, the greater the probability that the tuning process will finish with a poor parameters setting, compromising the quality of the results (and, even, the execution time) of an algorithm which has been tuned by this process. For HORA approach, this limitation does not exist, since it starts from any candidate configuration and dynamically builds the space of parameters settings.

In general, the metaheuristics employ some degree of randomness to diversify the searches and avoid confinement in the search space. Thus, a single run of these algorithms can result in different solutions from the next run. So, to test the quality of the settings, the experimental results were collected after 5 runs of the metaheuristics SA and GA on the TSP. To compare the results, we use

$$\text{gap} = \frac{f(s) - f(s^*)}{f(s^*)} \times 100, \quad (3)$$

where $f(s)$ is the computed solution and $f(s^*)$ is the best known solution of the problem. Thus, lower the value of *gap*, the better the performance of the algorithms.

The results were collected using the scientific software Scilab (http://www.scilab.org) in an Intel® Core i5TM 1.8 GHz, 6 GB of memory, 1TB of hard disc on a Windows 8 64 bits.

The results were collected considering the settings of the metaheuristics SA and GA presented in Tables 2 and 3, for HORA and *Racing*, respectively. Tables 4 and 5 present the results for 5 runs of SA and GA, in 10 instances of the TSP

Table 5: Statistics of the fine-tuning approaches for the metaheuristic GA.

Inst.	gap_H	t_H	gap_R	t_R
Berlin52	44.60	109	34.83	52
St70	72.15	103	74.67	69
Rat99	106.69	123	106.69	82
Rd100	116.94	114	114.80	74
Lin105	143.98	95	150.88	61
Pr124	194.38	110	205.06	60
Bier127	99.38	100	88.84	63
Ch130	142.36	125	130.70	103
KroA200	227.62	125	216.68	103
Ts225	245.84	112	282.68	74
μ	*139.39*	*109*	*140.58*	*52*
σ	*62.33*	*11*	*70.97*	*17*

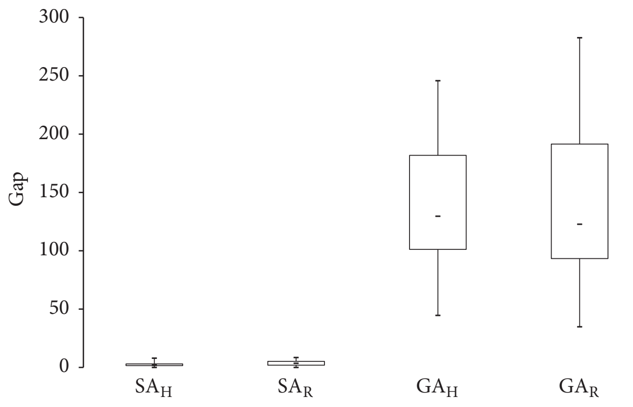


Figure 3: Performance of the studied metaheuristics tuned by different approaches.

benchmark, with the following stopping criteria: number of iterations without changes in the objective function (200 iterations) and maximum running time (300 seconds). In these tables, the results of the approaches are underlined in capital letters H (for HORA) and R (for *Racing*). The column *gap* is the best found value of (3), t is average of the runtime (t), and μ and σ are arithmetic mean and standard deviation of all selected instances.

From the results for the metaheuristic SA (Table 4) we note that the fine-tuning process by means of HORA achieves solutions with better quality. Note that the *gap* of SA tuned by HORA is, on average, about 30% better than the *gap* of the corresponding version tuned by *Racing*, although the running time for SA_H is, on average, about 4.5% greater than for SA_R. This deficit of performance may be related to the parameter T_0 (value of the initial temperature), which is slightly higher according to HORA and can require more time to find a good quality solution.

For the results achieved by GA (Table 5), we also note slightly better quality solutions when the metaheuristic is tuned by HORA, although this version demands, on average, the double of the running time than its corresponding version tuned by *Racing*.

The graphical analysis (Figure 3) reveals differences between the data distributions from the studied metaheuristics, such that the results of SA figure are closer to the optimal solution. According to the results (Tables 4 and 5) and supported by graphical analysis it is possible to confirm that the fine-tuning process by means of HORA can achieve better quality solutions.

5. Final Considerations

This paper presented a method addressed to the problem of tuning metaheuristics. The problem was formalized as a state-space, whose exploration is conducted effectively by a heuristic method combining Statistical (DOE) and Artificial Intelligence (racing algorithms).

The proposed method, called HORA, applies robust statistics on a limited number of instances from a class of problems, in order to define a search space of parameters. Thus, from the alternatives dynamically created at the neighborhood of some best known candidate configuration, it employs a racing method to consistently find the good settings.

From a case study, HORA was applied in the fine-tuning of two distinct metaheuristics. Its results were compared with the same metaheuristics tuned by means of a racing algorithm. The HORA method proved to be effective in terms of overall time of the tuning process, since it demands just a fraction of the time required by the other studied approach. Through the experimental studies it is noted that the metaheuristics SA and GA tuned by HORA can achieve similar results (eventually better) from those obtained by the *Racing* approach, but it is highlighted that the tuning process is much faster with HORA. This better performance can be explained by how the alternatives in the state-space are explored, which consists in creating alternatives at the neighborhood of some best known candidate configuration, and evaluate them by a racing method.

The aim of this study was to present the proposed approach and verify its effectiveness when applied on different metaheuristics. Our results suggest that HORA may be a promising and powerful tool to assist in the fine-tuning of different algorithms. Some additional studies can be conducted to verify its effectiveness considering other metaheuristics and problems, as well as explore other heuristic alternatives to pursue good configurations in the state-space of parameters settings.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] F. Dobslaw, "A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks," in *Proceedings of the 6th International Conference on Natural Computation*, vol. 6, pp. 1–6, Cairo, Egypt, 2010.

[2] S. Lessmann, M. Caserta, and I. M. Arango, “Tuning metaheuristics: a data mining based approach for particle swarm optimization,” *Expert Systems with Applications*, vol. 38, no. 10, pp. 12826–12838, 2011.

[3] C. Neumüller, S. Wagner, G. Kronberger, and M. Affenzeller, “Parameter meta-optimization of metaheuristic optimization algorithms,” in *Proceedings of 13th International Conference on Computer Aided Systems Theory (EUROCAST 2011)*, vol. 13, pp. 367–374.

[4] J. Ries, P. Beullens, and D. Salt, “Instance-specific multi-objective parameter tuning based on fuzzy logic,” *European Journal of Operational Research*, vol. 218, no. 2, pp. 305–315, 2012.

[5] H. Akbaripour and E. Masehian, “Efficient and robust parameter tuning for heuristic algorithms,” *International Journal of Industrial Engineering and Production Research, Tehran*, vol. 24, no. 2, pp. 143–150, 2013.

[6] M. Amoozegar and E. Rashedi, “Parameter tuning of GSA using DOE,” in *Proceedings of 4th International Conference on Computer and Knowledge Engineering ICCKE 2014*, pp. 431–436, October 2014.

[7] L. Calvet, A. A. Juan, C. Serrat, and J. Ries, “A statistical learning based approach for parameter fine-tuning of metaheuristics,” *SORT (Statistics and Operations Research Transactions)*, vol. 40, no. 1, pp. 201–224, 2016.

[8] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, NJ, USA, 8th edition, 2012.

[9] O. Maron and A. W. Moore, “Hoeffding races: accelerating model selection search for classification and function approximation,” in *Advances in Neural Information Processing Systems*, pp. 59–66, Morgan Kaufmann, San Mateo, Calif, USA, 1994.

[10] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp, “A racing algorithm for configuring metaheuristics,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 11–18, New York, NY, USA, 2002.

[11] B. Adenso-Díaz and M. Laguna, “Fine-tuning of algorithms using fractional experimental designs and local search,” *Operations Research*, vol. 54, no. 1, pp. 99–114, 2006.

[12] P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo, “Improvement strategies for the F-race algorithm: sampling design and iterative refinement,” in *Proceedings of the 4th International Workshop on Hybrid Metaheuristics*, pp. 108–122.

[13] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, “ParamILS: an automatic algorithm configuration framework,” *Journal of Artificial Intelligence Research*, vol. 36, pp. 267–306, 2009.

[14] M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective*, Springer, New York, NY, USA, 2nd edition, 2009.

[15] S. Smit and A. Eiben, “Comparing parameter tuning methods for evolutionary algorithms,” in *Proceedings of the IEEE Congress on Evolutionary Computing*, pp. 399–406, IEEE, Trondheim, Norway, May 2009.

[16] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Pearson, London, UK, 3rd edition, 2009.

[17] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, “The traveling salesman problem: a computational study,” in *Princeton Series in Applied Mathematics*, Princeton University Press, Princeton, NJ, USA, 2nd edition, 2007.

[18] G. Laporte, “What you should know about the vehicle routing problem,” *Naval Research Logistics*, vol. 54, no. 8, pp. 811–819, 2007.

[19] U. Derigs, *Optimization and Operations Research*, vol. 2, EOLSS Publishers Co. Ltd., Paris, France, 2009.

[20] R. Matai, S. P. Singh, and M. L. Mittal, “Traveling salesman problem: an overview of applications, formulations, and solution approaches,” in *Traveling Salesman Problem: Theory And Applications*, D. Davendra, Ed., Chapter 1, pp. 1–24, InTech, 2010.

[21] J. K. Lenstra, A. H. G. Rinnooykan, and P. Brucker, “Complexity of machine scheduling problems,” in *Annals of discrete mathematics*, P. L. Hammer, E. L. Johnson, B. H. Korte, and G. L. Nemhauser, Eds., pp. 343–362, Elsevier, Amsterdam, 1977.

[22] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by simulated annealing,” *American Association for the Advancement of Science. Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[23] V. Černý, “Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm,” *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1985.

[24] D. Bertsimas and J. Tsitsiklis, “Simulated annealing,” *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993.

[25] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Boston, Mass, USA, 1975.

[26] G. Reinelt, “TSPLIB: a traveling salesman problem library,” *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.

Hindawi
Journal of Optimization
Volume 2017, Article ID 5650364, 8 pages
https://doi.org/10.1155/2017/5650364

Research Article

A Genetic Algorithm Based Approach for Solving the Minimum Dominating Set of Queens Problem

Saad Alharbi[1] **and Ibrahim Venkat**[2]

[1] *Computer Science Department, Taibah University, Medina, Saudi Arabia*
[2] *School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia*

Correspondence should be addressed to Saad Alharbi; salharbi20@gmail.com

Received 20 February 2017; Revised 24 April 2017; Accepted 3 May 2017; Published 4 June 2017

Academic Editor: Gexiang Zhang

In the field of computing, combinatorics, and related areas, researchers have formulated several techniques for the Minimum Dominating Set of Queens Problem (MDSQP) pertaining to the typical chessboard based puzzles. However, literature shows that limited research has been carried out to solve the MDSQP using bioinspired algorithms. To fill this gap, this paper proposes a simple and effective solution based on genetic algorithms to solve this classical problem. We report results which demonstrate that near optimal solutions have been determined by the GA for different board sizes ranging from 8×8 to 11×11.

1. Introduction

The MDSQP which is otherwise known as the chess covering problem is one of the well-known chessboard problems which has been receiving continued interests by researchers including the computational intelligence domain. In the field of graph theory, specifically the study of dominating sets had addressed this problem even in the early 1970s. In $n \times n$ chessboard, cells are organized as n rows and n columns. In chess layman terms, a queen (say Q) placed on a square will dominate all squares associated with the row, column, and two diagonals with reference to Q. The idea behind this problem is to find the minimum number of queens required to dominate the whole chessboard. Domination here refers to the coverage of all the possible squares being attacked by those queens including the square dominated by the respective queens. Previous researches have been trying to find the domination number $\gamma(Qn)$ for the n-queens problem using mathematical and combinatorial approaches [1–8]. Many formulations have been derived and presented in these approaches. It has been observed that limited efforts have been carried out to employ evolutionary algorithms to address the MDSQP. In fact, most of the studies in the literature were devoted to address the original n-queens problem which is called a nonattacking problem. The results of such studies have demonstrated that evolutionary algorithms are capable of outperforming other principled approaches such as backtracking to solve such problems. However, these studies have focused only on the nonattacking n-queens problem, whereas, to the best of our knowledge, the MDSQP has been least considered using bioinspired evolutionary algorithms.

Evolutionary algorithms have been proved to be successful for solving and optimizing a wide range of complex problems including combinatorial ones, such as the one studied here, within reasonable computing time [9]. Generally speaking, evolutionary algorithms can be classified into two main categories which are swarm intelligence based approaches and classical evolutionary approaches. Swarm intelligence (SI) approaches are inspired from the social natural behavior such as animals group behavior when searching for food or avoiding certain threat [10]. Several SI approaches with reference to the literature include Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). ACO have been proved to efficiently solve several optimization problems [11]. PSO has also demonstrated good results in

Table 1: Domination number for some n values pertaining to the MDSQP.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\gamma(\mathbf{Qn})$	1	1	1	3	3	4	4	5	5	5	5	7	7	≤ 8	≤ 9	≤ 9	9	9

optimizing various complex problems such as the supply chain management (e.g., [12]) and shortest path problems [13, 14].

The classical evolutionary algorithms, on the other hand, including genetic algorithms, were successfully adopted in a wide range of optimization problems. Genetic algorithm is one of the most commonly used evolutionary algorithms in the literature which was first proposed in the 1970s [15]. GA was inspired from the natural process of searching and selection process which leads to the survival of the fittest individuals [15]. Researchers in the literature have demonstrated the efficiency of adopting GA in solving various optimization problems such as data mining [16] and network traffic control [17]. In fact, GA have been adopted in the literature solely to solve some problems such as [17, 18] and were hybridized with other approaches for more enhanced results. Commonly, GA is hybridized with local and heuristic search approaches such as Cuckoo search [19] and Tabu search [20]. Furthermore, various researchers hybridized it with other evolutionary approaches like dynamic programming as in [21].

Therefore, this paper intends to fill these gaps and present a bioinspired genetic algorithm (GA) to solve the MDSQP problem by considering board sizes gradually from the typical 8×8 chessboard up to a 11×11 board. The proposed approach gradually increments the number of queens placed on the board from just below the lower bounds suggested in the literature until an optimal solution has been determined (i.e., all the squares of the chessboard get dominated). The remaining paper is organized as follows: Section 2 discusses related work; Section 3 formulates the problem; Section 4 presents the evolutionary mechanism of the proposed technique; results are discussed in Section 5 and the paper is finally concluded with insights for future work in Section 6.

2. Related Work

Vast amount of efforts have been devoted to investigate chessboard problems. Problems related to chessboard can mainly be classified to the original n-queen problem and the dominating set (i.e., the covering problem). The original n-queen problem can be defined as placing n queens on $n \times n$ chessboard in an optimal way such as none of the queens should attack each other [22, 23]. It was first introduced in 1850 [24]. Numerous researches have been performed to solve this problem using different approaches; most of them adopted mathematical approaches and graph theory such as [25–27]. The study of such a problem may benefit in the understanding of various applications such as traffic control, deadlock prevention, and parallel memory storage [28]. The complexity of the n-queen problem is known as exponential where it gets more complex with large n values. Various approaches were adopted to solve this problem such as backtracking [29], neural networks [30–32], and evolutionary and optimization algorithms [33–42].

The domination set problem on the other hand has received less attention by computer science researchers. In fact, various researches have been devoted to investigate such a problem but were mainly adopting mathematical models. Burger and Mynhardt pointed out that the queens dominating problem is one of the most difficult chessboard problems [43]. It is commonly defined as finding the minimum numbers of queens required to cover all the squares of $n \times n$ chessboard. This number is termed as the domination number and referred by the notation $\gamma(Qn)$. Researchers have arrived at upper and lower bounds since early 1970s [3]. Many formulations and values were derived and found especially for small values of n [3, 43–45]. Table 1 shows some of these values. Figure 1 shows a brief taxonomy of typical chessboard problems. For instance, several studies were conducted to find the domination number of the normal domination [8]. In these studies, it is required to find the minimum queens to be optimally placed on $n \times n$ chessboard that will cover all squares in the board. However, many researchers considered the domination set problems under several variations. For instance, many studies were conducted to investigate the independent domination number which can be defined as the number of queens which do not attack each other and cover the entire $n \times n$ chessboard [3]. Various values and formulations were derived and proved in a mathematical context. Furthermore, the "diagonal queens domination problem" was frequently stated in the literature. In such a problem, it is required to find the number of minimum queens placed in the main diagonal and cover the entire $n \times n$ chessboard [3, 4]. Cockayne has also tried to find the domination number of the queens placed on a single column [3]. Furthermore, many researchers carried out studies to solve the problem of the domination set on a toroidal chessboard [1]. In such studies, it is required to find out the minimum number of queens that would cover the entire squares in the torus. Bozóki et al., on the other hand, shed the light on a different problem called "domination of the rectangular queens graph" where it is required to find the domination number of queens in a rectangular chessboard ($n \times m$) in which the number of squares in column and rows are different [46]. They derived lower bounds for small values of n and m particularly $4 \leq m \leq n \leq 18$.

As seen above, the queens domination set problem has been enormously studied especially using mathematical methods. However, literature still lacks the adoption of computational models particularly optimization approaches to solve this problem. Limited articles were found technically considering the problem. For instance, Fernau tried to analyze the complexity of this problem using backtracking, dynamic programming on subsets, and dynamic programming on path decomposition [47]. He indicated that the

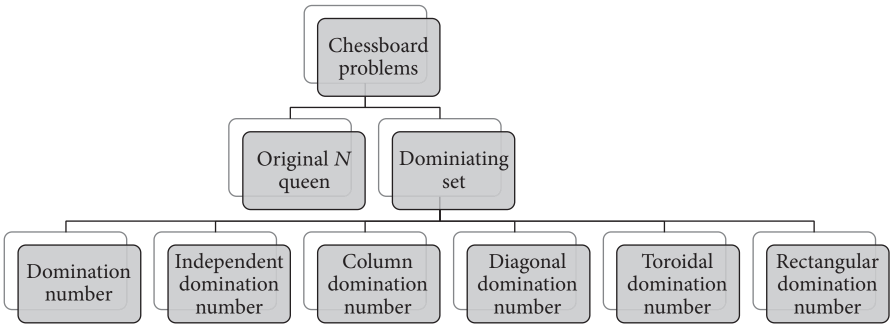

Chessboard problems
Original N queen
Dominiating set
Domination number
Independent domination number
Column domination number
Diagonal domination number
Toroidal domination number
Rectangular domination number

FIGURE 1: Basic taxonomy of chessboard problems.

	1	2	3	4	5	6	7	8
1	1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15	16
3	17	18	19	20	21	22	23	24
4	25	26	27	28	29	30	31	32
5	33	34	35	36	37	38	39	40
6	41	42	43	44	45	46	47	48
7	49	50	51	52	53	54	55	56
8	57	58	59	60	61	62	63	64

FIGURE 2: The proposed labelling scheme of a typical 8 × 8 chessboard.

complexity of the dominating queen problem is a challenging problem. On the other hand, Mohabbati-Kalejahi et al. (2012) presented a hybrid Imperialist Competitive Algorithm (ICA) with a local search algorithm in order to solve the nonattacking queens based domination problems [48]. The proposed hybrid algorithm was compared with the basic ICA in the two problems and the results indicated an improved performance in terms of average runtime and average fitness value. The solution was formulated for the nonattacking n queens problem. However, for the domination problem less details and information have been provided pertaining to the implementation details of the proposed algorithm. For example, crucial components of an evolutionary algorithm such as the formulation of a fitness function have not been presented in detail. Therefore, this work is intended to fill this gap in the literature by adopting a typical bioinspired genetic algorithm (GA) to solve the *Minimum Dominating Set of Queens Problem* (MDSQP). The proposed algorithm is described in the following sections.

3. Problem Formulation

In order to solve the problem and present it programmatically we firstly labelled all squares in the chessboard by sequencing numbers starting gradually from 1 to $n \times n$ starting from the top most left square to the bottom right most square. Figure 2 shows an example of the proposed labelling schema with reference to the commonly used 8×8 chessboard. In addition, all squares in the board have a location (analogous to the norms of the analytical geometry) in the (x, y) coordinate form, where x denotes the row number and y denotes the column number. For example, the location of the square 28 in Figure 2 is (4, 4). This location can be used to determine the dominated squares for a queen placed in a particular square. Let G be a set of chessboard squares $G = \{p_1, p_2, \ldots, p_{n\times n}\}$ and let S_q be the set of dominated squares by queen q for the case of i number of queens. Then it follows that

$$S_{qi} = \{p_1, p_2, \ldots, p_{n\times n}\}. \tag{1}$$

Defining a fitness function for this problem firstly requires defining a method to identify dominated squares for all queens on the board. For a queen placed on the location (x, y), a square n will be $n \in S_q$ if at least one of the following conditions holds:

$$x = xj \quad \forall j\ 1, 2, \ldots, n, \tag{2}$$

$$y = yj \quad \forall j\ 1, 2, \ldots, n, \tag{3}$$

$$\begin{gathered} xj = x - j, \\ y = y - j \\ \forall j\ 1, 2, \ldots, x - 1, \end{gathered} \tag{4}$$

$$\begin{gathered} xj = x + j, \\ yj = y - j \\ \forall j\ 1, 2, \ldots, y - 1, \end{gathered} \tag{5}$$

$$\begin{gathered} xj = x - j, \\ yj = y + j \\ \forall j\ 1, 2, \ldots, x - 1, \end{gathered} \tag{6}$$

$$\begin{gathered} xj = x + j, \\ yj = y + j \\ \forall j\ 1, 2, \ldots, n - 1. \end{gathered} \tag{7}$$

Equations (2) and (3) represent the dominated squares in terms of rows and columns, respectively. The rest of the equations from (4) to (7) represent the squares dominated

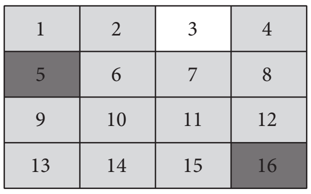


Figure 3: A typical example showing the proposed coding scheme.

by a queen placed in (x, y) for the diagonal directions. Specifically the dominated squares of the two diagonals pertaining to the four sides (upper left, lower left, upper right, and lower right). For example, squares 19, 10, and 1 in Figure 2 are instances of dominated squares by the queen placed on the square 28 and this case has been modeled using a logical algebraic expression using (4). The main objective is to maximize the number of dominated squares in a chessboard by k queens and finding optimal locations of these k queens is quite challenging owing to the complexity of permutations and combinations involved. Therefore, this is measured by the following fitness function:

$$f(x) = \frac{|S|}{|G|}, \tag{8}$$

where

$$S = s_{q_1} \cup s_{q_2} \cup \cdots \cup s_{q_k}. \tag{9}$$

The value of this function approaches 1 when all the squares of the chessboard get dominated at least once. In other words, a fitness value <1 could cohesively indicate that there are some squares or at least 1 square which has not been dominated by any ith queen (s_{q_i}).

4. Formulation of the Problem Using Genetic Algorithms

The proposed GA formulation has been modeled to solve the $n \times n$ chessboard. The overall structure of the proposed GA can be described in Figure 3.

4.1. Coding. Individuals have been coded in a matrix of order $k \times m$. Intuitively $k \times m$, the product of the columns and rows, represents the population size (number of possible candidate solutions). The elements of the matrix represent m bit binary encodings of the labels (each square has been labelled as discussed before) of the queens being placed on the chessboard. The following matrices show typical examples for the case of few possible candidate solutions

$$A = \begin{bmatrix} 1100110 & 0011010 \\ 0101100 & 1100010 \end{bmatrix}. \tag{10}$$

Furthermore, the dominated square set has also been coded in a matrix S as shown below; the total number of unique dominated squares is represented by the number of rows.

$$S = \begin{bmatrix} 1110010 \\ 0000001 \end{bmatrix}. \tag{11}$$

The above matrix shows an example instance of a dominated set containing two squares in the chessboard. Furthermore, let us assume that we would like to encode only one candidate solution in a 4×4 chessboard. Figure 3 shows that two queens are placed in the squares labelled 6 and 16. By converting the decimal values of these squares into 8-digit binary code and taking into consideration having two queens, therefore, we get a 1×2 matrix which is shown below:

$$A = \begin{bmatrix} 00000110 & 00010000 \end{bmatrix}. \tag{12}$$

Furthermore, to identify the dominated squares by the two queens placed in the chessboard, (2) to (7) must be applied. Accordingly, Figure 3 shows that all squares in the chessboard except label 3 are dominated by the two queens. Therefore, the dominated set S is as follows:

$$S = \{1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}. \tag{13}$$

By converting the elements of S into 8-digit binary code we will get the following dominated set matrix S:

$$S = \begin{bmatrix} 00000001 \\ 00000010 \\ 00000011 \\ \vdots \\ \vdots \\ 00010000 \end{bmatrix}. \tag{14}$$

Illustration of typical crossover operations is as follows:

$$\begin{array}{lcl} 110|\,0100 & \Longrightarrow & 1101110 \\ 001|\,1110 & & 0010100 \end{array} \tag{15a}$$

$$\begin{array}{lcl} 00|\,11111 & \Longrightarrow & 000\mathbf{1}000 \\ 11|\,00000 & & 111111\mathbf{0}. \end{array} \tag{15b}$$

4.2. Initial Population and Fitness Evaluation. A random population consisting of 100 individuals (possible candidate solutions) is generated at the initialization stage of the GA making sure that queens are placed in different squares in the board. The domination rate of the queens is computed using the fitness function as modeled in Section 2. The selection criteria of our proposed techniques are as follows. Individuals are sorted based on their fitness value and the top 50% of the population (candidates with the potential fitness capability) will be allowed to survive and will be considered into the next generation. Candidates whose fitness capability is below 50% will be discarded (they will not be allowed to survive). Subsequent to this selection process, the classical roulette-wheel selection method has been adopted for selecting individuals that will be mated for reproduction.

4.3. Crossover and Offspring Generation. The new generation is produced by implementing the one-point crossover between two parents. The steps involved in the proposed crossover operator can be described as follows:

(i) Two individuals are selected according to the selection operator.

(ii) Generate a random crossover point in the two parents.

(iii) Exchange genetic materials after the crossover point.

(iv) Validate new individuals so that the individuals are retained within the boundaries of the chessboard.

(v) If an offspring's location happens to be to zero owing to stochastics, generate a random adjustment point.

(vi) Change the value of the adjustment position to 1.

(vii) If an offspring's location is larger than $n \times n$, based on the chessboard size change the value of the digits of the chromosome causing this issue to 0.

Equation (15a) shows an example of a crossover operation between two parents to produce two children (offspring); (15b) shows an example of a crossover operation to illustrate the validation and adjustment process.

Furthermore, we have also implemented mutation with the probability of 0.05. In mutation, we simply select two random positions for the selected individuals and substitute the values of these positions to their complements (i.e., change zero to one and vice versa). In order to avoid the production of individuals outside the boundaries of the chessboard, the same crossover validation and adjustment mechanism have been found to be efficient.

5. Experiments and Results

The proposed GA has been implemented using MATLAB and experimented using a 2.6 Ghz PC enabled with Intel i5 processor installed with Microsoft Windows 8 operating system. It was independently tested using a different size of the chessboard starting from the common chessboard size 8×8 to 11×11. In each test instance the number of queens placed in the chessboard is gradually incremented starting from the lower bounds reported in the literature until the maximum fitness has been reached. Furthermore, the proposed GA was iterated 1000 times for each instance and the best fitness values were recorded. The results in this contribution are reported after running the GA for five times for each test instance. Table 2 shows a summary of the maximum fitness reached by the GA for the value of $n = 8$ to $n = 11$ as well as the queen numbers placed in the chessboard. The results obtained were promising and close to upper bounds reported in the literature for MDSQP.

It can be noticed that the GA could cover 88% and 95% of the 8×8 chessboard by placing three and four queens on the board, respectively. The GA was also able to find the optimal solution for the same size board when placing five queens meaning that the domination number of 8×8 chessboard is five (i.e., $\boldsymbol{\gamma}(\mathbf{Qn})$). These findings compare well with the results reported in the literature. As the proposed GA is considered to be a pioneer on the literature considering the MDSQP, the obtained results will be compared with the values in the literature which have been derived mathematically. Table 3 shows the optimal domination numbers obtained by the GA compared with these. Table 2 shows also that the GA was able to dominate 99% of the 9×9 chessboard when five queens have been placed in the board, although five is derived as the domination number for a 9×9 chessboard ($\boldsymbol{\gamma}(\mathbf{Qn}) = \mathbf{5}$). However, an optimal solution was reached by increasing number of queens placed on the board to six ($\boldsymbol{\gamma}(\mathbf{Qn}) = \mathbf{6}$) (see Table 3). Similarly, five was reported as the domination number for the case of 10×10 and 11×11 chessboard; however, the GA was not able to cover the whole board. In fact, only 95% of the board was covered when $n = 10$ and five queens were placed ($k = 5$) (see Table 2). The domination percentage has been improved slightly when adding one queen ($k = 6$) to 10×10 board. However, an optimal solution was found when placing seven queens to the board in the case of n =10, 11 ($\boldsymbol{\gamma}(\mathbf{Qn}) = \mathbf{7}$). All solutions produced were graphically simulated by the GA at the end of the execution. Figure 4 shows the simulations pertaining to the optimal solutions of the proposed GA for the case of $n = 8$ to 11. The red circles (dots) represent the queens and the blue circles represent the dominated squares in a typical $n \times n$ chessboard. It also shows the fitness value of the presented solution corresponding to the generation number of the solution. In addition, the performance of the proposed GA solution was measured after the execution of each test instance. The results indicated that the performance in terms of fitness value is increasing over iterations. Due to the space limitation we present the performance results of the 10×10 chessboard in Figure 5. It can clearly be observed that the fitness values in all the cases have been improved when the iterations tend to approach their upper limits.

It can be seen that the solutions yielded by the proposed technique could yield a good domination performance (95%

Table 2: Progression of fitness values: n is the board size and k is number of queens placed on the board.

n	k	Fitness value
8	3	0.88
	4	0.95
	5	1
9	5	0.99
	6	1
10	5	0.95
	6	0.97
	7	1
11	6	0.97
	7	1

Table 3: The obtained results by the GA and domination numbers $\boldsymbol{\gamma}(\mathbf{Qn})$ reported by the literature.

n	8	9	10	11
GA	5	6	7	7
Literature	5	5	5	5

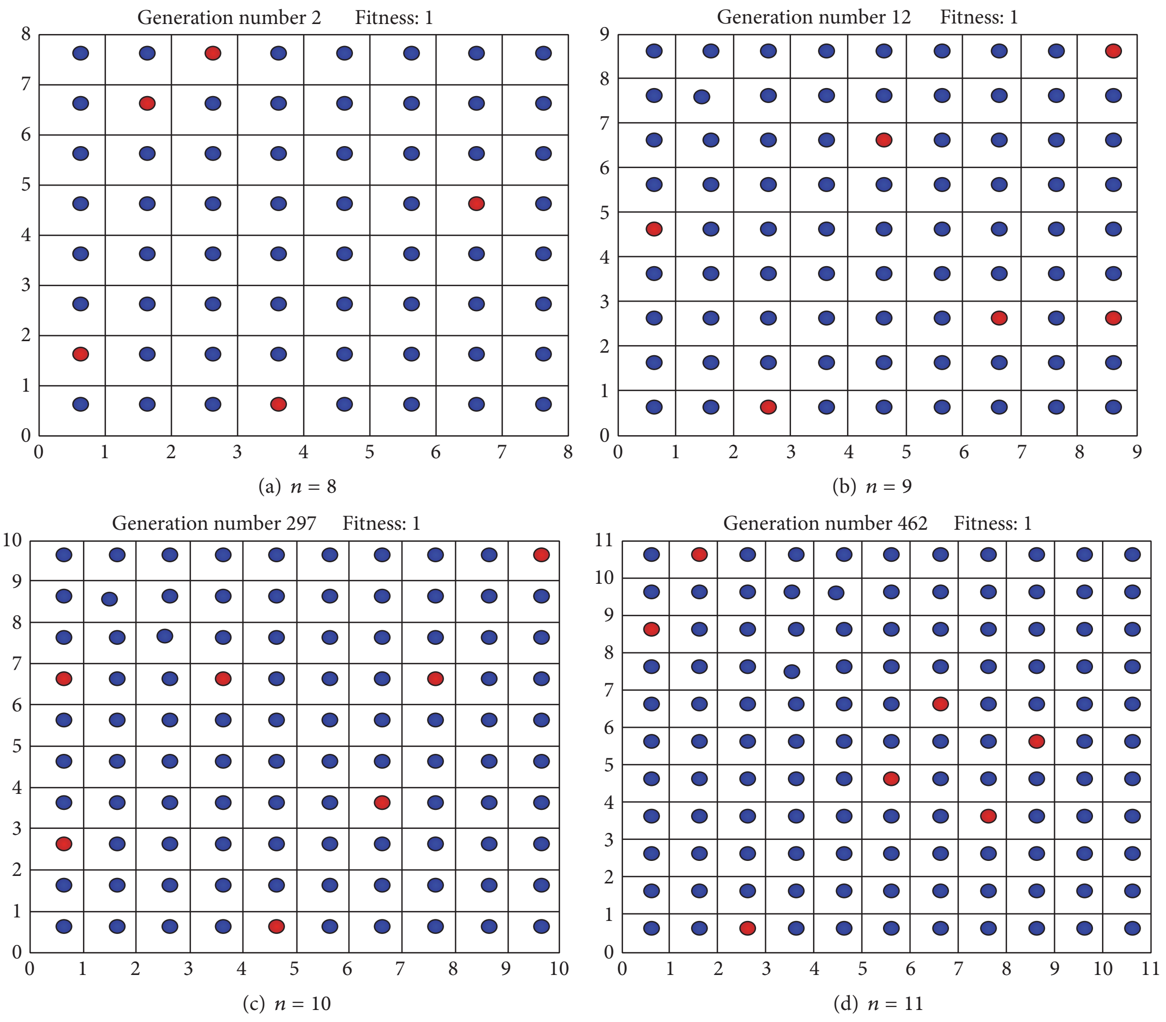

Generation number 2 Fitness: 1
(a) $n = 8$
Generation number 12 Fitness: 1
(b) $n = 9$
Generation number 297 Fitness: 1
(c) $n = 10$
Generation number 462 Fitness: 1
(d) $n = 11$

Figure 4: Optimal solutions yielded by the proposed GA based technique. Red dots represent queens and blue dots represent squares dominated by the queens.

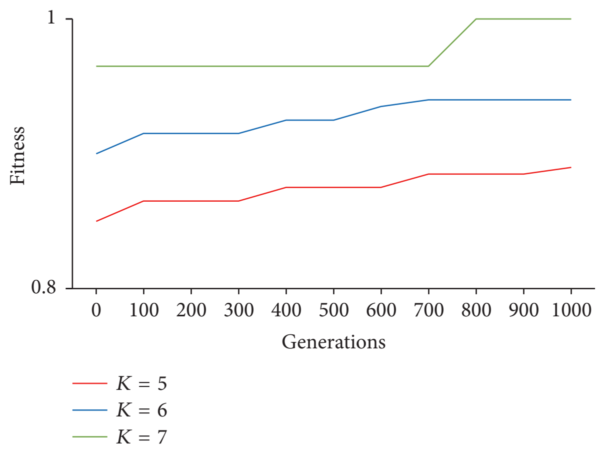


Figure 5: Performance of the GA for a typical case; $n = 10$ and $k = 5, 6$ and 7.

to 99%) even for larger board sizes, particularly for the cases of $n = 10$ and $n = 11$.

6. Conclusion

In this paper, we have proposed a *genetic algorithm* (GA) based technique to solve the classical *Minimum Dominating Set of Queens Problem* (MDSQP). Experimental results demonstrate that the proposed GA based solution was able to find optimal to near optimal solutions for even large sizes of boards ($n = 10$ and $n = 11$). For the case of a typical 8×8 chessboard, the proposed solution yielded a perfect optimal solution on a par with the solutions reported in the literature. Furthermore, experimental results also systematically demonstrate that the average fitness value approached towards the maximum over subsequent iterations. These results will be a basis for our future work where we intend to deploy other bioinspired computing techniques such as membrane computing.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Burger, C. Mynhardt, and W. D. Weakley, "The domination number of the toroidal queens graph of size 3k x 3k," *Australasian Journal of Combinatorics*, vol. 28, pp. 137–148, 2003.

[2] A. P. Burger and C. M. Mynhardt, "An improved upper bound for queens domination numbers," *Discrete Mathematics*, vol. 266, no. 1–3, pp. 119–131, 2003.

[3] E. J. Cockayne, "Chessboard domination problems," *Annals of Discrete Mathematics*, vol. 48, pp. 13–20, 1991.

[4] E. J. Cockayne and S. T. Hedetniemi, "On the diagonal queens domination problem," *Journal of Combinatorial Theory, Series A*, vol. 42, no. 1, pp. 137–139, 1986.

[5] T. Kikuno, N. Yoshida, and Y. Kakuda, "A linear algorithm for the domination number of a series-parallel graph," *Discrete Applied Mathematics*, vol. 5, no. 3, pp. 299–311, 1983.

[6] A. Klobučar, "Domination numbers of cardinal products P_6 × P_n," *Mathematical Communications*, vol. 4, no. 2, pp. 241–250, 1999.

[7] N. Sari and D. IH Agustin, *On the Domination Number of Some Graph Operations*, 2016.

[8] S. S. Venkatesan and Venkatesan S., "Tight lower bounds for connected queen domination problems on the chessboard," https://arxiv.org/abs/1608.02531.

[9] B. Doerr, A. Eremeev, C. Horoba, F. Neumann, and M. Theile, "Evolutionary algorithms and dynamic programming," in *Proceedings of the 11th annual conference on genetic and evolutionary computation*, pp. 771–778, ACM, Québec, Canada, 2009.

[10] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[11] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, pp. 21–57, 2014.

[12] R. S. Kadadevaramath, J. C. H. Chen, B. L. Shankar, and K. Rameshkumar, "Application of particle swarm intelligence algorithms in supply chain network architecture optimization," *Expert Systems with Applications*, vol. 39, no. 11, pp. 10160–10176, 2012.

[13] Y. Marinakis, A. Migdalas, and A. Sifaleras, "A hybrid particle swarm optimization—variable neighborhood search algorithm for constrained shortest path problems," *European Journal of Operational Research*, vol. 261, no. 3, pp. 819–834, 2017.

[14] M. Akhand, S. Hossain, and S. Akter, "A comparative study of prominent particle swarm optimization based methods to solve traveling salesman problem," *International Journal of Swarm Intelligence and Evolutionary Computation*, vol. 5, no. 139, p. 2, 2016.

[15] M. Srinivas and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, no. 6, pp. 17–26, 1994.

[16] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine Learning*, vol. 3, no. 2-3, pp. 95–99, 1998.

[17] A. Barolli, M. Takizawa, F. Xhafa, and L. Barolli, "Application of genetic algorithms for QoS routing in mobile ad-hoc networks: A survey," in *Proceedings of the 5th International Conference on Broadband Wireless Computing, Communication and Applications (BWCCA '10)*, pp. 250–259, Fukuoka, Japan, November 2010.

[18] B. M. Varghese and R. J. S. Raj, "A survey on variants of genetic algorithm for scheduling workflow of tasks," in *Proceedings of the 2nd International Conference on Science Technology Engineering and Management (ICONSTEM '16)*, 2016.

[19] A. Abu-Srhan and E. Al Daoud, "A hybrid algorithm using a genetic algorithm and cuckoo search algorithm to solve the traveling salesman problem and its application to multiple sequence alignment," *International Journal of Advanced Science and Technology*, vol. 61, pp. 29–38, 2013.

[20] A. M. Allakany, T. M. Mahmoud, K. Okamura, and M. R. Girgis, "Multiple constraints QoS multicast routing optimization algorithm based on Genetic Tabu Search Algorithm," *Advances in Computer Science*, vol. 4, no. 3, 2015.

[21] D. T. Pham, T. T. B. Huynh, and T. L. Bui, "A survey on hybridizing genetic algorithm with dynamic programming for solving the traveling salesman problem," in *Proceedings of the International Conference on Soft Computing and Pattern Recognition (SoCPaR '13)*, pp. 66–71, Hanoi, Vietnam, December 2013.

[22] I. Rivin, I. Vardi, and P. Zimmermann, "The *n*-queens problem," *The American Mathematical Monthly*, vol. 101, no. 7, pp. 629–639, 1994.

[23] A. Bruen and R. Dixon, "The *n*-queens problem," *Discrete Mathematics*, vol. 12, no. 4, pp. 393–395, 1975.

[24] C. Letavec and J. Ruggiero, "The *n*-queens problem," *INFORMS Transactions on Education*, vol. 2, no. 3, pp. 101–103, 2002.

[25] R. Sosic and J. Gu, "3,000,000 Queens in less than one minute," *ACM SIGART Bulletin*, vol. 2, no. 2, pp. 22–24, 1991.

[26] M. R. Engelhardt, "A group-based search for solutions of the *n*-queens problem," *Discrete Mathematics*, vol. 307, no. 21, pp. 2535–2551, 2007.

[27] Z. Szaniszlo, M. Tomova, and C. Wyels, "The *N*-queens problem on a symmetric Toeplitz matrix," *Discrete Mathematics*, vol. 309, no. 4, pp. 969–974, 2009.

[28] J. Bell and B. Stevens, "A survey of known results and research areas for *n*-queens," *Discrete Mathematics*, vol. 309, no. 1, pp. 1–31, 2009.

[29] R. Sosič and J. Gu, "Efficient local search with conflict minimization: a case study of the n-queens problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 5, pp. 661–668, 1994.

[30] T. Kwok and K. A. Smith, "Experimental analysis of chaotic neural network models for combinatorial optimization under a unifying framework," *Neural Networks*, vol. 13, no. 7, pp. 731–744, 2000.

[31] N. Funabiki, Y. Takenaka, and S. Nishikawa, "A maximum neural network approach for N-queens problems," *Biological Cybernetics*, vol. 76, no. 4, pp. 251–255, 1997.

[32] J. Mandziuk and B. Macuk, "A neural network designed to solve the N-queens problem," *Biological Cybernetics*, vol. 66, no. 4, pp. 375–379, 1992.

[33] Z. Wang, D. Huang, J. Tan, T. Liu, K. Zhao, and L. Li, "A parallel algorithm for solving the n-queens problem based on inspired computational model," *BioSystems*, vol. 131, pp. 22–29, 2015.

[34] A. Maroosi and R. C. Muniyandi, "Accelerated execution of P systems with active membranes to solve the *N*-queens problem," *Theoretical Computer Science*, vol. 551, pp. 39–54, 2014.

[35] R. Maazallahi, A. Niknafs, and P. Arabkhedri, "A polynomial-time dna computing solution for the *n*-queens problem," *Procedia—Social and Behavioral Sciences*, vol. 83, pp. 622–628, 2013.

[36] A. Draa, S. Meshoul, H. Talbi, and M. Batouche, "A quantum-inspired differential evolution algorithm for solving the N-queens problem," *Neural Networks*, vol. 1, p. 12, 2011.

[37] B. Keswani, *Implementation of n-Queens Puzzle Using Meta-Heuristic Algorithm (Cuckoo Search) [Dissertation]*, Suresh Gyan Vihar University, 2013.

[38] A. A. Shaikh, A. Shah, K. Ali, and A. H. S. Bukhari, "Particle swarm optimization for N-queens problem," *Journal of Advanced Computer Science & Technology*, vol. 1, no. 2, pp. 57–63, 2012.

[39] J. E. A. Heris and M. A. Oskoei, "Modified genetic algorithm for solving n-queens problem," in *Proceedings of the Iranian Conference on Intelligent Systems (ICIS '14)*, Bam, Iran, February 2014.

[40] S. Khan, M. Bilal, M. Sharif, M. Sajid, and R. Baig, "Solution of n-Queen problem using ACO," in *Proceedings of the IEEE 13th International Multitopic Conference (INMIC '09)*, pp. 1–5, IEEE, Islamabad, Pakistan, December 2009.

[41] N. Vaughan, "Swapping algorithm and meta-heuristic solutions for combinatorial optimization n-queens problem," in *Proceedings of the Science and Information Conference (SAI '15)*, pp. 102–104, London, UK, July 2015.

[42] E. Masehian, H. Akbaripour, and N. Mohabbati-Kalejahi, "Landscape analysis and efficient metaheuristics for solving the n-queens problem," *Computational Optimization and Applications*, vol. 56, no. 3, pp. 735–764, 2013.

[43] A. P. Burger and C. M. Mynhardt, "An upper bound for the minimum number of queens covering the $n \times n$ chessboard," *Discrete Applied Mathematics*, vol. 121, no. 1–3, pp. 51–60, 2002.

[44] W. D. Weakley, "Upper bounds for domination numbers of the queen's graph," *Discrete Mathematics*, vol. 242, no. 1–3, pp. 229–243, 2002.

[45] P. Gibbons and J. Webb, "Some new results for the queens domination problem," *Australasian Journal of Combinatorics*, vol. 15, pp. 145–160, 1997.

[46] S. Bozóki, P. Gál, I. Marosi, and W. D. Weakley, "Domination of the rectangular queen's graph," https://arxiv.org/abs/1606.02060.

[47] H. Fernau, "Minimum dominating set of queens: a trivial programming exercise?" *Discrete Applied Mathematics*, vol. 158, no. 4, pp. 308–318, 2010.

[48] N. Mohabbati-Kalejahi, H. Akbaripour, and E. Masehian, "Basic and hybrid imperialist competitive algorithms for solving the non-attacking and non-dominating n-queens problems," in *Computational Intelligence: International Joint Conference, IJCCI 2012 Barcelona, Spain, October 5–7, 2012 Revised Selected Papers*, K. Madani, A. D. Correia, A. Rosa, and J. Filipe, Eds., pp. 79–96, Springer, 2015.

Hindawi
Journal of Optimization
Volume 2017, Article ID 5723239, 11 pages
https://doi.org/10.1155/2017/5723239

Research Article

A NNIA Scheme for Timetabling Problems

Yu Lei and Jiao Shi

School of Electronics and Information, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China

Correspondence should be addressed to Yu Lei; leiy@nwpu.edu.cn

Received 29 December 2016; Revised 6 March 2017; Accepted 16 March 2017; Published 30 May 2017

Academic Editor: Linqiang Pan

This paper presents a memetic multiobjective optimization algorithm based on NNIA for examination timetabling problems. In this paper, the examination timetabling problem is considered as a two-objective optimization problem while it is modeled as a single-objective optimization problem generally. Within the NNIA framework, the special crossover operator is utilized to search in the solution space; two local search techniques are employed to optimize these two objectives and a diversity-keeping strategy which consists of an elitism group operator and an extension optimization operator to ensure a sufficient number of solutions in the pareto front. The proposed algorithm was tested on the most widely used uncapacitated Carter benchmarks. Experimental results prove that the proposed algorithm is a competitive algorithm.

1. Introduction

The examination timetabling problem has long been a challenging area for researchers in the fields of operational research and artificial intelligence, especially at the time that the Toronto benchmark dataset was stated by Carter and Laporte (1996) [1]. The problem has been more difficult because universities are recruiting more students into a larger variety of courses with a growing number of combined degree courses [2] (Merlot et al. 2002). In the past 40 years there are many methods that have been applied to this problem. The represented techniques include constraint-based techniques [3], population-based techniques including genetic algorithms [4], graph coloring techniques [5, 6], ant colony optimization [7], scatter search [8], local search methods including tabu search [9] and simulated annealing [10, 11], variable neighborhood search [12], and hybrid and hyperheuristic approaches [5]. Generally, this problem is modeled as a single-objective optimization problem; only the number of clashes is considered by researchers.

To minimize the number of clashes in an exam timetable, Burke and Newall (2005) [13] stated that the clashes can be eliminated if a large number of periods were allocated. Burke et al. (1998) [14] also stated that longer timetables are needed to decrease the number of clashes. It is obvious that the ETTP is definitely a two-objective optimization problem: the number of clashes and the number of periods. Within the reasonable scope of the number of periods, the number of clashes must be minimized as much as possible. Hence, it is needed to minimize multiple conflicting cost functions, which can be best solved through the method of multiobjective optimization [15] that imported several features from the research on the graph coloring problem and used a variable-length chromosome representation that this paper also adopts.

Evolutionary multiobjective optimization (EMO), whose main goal is to handle multiobjective optimization problems (MOPs), has become a hot topic in the field of evolutionary computation. By simultaneously optimizing more than one objective, Multiobjective Optimization Evolutionary Algorithms (MOEAs) can acquire a set of solutions considering the influence of all the objective functions. Each of those solutions cannot be said better than the other and corresponds to the tradeoffs between those different objectives. Multiobjective examination timetabling problem as a MOP has two contradictory objectives. The optimization of one objective tends to minimize the number of clashes; the other objective tends to decrease the number of time periods. Many MOEAs have been proposed in recent years. Malim et al. (2006) [16] studied three different Artificial Immune systems and indicated that the algorithms can be appropriate for both course and exam timetabling problems. However, after published they were found to represent a mistake in the code, and it is invalid [17].

Many of the existing methods for exam timetabling problems are applicable to single-objective exam timetabling problems. By calculation, these single-objective optimization algorithms only can obtain one result, and the computational efficiency is poor. In this paper, we proposed a novel MOEA-based approach for multiobjective examination timetabling problem. By calculation, multiple results can be obtained by our proposed algorithms. In order to simultaneously optimize the two objectives, we adopt the framework of multiobjective immune algorithm NNIA [18] with some modifications. The NNIA simulates the phenomenon of the multifarious antibodies symbiosis and a small number of antibody's activation in the immune response according to a method of selecting the nondominated neighborhood individuals. It chooses the small number of relatively isolated individuals as active antibodies and clones according to the crowding-distance value and then applies on the operators of recombination and mutation to strengthen the searching of the sparse area of the pareto front. The reasons we adopt the frame of the NNIA are that NNIA is proposed by ourselves and the clone strategy can make the pareto front uniform and get the satisfied solutions. The main contributions are that we adopt elitism group strategy to keep the diversity of the group and the two vertical local search operators to get the optimized solutions. Experiments show that the proposed algorithm is able to find a set of tradeoff solutions between the two objectives.

The paper is organized as follows. Section 2 describes the background information with problem formulation and related works. Section 3 gives a description of the proposed algorithm in detail. Section 4 presents the experimental study. Finally, conclusions are given in Section 5.

2. Background

2.1. Mathematical Model. As previously mentioned, the format of examination timetabling problems described in this paper was first formulated by Carter and Laporte [1] in 1996. In this problem, a set of exams $E = \{e_1, e_2, \ldots, e_{|E|}\}$ need to be scheduled into a set of periods $P = \{1, 2, \ldots, |P|\}$ with each period having a seating capacity s. There are three periods per weekday and a Saturday morning period. No exam is held on Sundays. It is assumed that the exam period starts on a Monday. The problem can be formally specified by first defining the following:

$$\min \quad \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} \sum_{p=1}^{|P|-1} a_{ip} a_{j(p+1)} c_{ij} \tag{1}$$

$$\min \quad |P| \tag{2}$$

$$\text{sub.} \quad \sum_{i=1}^{|E|-1} \sum_{j=i+1}^{|E|} \sum_{p=1}^{|P|-1} a_{ip} a_{jp} c_{ij} = 0, \tag{3}$$

$$\sum_{p=1}^{|P|} a_{ip} \leq 1, \quad \forall i \in \{1, \ldots, |E|\}, \tag{4}$$

where a_{ip} is one if exam e_i is allocated to period p; otherwise, a_{ip} equals zero. c_{ij} is the number of students registered for exams e_i and e_j.

Equations (1) and (2) are the two objectives of minimizing the number of clashes and timetable length, respectively. Equation (3) is the constraint that no student is to be scheduled to take two exams at any one time, while (4) states that every exam can only be scheduled once in any timetable.

To evaluate the quality of one feasible timetable, a function evaluating the average cost for per student based on soft constraints has been proposed. It can be presented as follows:

$$\text{fitness} = \frac{\left(\sum_{s=0}^{4} \omega_s \times N_s\right)}{S}, \tag{5}$$

where $\omega_s = 2^s$ $(s = 0, 1, 2, 3, 4)$ is the weight that represents the importance of scheduling exams with common students either 4, 3, 2, 1, or 0 timeslots away in one timetable and N_s is the number of students involved in the violation of the soft constraint. S is the total number of students in the problem. For this reason that (5) emphasizes the most important indicators, that is, whether the exams in the timetable are allocated throughout the timetable equally, we use this function as one of the objectives in our algorithm. The two objectives of our algorithm optimized are described as follows:

$$\begin{aligned} \min \quad & f_1 = |P| \\ \min \quad & f_2 = \frac{\left(\sum_{s=0}^{4} \omega_s \times N_s\right)}{S}. \end{aligned} \tag{6}$$

2.2. Related Works. The ETTP is a semiannual or annual problem for colleges and is studied by many operational researches widely due to its complexity and utility. There have been proposed a large range of approaches to solve the problem, discussed in the existing literature. These approaches can be classified into the following broad categories [19]: graph-based sequential techniques, local search-based techniques, population-based techniques, and hyperheuristics.

The graph coloring heuristics are one of the earliest algorithms. Welsh and Powell [20] in 1967 proposed a bridge that is built between graph coloring and timetabling and made a great contribution to the field of the timetabling. The five ordering strategies which extended from graph coloring heuristics on examination timetabling problems and a series of examination timetabling problems were introduced by Carter, Laporte, and Lee in 1996, called University of Toronto Benchmark Data. By developing two variants of selection strategies, Burke et al. [21] studied the influence of bringing a random element into the employment of graph heuristics in 1998. These simple strategies showed improved pure graph heuristics on the sides of both the quality and diversity of the solutions when tested on three of the Toronto datasets. Asmuni et al. [22] in 2005 employed fuzzy logic to order the exams to be scheduled on account of graph coloring heuristics on the Toronto datasets and indicated that it is an appropriate evaluation for arranging the exams. Corr et al. [23] investigated a neural network, the objective of which is to arrange the most difficult exams at an early stage of solution construction. The work has showed the feasibility of using neural network as a generally adaptive applicable technique on timetabling problems.

The local search-based techniques represent a large portion of the work which has appeared in the last decade [1]. Mainly because various constraints can be handled relatively easily, they have been applied on a variety of timetabling problems. Di Gaspero and Schaerf (2001) [24] carried out a valuable investigation on a family of tabu search based techniques whose neighborhoods concerned those which contributed to the violations of hard and soft constraints. Burke et al. [5] investigated variants of Variable Neighbourhood Search and obtained the best results in the literature across some of the problems in the Toronto datasets. Caramia et al. [25, 26] developed a fine-tuned local search method where a greedy scheduler assigned examinations into the least timeslots and a penalty decrease improved the timetable without increasing the number of timeslots.

The genetic algorithm is one of the most typical representatives of the population-based techniques. It is noticed that the algorithm has a good performance in the literatures. Particularly, the hybridizations of genetic algorithms with local search methods, memetic algorithms, have an excellent performance in this area. In 1994, Corne et al. [27] introduced genetic algorithms to solve general educational timetabling problems. The function of this work is that certain problem structures in some particularly generated graph coloring problems cannot be handled by obtaining direct representation in the genetic algorithms. Ross et al. [28] in 1996 indicated that by testing on specially generated graph coloring problems of different homogeneity and connectivity the transition regions were existent in solvable timetabling problems. The study can make researchers understand how different algorithms perform on complex timetabling problems. Terashima-Marin et al. [29] in 1999 indicated a clique-based crossover on the timetabling problems which was turned into graph problems. Erben [30] (2001) indicated a grouping genetic algorithm with appropriate encoding, crossover and mutation operators, and fitness functions studied. This method requires less computational time than some of the methods in the literature. Burke and Landa Silva [31] discussed some issues concerning the design of memetic algorithms for scheduling and timetabling problems. Burke et al. [21] developed a memetic algorithm to reassign single exams and sets of exams and employed light and heavy mutation operators. However, neither of these mutations on their own improved the solution quality. Malim et al. [16] developed three variants of Artificial Immune systems and indicated that the algorithms can be suitable for course and exam timetabling problems. However, there was a problem in the results; after publication they were showed to represent an error in the code and invalidness.

More and more researchers pay attention to the hyperheuristics approach. In 2003 Ahmadi et al. [32] investigated a variable neighborhood search, aiming to find good combinations of heuristics for different examination timetabling problem. Kendall and Hussin [33] in 2005 developed a tabu search hyperheuristic; they adopted moving strategies and constructive graph heuristics to be the low level heuristics.

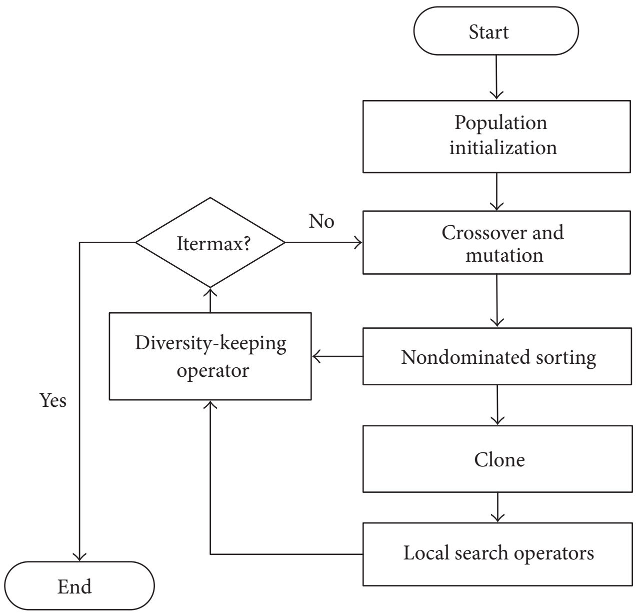


Figure 1: The flow of algorithm.

In 2007 Burke et al. [5] researched obtaining tabu search to find sequences of graph heuristics to construct solutions for timetabling problems and considered the effects of various numbers of low level graph heuristics on the examination timetabling problems. By conducting an empirical study on both benchmark functions and exam timetabling problems, Bilgin et al. [34] (2007) studied 7 heuristic selection methods and 5 acceptance criteria within a hyperheuristic. The memetic algorithm hyperheuristic with a single hill climber at a time showed that it performed better on approaches tested. For the interested readers, more details can be referred from [5].

In summary, during the recent years, there are an increasing number of excellent algorithms; almost all of these algorithms were tested on either benchmark datasets or in real applications, which had made quite good achievements. In this paper, we also proposed a multiobjective optimization algorithm, called Nondominated Neighbor Immune Algorithm (NNIA) in [18]. NNIA adopts an immune inspired operator, a nondominated neighbor-based selection technique, two heuristic search operators, and elitism. It indicates that NNIA is an effective method for solving MOPs by a number of experiments. Due to its good performance, we will adopt the framework of NNIA with some modifications, which will be described in the following section. The contribution of this paper is that we solve this task by using multiobjective optimization technique.

3. The Proposed Algorithm

3.1. Algorithmic Flow of MOEA Based on NNIA. The algorithmic flow of our algorithm is presented in Figure 1. At the start of the algorithm, a conflict matrix C was created according to Burke and Newall [13], which has dimensions $|E|$ by $|E|$ with the definition c_{ij} from Section 2.1 being its (i, j)th element. This matrix can check and eliminate the conflicts in the timetables efficiently.

Because of the large-scale individual of the population, it is inadvisable to search in the normal population. Elitism strategy and crowded selection optimization mechanism are put forward. In our algorithm we adopt elitism strategy two times to reduce the computation burden and extend the range of nondominated solutions in elitism group. In the normal population, the children population after crossover and mutation is mixed with parent population. Then the nondomined solutions of this new population are put into the elitism group. The purpose of the strategy is to offer more nondominated solutions to the elitism group.

3.2. Hyper-Heuristic Initialization. As a common step, initialization is to produce an initial population. In our algorithm, there are a large number of feasible solutions that need to be optimized, but the process of generating the feasible solutions is hard for most of the examination timetabling problems. The difficulty level that feasible solutions generated for different examination timetabling problems is different. It is hard to make some unfeasible solutions be the feasible ones with some conventional ways in the subsequent operation. The result of the algorithm is influenced by the number of feasible solutions in the initial population for some issues. The initialization in our algorithm produces a set of solutions randomly and updates the random solutions to be the feasible ones with the simple genetic algorithm. The details are shown in the following.

The hyperheuristic initialization is that the exams are selected for insertion with the help of some heuristic information when used in the graph coloring problem [21, 35]. The heuristics we use in this paper are as follows:

(1) Largest degree (LD): exams with the largest number of conflicts with other exams are inserted first.

(2) Largest weighted degree (LWD): it is the same as LD but weighted by the number of students involved.

(3) Saturation degree (SD): exams with the fewest valid timeslots, in terms of satisfying the hard constraints, remaining in the timetable are inserted first.

There are two terminated conditions, maximum iteration number and the maximum number of feasible solutions, which can make the whole algorithm achieve the stable result for most examination timetabling problems. But our algorithm has obvious superiority for the problems that the feasible solutions are hardly generated, because of this strategy of the initialization. The advantage of the hyperheuristic initialization is that we can get the timetables with the lengths being close to the demands of the users. The result can be seen in the next section.

3.3. Local Search Operators. Some researchers indicated that adopting local search within evolutionary algorithms is an much effective approach for finding high quality exam timetables which can also contribute to the intensification of the optimization results [36, 37]. A description of the two-direction local search operators this paper adopted is given below.

The first kind is to minimize the timetable length as far as possible without concerning the conflict number, aiming at the operator in the local search to minimize the time periods among the nondominated elitism group.

The selected individual is P, the search depth is SD, the maximum mutation probability is $P_{\max}$, the original mutation probability is P_{ori}, the probability increasing step length is r, the exam timetable is T, the maximum iteration number is Itermax, and the number of examinations is E_{num}.

Step 1. Set the original mutation probability P_{ori}.

Step 2. Set search depth variable sign $V = 0$.

Step 3. Randomly select $\lceil P_{\text{ori}} \times E_{\text{num}} \rceil$ exams in E, which is denoted as E_m. Delete all the selected exams in E.

Step 4. Rearrange the exams in E_m according to the maximum number of conflicts and the timeslots in T according to the heuristics. Then insert the exams in E_m into the timetable T; if you can not insert them into the inherent timeslots, extend the timeslots until all exams are arranged; the newly produced timeslots are T_{new}.

Step 5. If the number of T_{new} is less than the number of T, replace T with T_{new}, and stop; otherwise, $V = V + 1$, and go to Step 3.

Step 6. If V = depth, then $P_{\text{ori}} = P_{\text{ori}} + r$ and go to Step 2.

Step 7. If $P_{\text{ori}} = P_{\max}$, compare the timeslots of T_{new} and T; then put the smaller one into the elitist group and stop; otherwise, go to Step 6 and go on.

The second kind aims at minimizing the number of conflicts without concerning the number of timeslots. The details are described below.

The selected probability is P, the search depth is SD, the maximum selected probability is $P_{\max}$, the original mutation probability is P_{ori}, the probability increasing step length is r, the exam timetable is T, the maximum iteration number is Itermax, and the number of examinations is E_{num}.

Step 1. Set the original examination selected probability $P_{\text{ori}} = P$.

Step 2. Set search depth variable sign $V = 0$.

Step 3. Randomly select $\lceil P_{\text{ori}} \times E_{\text{num}} \rceil$ exams in E and the newly set E_{temp}.

Step 4. Rearrange exams in E_{temp} according to the number of conflicts, delete all the exams in E_{temp}, and sort all the timeslots in the changed timetable T.

Step 5. Insert the exams in E_{temp} into the timetable T at the premise of not affecting the number of timeslots; the newly produced individual is T_{new}.

Step 6. Compare the number of conflicts in T_{new} with which, in T according to the formulation, if the former is

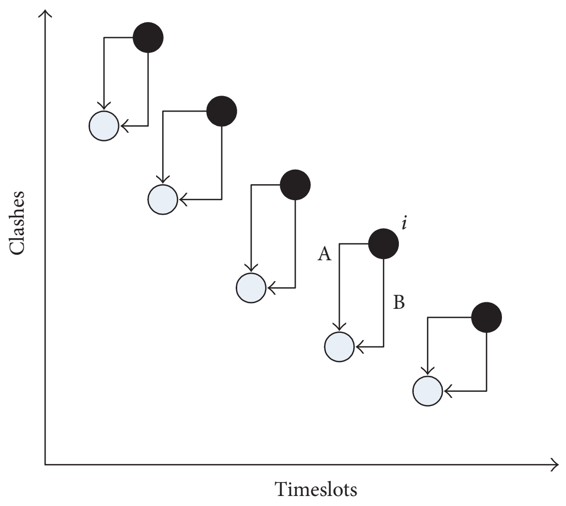


FIGURE 2: Elitism group local search.

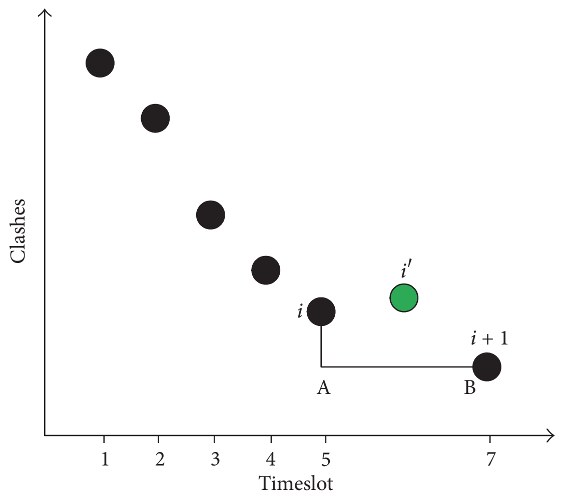


FIGURE 3: The computation of congestion degree.

smaller, replace original individuals with T_{new} and then stop. Otherwise, $V = V + 1$; go to Step 3.

Step 7. If $V = \text{SD}$, judge the number of conflicts in T_{new} and T; if the former is smaller, replace T with T_{new}; put the smaller one into the elitist group; otherwise, $P = P + r$; go to Step 2.

Step 8. If $P = P_{\max}$, judge the number of conflicts in T_{new} and T; if the former is smaller, replace T with T_{new}; then put the smaller one into the elitist group; otherwise, T remain unchanged.

3.4. Diversity-Keeping Operators

3.4.1. Elitism Group Strategy. Although the local search can intensify the optimization results, the discrete optimization is different from the continuous optimization that a small disturbance in decision domain may probably let individuals transform irregularly and even result in deterioration. As such, in order to avoid this phenomenon we put forward a novel local search exploitation with an extra elitism group to save nondominated solutions in every generation. However, normal population is just offering a space of updating the nondominated solutions. In our algorithm we also introduce a corresponding elitism strategy and a crowded selection optimization mechanism; the details will be introduced in the next part.

The local search operators are applied after the strategy of extension and optimization of the elitism group. The operators avoid an objective in an individual deterioration and then minimize the other objective and get the new elitist solutions mixed with original elitist solutions to conduct a nondominated sorting. The frontier may extend to the two different directions as far as possible according to the operators. The local search is searching vertically between two objectives in order which is shown as Figure 2. It indicates that one objective is optimized prior and then the other is shown as rout A or rout B of the individual i in the figure.

3.4.2. Extension Optimization Strategy. Due to the normal selection and mutation operators making a little contribution to the nondominated elitism group, we present a strategy to extend and optimize the elitism group based on the congestion degree which is shown in Figure 3. Compute the difference of timetable length between every individual and its right side one. If the D-value is 1, then extend a time period of this individual and randomly select some exams into the time period. We can get a uniform frontier by this means. As is shown in Figure 3, the crowding degree of individual i is the length of the line segment AB. Assume that the crowding degree of individual i is 2; according to our theory, expend the individual i and get i' and then put it into the original elitism group. Finally, we take local search operators to the generations after extending the time periods.

4. Experimental Analysis

Our algorithm is programmed in Matlab and simulations are performed on the 2.8 GHz Core Personal Computer. We use 9 uncapacitated benchmark examinations timetabling datasets proposed by Carter and Laporte [1] to evaluate the effectiveness of our algorithm. The details of the proposed benchmarks are shown in Table 1. As no dataset is designed to evaluate the multiobjective timetabling algorithms, we just use the datasets used in the single objective to evaluate the feasibility and reasonableness of our algorithm. The parameter settings are presented in Table 2. The population size is 100 and the maximum iteration is 100. The other parameters' choosing reason will be described in the following parts.

In the following sections, we will study our algorithm in two sides. One is to make analysis on the contribution of diversity-keeping local search operators searching in two different directions orderly which is shown with four comparative experiments. The other one is to discuss the contribution of elitism group strategy applied on our algorithm two times with four experiments presented.

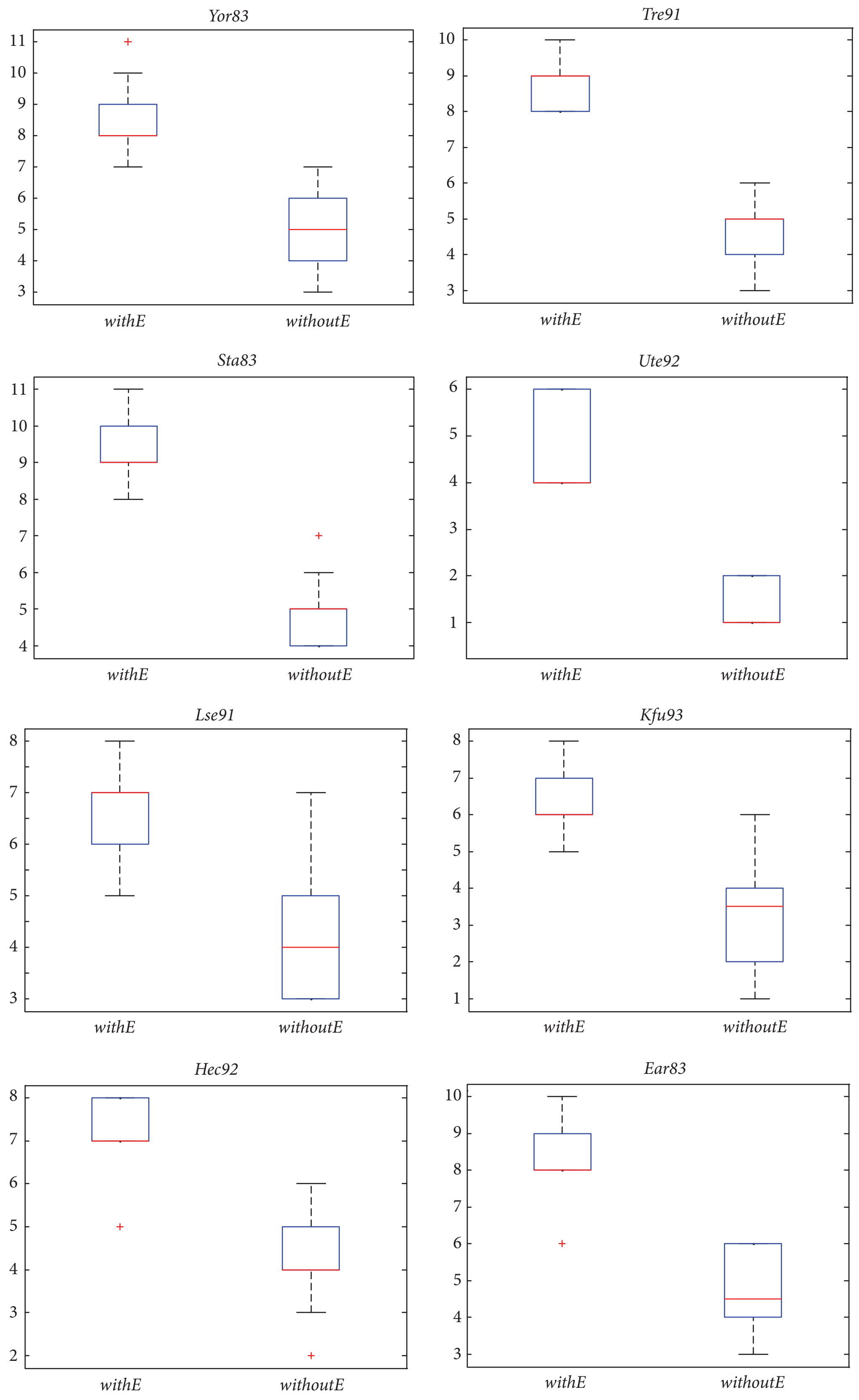


Figure 4: Performance comparison for MOEA with and without diversity-keeping strategy.

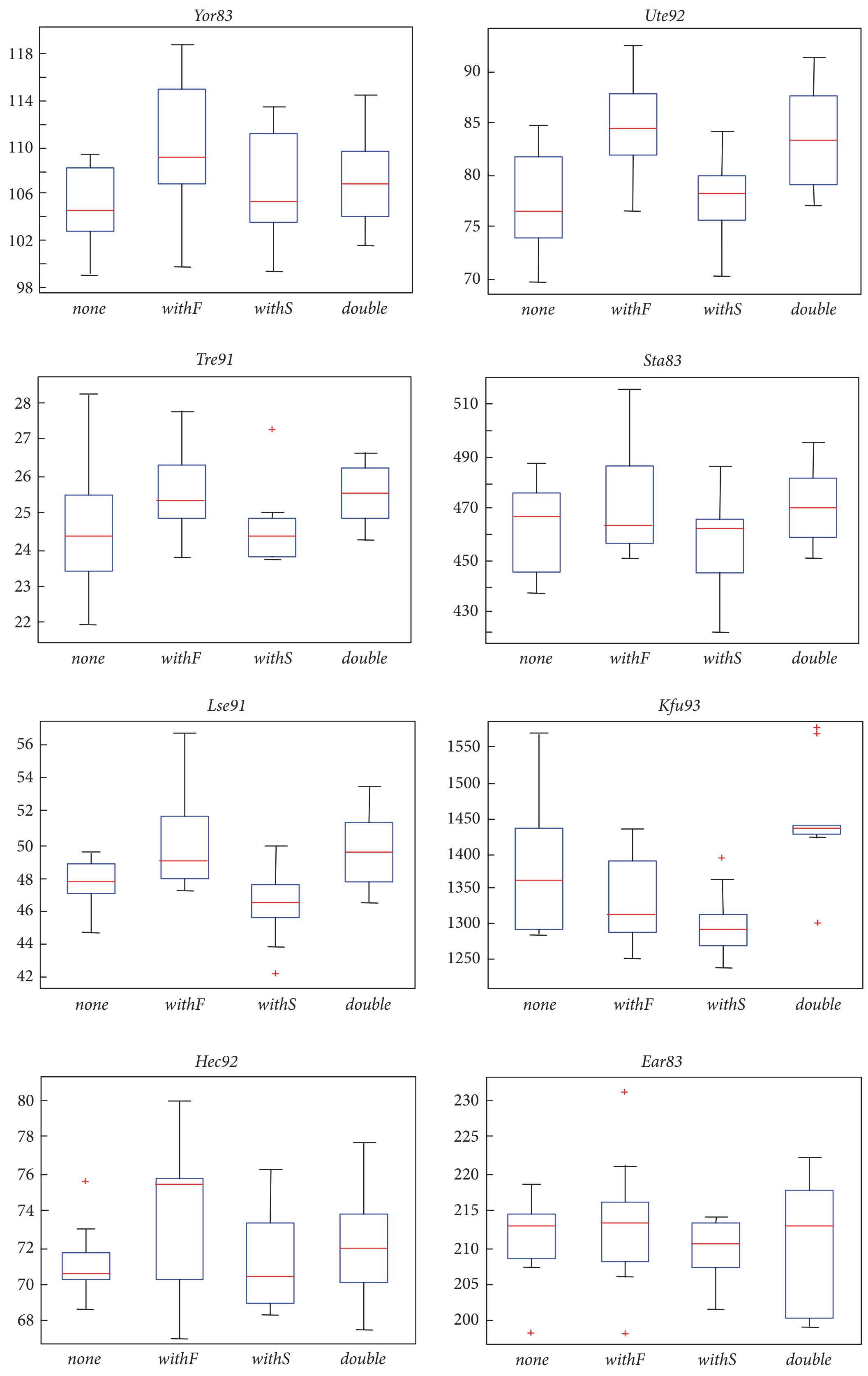


Figure 5: Performance comparison for MOEA with different local search settings.

Table 1: Characteristics of datasets.

Dataset	Timeslots	Exams	Students	Conflict density
Car 91	35	682	16925	0.13
Car 92	32	543	18419	0.14
Ear 83	24	190	1125	0.27
Hec 92	18	81	2823	0.42
Kfu 93	20	461	5349	0.6
Lse 91	18	381	2726	0.6
Rye 92	23	486	11483	0.08
Sta 83	13	139	611	0.14
Tre 92	23	261	4360	0.18
Ute 92	10	184	2750	0.08
York 83	3521	181	941	0.29

Table 2: Parameter setting for simulation study.

Parameter	Value
Population size	100
E: the number of exams scheduled by each heuristic	2
P_c: the crossover probability	0.8
P_m: the mutation probability	0.2
SD: the search depth of the local search	10
Itermax: the maximum iteration number	500

4.1. Contribution of Diversity-Keeping Strategy. This section presents the performance of the diversity-keeping operators. To assess the effectiveness of the strategies, a comparison was conducted as in Figure 4. *withE* is saying that the algorithm runs with the strategies while *withoutE* is indicating that the algorithm runs without the strategies. The boxplots were drawn according to the statistic number of solutions. The eight data in this experiment were running ten times independently. From Figure 4, it can be observed obviously that the operator of *withE* does better than *withoutE* does in every dataset. The results well proved the efficiency of diversity-keeping operator.

4.2. Contribution of Local Search Operators. In this section, we use the hypervolume as an indicator to estimate the effectiveness of the algorithm; in our comparable experiments the indicator r_p is the maximum value of the dataset to be compared in all dimensions.

To prove the efficiency of the proposed two local search operators, this section shows the performance of the algorithms with and without local search operators. As is shown in Figure 5, *none* is the setting that does not use local search at all, and *withF* is the setting with the local search aiming at minimizing the conflict number, while the setting *withS* is the local search to minimize the timeslots and *double* incorporates two local search operators. Ten independent runs of the four settings are conducted to obtain statistical results. From Figure 5, the contribution of local search operators is obvious, since the operators which use two local search are able to generate solutions with significantly lower number of clashes. For the hypervolume value of the nondominated

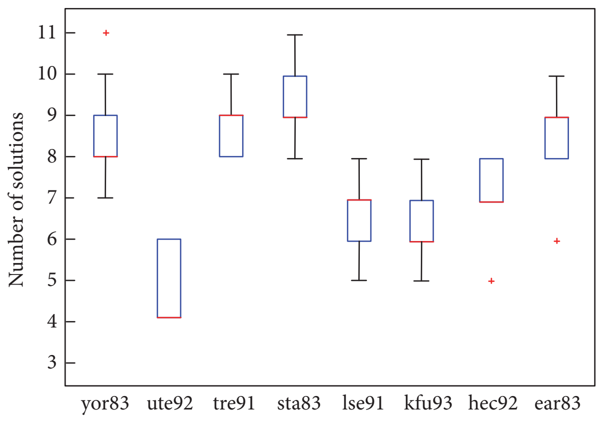


Figure 6: Number of pareto optimal solutions for the datasets.

solutions of datasets Ute92, Sta83, Lse91, Kfu93, Hec92, and Ear83, the application of the two local search settings shows better results than other three settings.

From the statistic boxplots we can see that our algorithm is robust to the indicator of the students conflict numbers. The outliers are few and the differences between the highest and the lowest values are small, which also demonstrate the robustness of our algorithm.

4.3. Performance of Multiobjective Algorithm Based on NNIA. This section presents the multiobjective optimization performance of the algorithm based on NNIA. On top of showing the advantages of our algorithm, the role of the two objectives will be validated as follows. The experiment was conducted running ten times independently.

The boxplots in Figure 6 show the number of pareto optimal solutions of the eight datasets. The number of pareto optimal solutions for the dataset *ute*92 is nearly four. The number of pareto optimal solutions for other datasets is almost distributed from seven to ten.

Experiments were conducted to further show the results. From Figure 7, we can see the details of both the number of periods and its corresponding clashes. All of the datasets tested perform good, the pareto optimal solutions are distributed uniformly, and the clashes are relatively small.

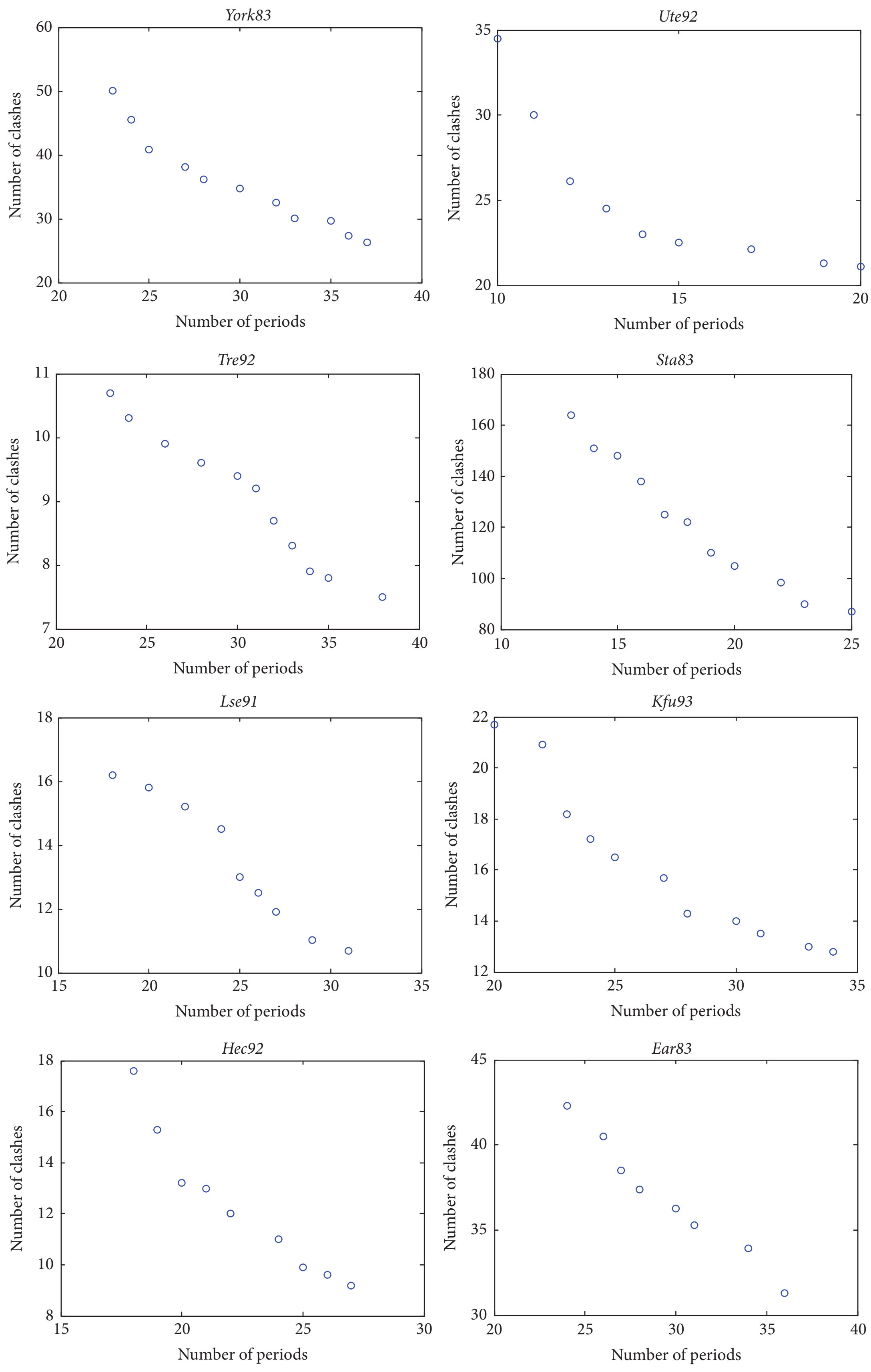


Figure 7: Pareto optimal solutions for the datasets.

The experiments support our algorithm powerfully and the multiobjective exam timetabling problem is solved well.

5. Conclusions

In this paper, the exam timetabling problem has been regarded as a multiobjective optimization problem which involves the minimizing of the number of clashes and number of periods in a timetable. A multiobjective evolutionary algorithm, based on NNIA, featured with elitism group strategy, congestion degree based on extension optimization strategy, and two local search operators, has been presented.

The proposed MOEA is different from most existing single-objective-based methods in the fact that it optimizes two objectives concurrently and get a set of solutions reasonable instead of producing single-length timetables. It has been proved that such an approach is more universal and would be able to function effectively. The results also show that the algorithm can generate relatively short clash-free timetables and various solutions which are convenient for deciders to choose on their own preference.

The work we do in this paper focuses on the temporal aspect of the ETTP, which has solved the problem well in a sense. However, it still has some shortcomings, how to balance the diversity and approximation, which can be subjected for future study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. W. Carter and G. Laporte, "Recent developments in practical examina-tion timetabling," in *Proceedings of the 1st International Conference on Practice and Theoryof Automated Timetabling*, E. K. Burke and P. Ross, Eds., vol. 1153 of *Springer Lecture Notes in Computer Science*, pp. 3–21, 1996.

[2] L. T. G. Merlot, N. Boland, B. D. Hughes, and P. J. Stuckey, "A hybrid algorithm for the examination timetabling problem," in *Proceedings of the 4th Internationalconference on the Practice and Theory of Automated Timetabling (PATAT '02)*, K. Burke and P. De Causmaecker, Eds., vol. 2740 of *Lecture Notes in Computer Science*, pp. 207–231, Springer, Berlin, Germany, 2002.

[3] T. Müller, "ITC2007 solver description: a hybrid approach," *Annals of Operations Research*, vol. 172, no. 1, pp. 429–446, 2009.

[4] P. Ross, E. Hart, and E. Corne, "Some observations about GAbasedexam timetabling," in *Proceedings of the 2nd International Conference on Practiceand Theory of Automated Timetabling*, E. K. Burke and M. W. Carter, Eds., vol. 1408 of *Lecture Notes in Computer Science*, pp. 115–129, Springer, 1998.

[5] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, no. 1, pp. 177–192, 2007.

[6] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "Roulette wheel graph colouring for solving examination timetabling problems," in *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications*, vol. 5573 of *Lecture Notes in Comput. Sci.*, pp. 463–470, Springer, 2009.

[7] M. Eley, "Ant algorithms for the exam timetabling problem," in *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT '07)*, E. K. Burke and H. Rudova, Eds., vol. 3867 of *Lecture Notes in Computer Science (2007)*, pp. 364–382, Springer.

[8] N. Mansour, V. Isahakian, and I. Ghalayini, "Scatter search technique for exam timetabling," *Applied Intelligence*, vol. 34, no. 2, pp. 299–310, 2011.

[9] G. De Smet, "ITC2007—Examination Track," in *Proceedings of the Practice and Theory of Automated Timetabling (PATAT '08)*, Montreal, Canada, 2008.

[10] E. Burke, Y. Bykov, J. Newall, and S. Petrovic, "A time-predefined local search approach to exam timetabling problems," *IIE Transactions*, vol. 36, no. 6, pp. 509–528, 2004.

[11] J. M. Thompson and K. A. Dowsland, "A robust simulated annealing based examination timetabling system," *Computers and Operations Research*, vol. 25, pp. 637–648, 1998.

[12] R. Qu and E. K. Burke, "Hybrid variable neighbourhood hypeheuristics for exam timetabling problems," in *Proceedings of the MIC2005: The Sixth Meta-Heuristics International Conference*, Vienna , Austria, 2005.

[13] E. K. Burke and J. P. Newall, "Solving examination timetabling-problems through adaptation of heuristic orderings," *Annals of Op-erational Research*, vol. 129, pp. 107–134, 2005.

[14] E. K. Burke, D. G. Elliman, P. H. Ford, and R. F. Weare, "Specialisedrecombinative operators for the timetabling problem," in *Evolutionary Computing: AISB Workshop*, pp. 75–85, Springer, Sheffield, UK, 1998.

[15] C. Y. Cheong, K. C. Tan, and B. Veeravalli, "A multi-objective evolutionary algorithm for examination timetabling," *Journal of Scheduling*, vol. 12, no. 2, pp. 121–146, 2009.

[16] M. R. Malim, A. T. Khader, and A. Mustafa, "Artificial immune-algorithms for university timetabling," in *Proceedings of The 6th International Conference on Prac-Tice and Theory of Automated Timetabling*, E. K. Burke and H. Rudova, Eds., pp. 234–245, Brno, Czech Republic, August 2006.

[17] R. Qu, E. K. Burke, B. McCollum, L. T. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling*, vol. 12, no. 1, pp. 55–89, 2009.

[18] M. Gong, L. Jiao, H. Du, and L. Bo, "Multiobjective immune algorithm with nondominated neighbor-based selection," *Evolutionary Computation*, vol. 16, no. 2, pp. 225–255, 2008.

[19] E. K. Burke, J. Kingston, and D. de Werra, "Applications to timetabling," in *Handbook of Graph Theory*, J. Gross and J. Yellen, Eds., pp. 445–474, Chapman Hall, London, UK, 2004.

[20] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of agraph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85-86, 1967.

[21] E. K. Burke, J. P. Newall, and R. F. Weare, "A memeticalgorithm for university exam timetabling," in *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT '95)*, E. K. Burke and P. Ross, Eds., vol. 1153 of *Lecture Notes in Computer Science*, pp. 241–250, Springer, Edinburgh, Scotland, 1996.

[22] H. Asmuni, E. K. Burke, J. Garibaldi, and B. McCollum, "Fuzzy multipleordering criteria for examination timetabling," in *Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling*, E. K. Burke and M. Trick, Eds., vol. 3616 of *Springer Lecture Notes in Computer Science*, pp. 334–353, 2005.

[23] P. H. Corr, B. McCollum, M. A. J. McGreevy, and P. McMullan, "A newneural network based construction heuristic for the examination timetablingproblem," in *Proceedings of the International Conference on Parallel Problem Solving From Nature (PPSN)*, pp. 392–401, Springer, Reykjavik, Iceland, September 2006.

[24] L. Di Gaspero and A. Schaerf, "Tabu search techniques for examination timetabling," in *Proceedings of the 3rd International Conference on Practice and Theory of Automated Timetabling III*, E. K. Burke and W. Erben, Eds., vol. 2079 of *Lecture Notes in Computer Science*, pp. 104–117, Springer, Berlin, Germany, 2001.

[25] M. Caramia, P. DellOlmo, and G. F. Italiano, "New algorithms for examination timetabling," in *Proceedings of the 4th International Workshop, on Algorithm Engineering (WAE 2000)*, S. Naher and D. Wagner, Eds., vol. 1982 of *Lecture Notes in Computer Science*, pp. 230–241, Springer, Berlin, Germany, 2000.

[26] M. Caramia, P. Dell'Olmo, and G. F. Italiano, "Novel local-search-based approaches to university examination timetabling," *INFORMS Journal on Computing*, vol. 20, no. 1, pp. 86–99, 2008.

[27] D. Corne, P. Ross, and H. Fang, "Evolutionary timetabling: Practice, prospects and work in progress," in *Proceedings of the UK Planning and Scheduling SIG Workshop*, P. Prosser, Ed., 1994.

[28] P. Ross, D. Corne, and H. Terashima-Marin, "The phase transition niche for evolutionary algorithms in timetabling," in *Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling*, E. K. Burke and P. Ross, Eds., vol. 1153 of *Lecture Notes in Computer Science*, pp. 309–324, 1996.

[29] H. Terashima-Marin, P. Ross, and M. Valenzuela-Rendon, "Clique-based crossover for solving the timetabling problem with GAs," in *Proceedings of the Congress on Evolutionary Computation, CEC 1999*, pp. 1200–1206, Washington, DC, USA, July 1999.

[30] W. Erben, "A grouping genetic algorithm for graph colouring and examtimetabling," in *Proceedings of the 3rd International Conference on Practice and Theory of Automated Timetabling*, E. K. Burke and W. Erben, Eds., vol. 2079 of *Lecture Notes in Computer Science*, pp. 132–156, Springer, Berlin, Germany, 2001.

[31] E. K. Burke and J. D. Landa Silva, "He design of memetic algorithms forscheduling and timetabling problems," in *Proceedings of the Recent Advances in Memetic Algorithms and Related Search Technologies. Studies in Fuzziness and Soft Computing 166*, W. E. Hart, N. Krasnogor, and J. E. Smith, Eds., pp. 289–312, Springer, Berlin, Germany, 2004.

[32] S. Ahmadi, R. Barone, P. Cheng, P. Cowling, and B. McCollum, "Erturbation based variable neighbourhood search in heuristic space for examination time tabling problem," in *Proceedings of the Multidisciplinary International Scheduling: Theory and Applications (MISTA '03)*, pp. 155–171, Nottingham, UK, August 2003.

[33] G. Kendall and N. M. Hussin, "A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA university of technology," in *Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling*, E. K. Burke and M. Trick, Eds., vol. 3616 of *Lecture Notes in Computer Science*, pp. 199–218, 2005.

[34] B. Bilgin, E. Ozcan, and E. E. Korkmaz, "An experimental study on hyper-heuristics and exam timetabling," in *Proceedings of the 6th International Conference on Practice and Theory of Automated Timetabling*, E. K. Burke and H. Rudova, Eds., vol. 3867 of *Lecture Notes in Computer Science*, pp. 394–412, Springer, 2007.

[35] E. K. Burke, D. G. Elliman, P. H. Ford, and R. F. Weare, "Examination timetabling in British universitiesa survey," in *Proceedings of The 1st International Conference on the Practice and Theory of Automated Timetabling (PATAT '96)*, E. K. Burke and P. Ross, Eds., vol. 1153 of *Lecture Notes in Computer Science*, pp. 76–90, Springer, Edinburgh, Scotland, 1996.

[36] Y. Lei, M. Gong, L. Jiao, W. Li, Y. Zuo, and Q. Cai, "A double evolutionary pool memetic algorithm for examination timetabling problems," *Mathematical Problems in Engineering*, vol. 2014, Article ID 867645, 13 pages, 2014.

[37] Y. Lei, M. Gong, J. Zhang, W. Li, and L. Jiao, "Resource allocation model and double-sphere crowding distance for evolutionary multi-objective optimization," *European Journal of Operational Research*, vol. 234, no. 1, pp. 197–208, 2014.

Hindawi
Journal of Optimization
Volume 2017, Article ID 4685923, 9 pages
https://doi.org/10.1155/2017/4685923

Research Article

A Novel Distributed Quantum-Behaved Particle Swarm Optimization

Yangyang Li,[1] Zhenghan Chen,[1] Yang Wang,[1] Licheng Jiao,[1] and Yu Xue[2]

[1]*Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Joint International Research Laboratory of Intelligent Perception and Computation, Xidian University, Xi'an, Shaanxi Province 710071, China*
[2]*School of Computer and Software, Nanjing University of Information Science and Technology (NUIST), Nanjing 210044, China*

Correspondence should be addressed to Yangyang Li; lyy_791@163.com

Received 29 December 2016; Accepted 4 April 2017; Published 3 May 2017

Academic Editor: Gexiang Zhang

Quantum-behaved particle swarm optimization (QPSO) is an improved version of particle swarm optimization (PSO) and has shown superior performance on many optimization problems. But for now, it may not always satisfy the situations. Nowadays, problems become larger and more complex, and most serial optimization algorithms cannot deal with the problem or need plenty of computing cost. Fortunately, as an effective model in dealing with problems with big data which need huge computation, MapReduce has been widely used in many areas. In this paper, we implement QPSO on MapReduce model and propose MapReduce quantum-behaved particle swarm optimization (MRQPSO) which achieves parallel and distributed QPSO. Comparisons are made between MRQPSO and QPSO on some test problems and nonlinear equation systems. The results show that MRQPSO could complete computing task with less time. Meanwhile, from the view of optimization performance, MRQPSO outperforms QPSO in many cases.

1. Introduction

With the development of information science, more and more data is stored, such as web content and bioinformatics data. For this reason, many basic problems have become more and more complex, which makes great troubles to current intelligent algorithm. As one of the most important issues in artificial intelligence, optimization problem in real-world applications also becomes harder and harder to be solved.

In the past 30 years, evolutionary algorithm is becoming one of the most effective intelligent optimization methods. In order to face the new challenge, distributed evolutionary algorithms (dEAs) have been blossomed rapidly. The paper [1] provides a comprehensive survey of the distributed EAs and some models are summarized. Master-slave, island, cellular, hierarchical, pool, coevolution, and multiagent models are listed and introduced. And the different models are analyzed from aspects like parallelism level, communication cost, scalability, and fault-tolerance. And some hotspots of dEAs, such as cloud and MapReduce-based implementations and GPU and CUDA-based implementations, are listed. But no results of dEAs on distributed computing devices are reported. Cloud can be applied in many aspects, and [2–8] have realized various specific applications of cloud. The paper [9] gives a review of the parallel and distributed genetic algorithms in graphics processing unit (GPU). Some works along this idea are reported, such as [10–12]. Cloud and MapReduce is a new and effective technology to deal with big data, which is proposed by Google in 2004 [13]. To respond to the requirement of parallelization and distribution, this physical platform is very convenient to deploy an algorithm to update to be parallel. The programmers only need to consider the map function and reduce function, and the other details are provided by the model itself. Many practical problems are solved with MapReduce model and cluster of servers, such as path problem in large-scale networks [14], seismic signal analysis [15], image segmentation [16], and location recommendation [17]. But the study about MapReduce-based

EAs is still in initial stage. Although some genetic algorithms [18–23] and particle swarm optimization realized by MapReduce [24] have been proposed. There are still many kinds of EAs which are not implemented with distributed model and parallel potential of these algorithms is not released. Based on these considerations, in our previous work [25], MapReduce is combined with coevolutionary particle swarm optimization, which shows that MapReduce-based CPSO obtain much better performance than CPSO. In another work [26], the quantum-behaved particle swarm optimization is transformed on MapReduce successfully. And the idea of this paper is based on it and continues extending that the background is introduced and a practical application is added.

Quantum mechanics and trajectory analysis gained extensive attention of scholars recently and sparkled in many areas, such as image segmentation [27], neural network [28], and population-based algorithms [29, 30]. In [31], Zhang presents a systematic review of quantum-inspired evolutionary algorithms. Quantum-behaved particle swarm optimization is a kind of PSO proposed by Sun et al. in 2004 [32]. Inspired by movement of particle in quantum space, a new reproduction operator of solution is proposed in this algorithm. Because a particle could arrives at any location in quantum space with a certain probability, a new solution at any location in feasible space also could be generated with a certain probability in QPSO. This mechanism is helpful for particles to avoid falling in local optimum. Some more detailed analysis has been reported in [33]. Unfortunately, when the algorithm faces large-scale and complex problem, the increasing computational cost becomes the bottleneck of this algorithm and without enough computing resource premature phenomenon could not been avoided, which urges the original algorithm to be parallel.

In order to follow this trend and enhance the capabilities of a standard QPSO, the MapReduce quantum-behaved particle swarm optimization is developed. The MRQPSO transplants the QPSO on MapReduce model and makes the QPSO parallel and distributed through partitioning the search space. Through the comparisons of MRQPSO and standard QPSO, it could be found that the proposed MRQPSO decreases the time of same function evaluations. And on some test problems QPSO increases the performance of solution and is more robust than QPSO.

The rest of this paper is organized as follows. Section 2 introduces the PSO and QPSO. Section 3 gives a brief presentation of MapReduce model. Section 4 describes the details of QPSO implementing on MapReduce. In Section 5, we show and analyze results of experiments, including the comparison with QPSO. Finally Section 6 concludes the work in this paper.

2. PSO and QPSO

2.1. The Particle Swarm Optimization Algorithm. Inspired by bird and fish flocks, Kennedy and Eberhart proposed PSO algorithm in 1995 [34]. This algorithm is a population-based intelligent search algorithm. In order to find the food as quickly as possible, the birds in a flock would trace their companions that are near to the food firstly. Then they would determine accurate area of food. The individual of PSO searches the optimum like the bird in a flock. Each particle has velocity and position. And the two parameters would be updated according to best value and global best value of the particles. The velocity and position of particle i at the dimension j are presented by v_{ij} and x_{ij}, respectively. The updating equation could be described as

$$\begin{aligned} v_{ij}(t+1) &= wv_{ij}(t) + c_1 r_1 \left(pB_{ij}(t) - x_{ij}(t)\right) \\ &\quad + c_2 r_2 \left(gB_{ij}(t) - x_{ij}(t)\right), \\ x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1), \end{aligned} \tag{1}$$

where $v_{ij}(t)$ and $x_{ij}(t)$ are the velocity and position. t represents the tth iteration. pB and gB are best value and global best value of the particle, respectively. r_1 and r_2 are random number uniformly distributed in $[0, 1]$. w, c_1, and c_2 are three parameters of the algorithm. w is initial weight proposed by Shi and Eberhart in 1998 [35] to control the balance of local and global optimum. c_1 and c_2 are the accelerated coefficients or learning factors. Usually, $c_1 = c_2 = 2$.

From the above equations, it could be found that few parameters are used in PSO, which makes PSO easy to be controlled and used. Meanwhile, it has better convergence performance and quicker convergence speed. These advantages make the PSO algorithm gain a lot of research attention. However, the PSO is not a global optimization algorithm [36]. The limited velocity constrains the search space in a limited area. So the PSO could not always find the global optimum. In other words, the premature convergence is the most serious drawback of the PSO.

2.2. The Quantum-Behaved Particle Swarm Optimization Algorithm. To overcome the shortcoming of the original PSO algorithm, Sun et al. proposed the quantum-behaved particle swarm optimization (QPSO) in 2004 [32]. This algorithm has a more superior performance comparing to the PSO. QPSO algorithm transfers the search space from classical space to quantum space. Particles can appear at any position, which implement the full search in the solution space.

According to uncertainty principle, the velocity and position of a particle cannot be determined simultaneously. In quantum space, a probability function of the position where a particle appears could be obtained from Schrodinger equation. The true position of one particle could be measured by Monte Carlo method. Based on these ideas, in QPSO, a local attractor is constructed by particle best solutions and global best solution as (2) for each particle.

$$P_{ij}(t+1) = \phi_j(t) * pB_{ij}(t) + \left(1 - \phi_j\right) * gB_{ij}(t), \tag{2}$$

where $P_{ij}(t)$ is a local attractor of the particle i at the dimension j. $\varphi_j(t)$ is random number distributed in $[0, 1]$. $pB_{ij}(t)$ is the particle best solution. $gB_{ij}(t)$ is the current global best solution.

The position of the particle is updated by

$$X_{ij}(t+1) = P_{ij}(t+1) \pm \alpha * \left| mbest - X_{ij}(t) \right| * \ln\left(\frac{1}{u}\right), \tag{3}$$

where α is the only parameter in the algorithm called creativity coefficient, which is a positive real number, to adjust the balance of local and global search. The definition of α refers to (4). $\text{Iter}_{\max}$ is the maximum number of iterations. u is random number distributed in $[0, 1]$, and *mbest* is the mean position and defined as follows:

$$\alpha = \frac{(1-0.5) * (\text{Iter}_{\max} - t)}{\text{Iter}_{\max}} + 0.5, \tag{4}$$

$$mbest = \left(\frac{1}{M}\sum_{i=1}^{M} P_{i1}, \frac{1}{M}\sum_{i=1}^{M} P_{i2}, \ldots, \frac{1}{M}\sum_{i=1}^{M} P_{id}, \right), \tag{5}$$

where M is the size of population. P_i is the global extremum of particle i.

In QPSO, first step is initializing the population randomly, which concludes the position of each particle, particle best value, and global best value. Next, calculate the mean position of jth dimension according to (5). Then, particle is evaluated again and the best and global best solution of the particles would be updated according to the fitness value. After that, the particle is updated as (2) and (3). When the number of iterations or accuracy requirements is satisfied, stop running and output the optimum.

Although the QPSO algorithm is superior to PSO, it still has some disadvantages. Because the particles in the QPSO fly discretely, the narrow area where the optimum is may be missed. In the case of too much computation, QPSO may spend too much time.

3. MapReduce

MapReduce [13] is a programming model proposed by Dean and Ghemawat. Inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages, this model is created for processing the large-scale data in parallel. The infrastructure of MapReduce provides detailed implementations of communications, load balancing, fault-tolerance, resource allocation, file distribution, and so forth [1]. Programmers do not need a lot of knowledge and experiments about parallel and distributed programming. They only need to pay attention to *map* and *reduce* function which the model consists of and then can implement algorithm to parallelization easily.

In this model, the computation takes a set of key/value pairs. The *map* function processes the *input* key/value pairs and then emits new lists of key/value pairs, called *intermediate* key/value pairs. The type of these two lists may be from different domain. The *map* function is called independently, and the parallelization is implemented in this way. After all *map* functions' processions are completed, the *reduce* function is called. *Intermediate* key/value pairs are grouped and passed to reduce function. The reduce phase merges and integrates these intermediate key/value pairs and outputs the *output* key/value pairs finally. And the type of *intermediate* and *output* pairs must be the same. The type of *map* and *reduce* functions can be written as follows:

$$\begin{aligned} \text{map: } & (k1, v1) \longrightarrow \text{list}(k2, v2), \\ \text{reduce: } & (k2, \text{list}(v2)) \longrightarrow \text{list}(v2). \end{aligned} \tag{6}$$

Because Google has not released the system to the public, Hadoop, the Apache Lucene project developed, has been used generally. This Java-based open-source platform is a clone of MapReduce infrastructure, and we can use it to design and implement our distributed computation.

4. The MRQPSO Algorithm

The particle swarm optimization algorithm [34] is one of the popular evolutionary algorithms. It has attracted much attention because of the merits of simple concept, rapid convergence, and good quality of solution. However, this algorithm is bothered by some weakness, such as premature phenomenon. Focusing on the shortcoming of original PSO, Sun et al. proposed an uncertain and global random algorithm named quantum-behaved particle swarm optimization (QPSO) in 2004 [32]. The new one puts the search space into quantum space to let the particle move to any location with different probability. Through this strategy, the premature phenomenon could be solved to a certain degree.

Although the QPSO has satisfying progress on premature phenomenon, it has not been prepared to challenge of problems with complex landscape or needing huge computation to be solved. Due to the particles of the QPSO flying discretely, they may miss the narrow area where the global optimum is. And as the problem is getting complex, the computational cost increases. So we implement the QPSO parallel and distributed by transplanting the algorithm on MapReduce model and we name this algorithm MRQPSO. The framework of MRQPSO could be described as in Algorithm 1, and the flowchart is shown as in Figure 1.

The proposed MRQPSO partitions the search space into many subspaces. For n-dimension search space, the range of each dimension is cut to be m_i parts ($0 < i < n$); then the space-partition is completed, and $m_1 * m_2 * \cdots * m_n$ subspace is obtained [25]. Then using several servers, several mappers perform QPSO on different subspace in parallel and independently. After all the mappers finished the calculation, the reducer merges and integrates the immediate value and outputs the best solution. The space-partition helps the particle get distributed uniformly, which ensures all areas have the particles fall in at the initialization phase. It is effective to avoid the particles overflying the narrow zone where the optimum may lie. And the parallel mappers could help MRQPSO to save time cost.

4.1. MRQPSO Map Function. Algorithm 1 shows the pseudo code of the map function of proposed MRQPSO. *pbest* is the position of particle with best value and *gbest* is the

```
Step 1. Divide feasible space into several subspace;
Step 2. Construct Mapper which performs QPSO on one subspace and outputs the obtained optimal
solution on this subspace;
Step 3. Construct Reducer which selects the best optimal solution on different subspace from mapper;
Step 4. Output the best optimal solution and its functional value.
```

Algorithm 1: Framework of MRQPSO.

```
function mapper (key, value) {
    initialize the positions of all particles
    evaluate the function values of positions then select the pbest and gbest
    // update the particle
    while the termination condition is not met {
        calculate the mbest and $\alpha$
        for each particle {
            update the pbest and gbest
            for each dimension {
                update position
            }
        }
        calculation + 1
    }
    emit a message (ID of gbest, string of gbest and fitness)
}
```

Algorithm 2: MRQPSO map.

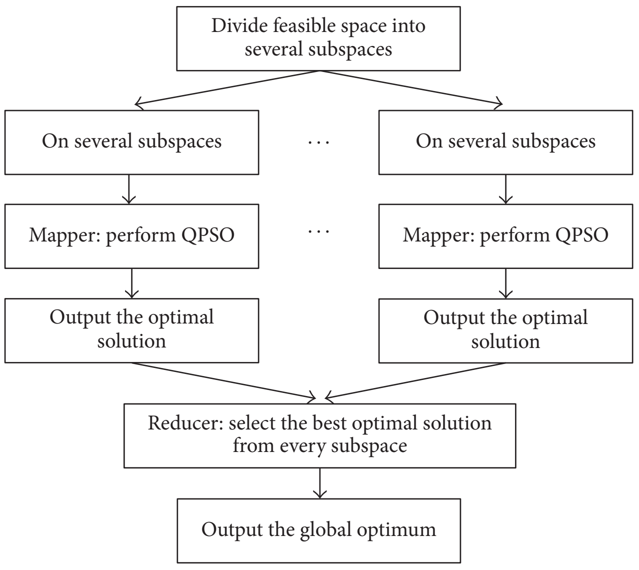

Divide feasible space into several subspaces
On several subspaces
...
On several subspaces
Mapper: perform QPSO
...
Mapper: perform QPSO
Output the optimal solution
Output the optimal solution
Reducer: select the best optimal solution from every subspace
Output the global optimum

Figure 1: Flowchart of MRQPSO.

position of solution with global best value. Several subspaces are saved as records and form data block. The mapper is called when a block starts a QPSO procedure. The input key/value pairs denote the massages of data block. The key is the ID of one record and the value is the string of search space. Then the mappers start to process the QPSO in every block independently. Once a block has been explored, another block will follow up immediately. Under ideal conditions, the larger the number of mappers is, the fewer process a single mapper processes, and the fuller the parallelization is. However, the mapper would spend time to be started in fact. If the data is big enough, the starting time can be neglected. But in our experiments, it will influence the results more or less.

After being processed by mappers, the immediate key/value pairs change to denote the information of *gbest* and global optimum of current data block, showed as Algorithm 2. And then the immediate key/value pairs are ready to transport to the reduce phase.

4.2. MRQPSO Reduce Function. The reduce function is in charge of merging and integrating the information which the mapper emitted. As Algorithm 3 shows, the reducer of MRQPSO is used to select the minimum from all subspaces. The mappers produced and transported the immediate key/value pairs which are received by the reducer after all mappers completed their work. At the reduce phase, all *gbest* and corresponding fitness of blocks are compared with each other. And the minimum of them is selected and outputted finally.

5. Experiment Result and Analysis

5.1. Performance of MRQPSO on Test Problems. To validate the proposed MRQPSO algorithm, we selected 8 functions to evaluate the ability of solving complex problems firstly.

Table 1: Benchmark functions used in this paper.

	Function	Description
Composition functions	F1	Composition function 1 (n = 5, rotated)
	F2	Composition function 2 (n = 3, unrotated)
	F3	Composition function 3 (n = 3, rotated)
	F4	Composition function 4 (n = 3, rotated)
	F5	Composition function 5 (n = 3, rotated)
	F6	Composition function 6 (n = 5, rotated)
	F7	Composition function 7 (n = 5, rotated)
	F8	Composition function 8 (n = 5, rotated)

```
function reducer (key, value) {
    combine the message emitted from mapper
    // get the gbest and global optimum
    for each data block {
        if fitness < global best {
            global best = fitness
        }
    }
    emit gbest and global optimum
}
```

Algorithm 3: MRQPSO reduce.

The scalable optimization problems are proposed in the CEC 2013 Special Session on Real-Parameter Optimization [37] and listed in Table 1. All the test composition functions are in the same search range: $[-100, 100]^D$. And they are all minimization problem with global optimum zeros. n is the number of kinds of basic benchmark functions.

Some parameter settings and environment are listed as follows.

We compared our proposed MRQPSO with original QPSO algorithm to test the optimization performance. Each function is run for 20 independent times and all the results are recorded in Tables 2–5. All experiments are run for $2^{13} \times 900$ function evaluations. $D = 10$.

(1) QPSO: this algorithm transforms search space from classic space to quantum space. In quantum space, particles can appear at any position and avoid premature convergence to some degrees. The population size of QPSO is 10.

(2) MRQPSO: this algorithm is a QPSO implementation on the MapReduce model, which achieves the parallel and distributed QPSO. The population sizes of MRQPSO were 10, 20, and 30, respectively, denoted by s on Table 2. The search space is partitioned into 2^{11} blocks averagely.

All experiments are run on VMware Workstation virtual machines version 12.0.0: one processor, 1.0 GB RAM. Hadoop version 1.1.2 and Java 1.7 were used in MapReduce experiments; we used three virtual machines while serial algorithm used one. CPU is core i7. Programming language is Java.

In Table 2, the best, worst, mean value, standard derivation, and running time of MRQPSO with different population sizes are listed. According to the results, as the population size becomes larger, the solutions become worse. It may be because that when the number of particles increases, the number of iterations of each particle decreased for the same function evaluations, which is not helpful to improve accuracy.

5.2. Comparison with QPSO on Test Problems. The results of MRCPSO are compared with QPSO algorithm in Tables 3 and 4. The population size is set to 10. We show two columns for each item to compare two algorithms clearly. From Table 3, MRQPSO has a better solution almost on all items. For F2 and F3, although the best value of QPSO is lower, MRQPSO meet this value nearly. Since these two values are close, one can consider that two algorithms are trapped into the same local optimum. And due to the less number of iterations of each particle, the MRQPSO cannot converge to a lower point like QPSO. In general, the MRQPSO has a better performance on mean value and standard derivation; this suggests the MRQPSO is more capable of searching for the optimum and overcoming the premature phenomenon and is more robust and steady than original QPSO.

The notable advantage in time is presented in Table 4. From this chart, the MRQPSO is more effective in saving time cost. And it seems that the more time QPSO spend, the more advantage MRQPSO has. Normally, it takes some time to start mapper. When a problem is so simple that the serial algorithm processes fast, the outstanding benefit of rapid convergence may weaken, such as F1–F3. But when search time gets longer, the mapper starting time even can be negligible, such as F4–F7, where the MRQPSO programs' running time reduced to half compared to the QPSO.

To summarize, we can discover that the MRQPSO has better solution performance and cost less running time. The proposed MRQPSO is more suitable and effective for dealing with complex problems.

5.3. Comparisons on Nonlinear Equation Systems. Nonlinear equation systems arise in many areas, such as economics [38],

Table 2: Performance of MRQPSO.

Fun	s	Min value	Max value	Mean function value	St. d	Mean running time (ms)
F1	10	3.17E + 01	1.23E + 02	8.10E + 01	3.26E + 01	52098
	20	2.98E + 01	1.21E + 02	9.34E + 01	2.86E + 01	55393
	30	4.16E + 01	1.37E + 02	1.07E + 02	3.11E + 01	63067
F2	10	4.26E + 01	1.36E + 02	9.07E + 01	2.09E + 01	57123
	20	1.14E + 02	2.50E + 02	1.84E + 02	3.58E + 01	59785
	30	1.82E + 02	4.53E + 02	2.98E + 02	6.93E + 01	68330
F3	10	1.78E + 02	4.76E + 02	3.65E + 02	7.50E + 01	58311
	20	1.88E + 02	7.83E + 02	5.54E + 02	1.57E + 02	61245
	30	4.84E + 02	9.22E + 02	7.58E + 02	1.12E + 02	68463
F4	10	1.04E + 02	1.13E + 02	1.09E + 02	1.75E + 00	942736
	20	1.08E + 02	1.14E + 02	1.11E + 02	1.39E + 00	978615
	30	1.07E + 02	1.16E + 02	1.13E + 02	2.46E + 00	1108999
F5	10	1.07E + 02	1.15E + 02	1.12E + 02	2.62E + 00	903781
	20	1.10E + 02	1.18E + 02	1.14E + 02	2.61E + 00	984774
	30	1.10E + 02	1.20E + 02	1.16E + 02	2.89E + 00	1134965
F6	10	1.06E + 02	1.11E + 02	1.08E + 02	1.09E + 00	982763
	20	1.04E + 02	1.11E + 02	1.08E + 02	2.04E + 00	1053224
	30	1.09E + 02	1.16E + 02	1.12E + 02	1.94E + 00	1183599
F7	10	1.54E + 02	3.42E + 02	2.37E + 02	5.08E + 01	970225
	20	1.61E + 02	3.54E + 02	2.49E + 02	5.23E + 01	1056708
	30	1.92E + 02	3.71E + 02	2.91E + 02	4.68E + 01	1239350
F8	10	1.04E + 02	1.23E + 02	1.14E + 02	5.16E + 00	76354
	20	1.08E + 02	1.25E + 02	1.15E + 02	3.93E + 00	81578
	30	1.15E + 02	1.43E + 02	1.28E + 02	8.02E + 00	89552

Table 3: Comparison between MRQPSO and QPSO on the optimum.

Fun	Min value		Max value		Mean function value		St. d	
	MR QPSO	QPSO	MR QPSO	QPSO	MR QPSO	QPSO	MR QPSO	QPSO
F1	**3.17E + 01**	2.13E + 02	**1.23E + 02**	8.32E + 02	**8.10E + 01**	4.45E + 02	**3.26E + 01**	1.51E + 02
F2	4.26E + 01	**3.69E + 01**	**1.36E + 02**	7.96E + 02	**9.07E + 01**	3.15E + 02	**2.09E + 01**	2.22E + 02
F3	1.78E + 02	**9.66E + 01**	**4.76E + 02**	9.95E + 02	**3.65E + 02**	4.97E + 02	**7.50E + 01**	2.60E + 02
F4	**1.04E + 02**	1.28E + 02	**1.13E + 02**	2.26E + 02	**1.09E + 02**	2.14E + 02	**1.75E + 00**	2.07E + 01
F5	**1.07E + 02**	1.10E + 02	**1.15E + 02**	2.26E + 02	**1.12E + 02**	2.11E + 02	**2.62E + 00**	2.42E + 01
F6	**1.06E + 02**	1.06E + 02	**1.11E + 02**	3.27E + 02	**1.08E + 02**	2.40E + 02	**1.09E + 00**	7.87E + 01
F7	**1.54E + 02**	5.06E + 02	**3.42E + 02**	7.07E + 02	**2.37E + 02**	5.80E + 02	**5.08E + 01**	6.09E + 01
F8	**1.04E + 02**	1.42E + 02	**1.23E + 02**	1.07E + 03	**1.14E + 02**	5.81E + 02	**5.16E + 00**	2.60E + 02

Table 4: Comparison between MRQPSO and QPSO on the running time.

Fun	Mean running time (ms)	
	MRQPSO	QPSO
F1	**52098**	60704
F2	**57123**	71371
F3	**58311**	72962
F4	**942736**	1811935
F5	**903781**	1739438
F6	**982763**	1855480
F7	**970225**	1945936
F8	**76354**	104103

engineering [39], chemistry [40], mechanics [41], medicine [42], and robotics [43], widely.

Generally, a nonlinear equation system could be described as [44]

$$\begin{gathered} e_1(\vec{x}) = 0, \\ e_2(\vec{x}) = 0, \\ \vdots \\ e_M(\vec{x}) = 0, \\ \text{where } \vec{x} \in \Omega^D. \end{gathered} \tag{7}$$

Table 5: Comparison between MRQPSO and QPSO on nonlinear equation systems.

		Fun 1		Fun 2		Fun 3	
		QPSO	MRQPSO	QPSO	MRQPSO	QPSO	MRQPSO
Value	Mean	$2.88E-04$	**3.77E − 05**	$1.37E-03$	**0.00E + 00**	**7.47E − 06**	$1.37E-05$
	Max	$4.71E-03$	**1.11E − 04**	$1.52E-02$	**0.00E + 00**	**2.09E − 05**	$3.53E-05$
	Min	**7.22E − 16**	$4.73E-06$	**0.00E + 00**	**0.00E + 00**	**9.10E − 07**	$1.28E-06$
Time	Mean	152362	**94077**	123579	**79761**	115000	**75855**
	Max	160675	**96551**	129658	**86161**	116815	**80671**
	Min	144371	**93284**	119379	**77345**	110133	**73275**

M is the number of equations. D is the dimension of variable. e_i is the ith equation in the system. Usually, at least one equation is nonlinear. If one solution could give all the equations in the system true statement, this solution is an optimal solution of this equation system.

In order to obtain the optimal solutions of a nonlinear equation system, an optimization problem like (8) or (9) could be constructed. Optimal solutions of (8) or (9) are the optimal solutions of nonlinear equation system (7)

$$\begin{aligned} \min \quad & \sum_{i=1}^{M} \left| e_i\left(\vec{x}\right)\right| \\ \text{where} \quad & \vec{x} \in \Omega^D \end{aligned} \tag{8}$$

or

$$\begin{aligned} \min \quad & \sum_{i=1}^{M} e_i^2\left(\vec{x}\right) \\ \text{where} \quad & \vec{x} \in \Omega^D. \end{aligned} \tag{9}$$

In this article, optimization problem like (8) is used to deal with one nonlinear equation system. And MRQPSO are compared with QPSO on Fun 1–Fun 3. The details of these three problems are listed as follows.

Fun 1:

$$\begin{gathered} \sum_{i=1}^{D} x_i^2 - 1 = 0, \\ \left|x_1 - x_2\right| + \sum_{i=3}^{D} x_i^2 = 0, \end{gathered} \tag{10}$$

where D is 20. Feasible area is $[-1,1]^{20}$. Both equations are nonlinear. And 2 theoretical optimal solutions exist.

Fun 2:

$$\begin{gathered} x_1^2 + x_3^2 = 1, \\ x_2^2 + x_4^2 = 1, \\ x_5 x_3^3 + x_6 x_4^3 = 0, \\ x_5 x_1^3 + x_6 x_2^3 = 0, \\ x_5 x_1 x_3^2 + x_6 x_4^2 x_2 = 0, \\ x_5 x_3 x_1^2 + x_6 x_2^2 x_4 = 0, \end{gathered} \tag{11}$$

where D is 6. Feasible area is $[-1,1]^6$. All the six equations are nonlinear. And infinite theoretical optimal solutions exist.

Fun 3:

$$\begin{gathered} \left(x_k + \sum_{i=1}^{D-k-1} x_i x_{i+k}\right) x_D - c_k = 0, \quad 1 \le k \le D-1, \\ \sum_{l=1}^{D-1} x_l + 1 = 0, \end{gathered} \tag{12}$$

where D is 20. Feasible area is $[-1,1]^{20}$. In the system, one equation is linear and other nineteen equations are nonlinear. Infinite theoretical optimal solutions exist.

Some parameters and environment used on solving nonlinear equation system are listed as follows. Each algorithm is performed 20 times on each problem independently. All experiments are run for $2^{10} \times 1000$ function evaluations. The population size is 10. And search space of MRQPSO is partitioned into 2^{10} blocks. Results from QPSO and QPSO are compared and reported in Table 4.

Two aspects are considered in the comparisons. One is running time of the two algorithms. The other is the obtained minimized objective function value. The results are reported in Table 4. The better results are marked with blackbody.

From Table 5, it could be found that both the two algorithms have good performance on objective function value. On Fun 1 and Fun 2, MRQPSO have slight advantage than QPSO on mean and max value. On Fun 3 and min value, QPSO has advantage than QPSO. Here it seems that QPSO obtained much better result on min value of Fun 1. But actually, both of the solutions obtained by MRQPSO and QPSO are very close to the theoretical optimal solutions. But in MRQPSO, the computing resource is assigned on different areas. So during the latter search of MRQPSO, no computing resource as much as QPSO could be used to improve accuracy. This may be the reason for the worse performance of MRQPSO on min value.

But from the view of time cost, it is clear that MRQPSO outperformed QPSO on all the cases. And the advantage is significant. Because three virtual devices are used to evaluate solutions in feasible space at the same time, the computing task could be completed with less time.

6. Conclusion

This paper developed a MRQPSO algorithm and implemented serial QPSO into the MapReduce model, speculative parallelization, and distribution of QPSO. The proposed method was applied to solve the composition benchmark functions and nonlinear equation systems and got satisfactory solutions basically. Moreover, from the comparisons between MRQPSO and QPSO, the results showed us the parallel one outperformed the serial one on both quality of solution and time cost. MRQPSO can be considered as a suitable algorithm to solve large-scale and complex problems. In order to solve more complex practical problems, a cluster with more servers is needed to be constructed and used to test the performance of MRQPSO, which would be a further work.

Consent

Informed consent was obtained from all individual participants included in the study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (nos. 61272279, 61272282, 61371201, and 61203303), the Program for New Century Excellent Talents in University (no. NCET-12-0920), the Program for New Scientific and Technological Star of Shaanxi Province (no. 2014KJXX-45), the National Basic Research Program (973 Program) of China (no. 2013CB329402), the Program for Cheung Kong Scholars and Innovative Research Team in University (no. IRT_15R53), and the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) (no. B07048).

References

[1] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan et al., "Distributed evolutionary algorithms and their models: a survey of the state-of-the-art," *Applied Soft Computing Journal*, vol. 34, pp. 286–300, 2015.

[2] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2016.

[3] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2016.

[4] Z. Fu, F. Huang, X. Sun, A. Vasilakos, and C.-N. Yang, "Enabling Semantic Search based on Conceptual Graphs over Encrypted Outsourced Data," *IEEE Transactions on Services Computing*, no. 99, pp. 1–1, 2016.

[5] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing," *IEICE Transactions on Communications*, vol. E98B, no. 1, pp. 190–200, 2015.

[6] Q. Liu, W. Cai, J. Shen, Z. Fu, X. Liu, and N. Linge, "A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment," *Security and Communication Networks*, vol. 9, no. 17, pp. 4002–4012, 2016.

[7] Y. Kong, M. Zhang, and D. Ye, "A belief propagation-based method for task allocation in open and dynamic cloud environments," *Knowledge-Based Systems*, vol. 115, pp. 123–132, 2017.

[8] Z. Fu, X. Sun, S. Ji, and G. Xie, "Towards efficient content-aware search over encrypted outsourced data in cloud," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, (IEEE INFOCOM '16)*, San Francisco, Calif, USA, April 2016.

[9] F. M. Johar, F. A. Azmin, M. K. Suaidi et al., "A review of genetic algorithms and parallel genetic algorithms on graphics processing unit (GPU)," in *Proceeding of the 2013 IEEE International Conference on Control System, Computing and Engineering, (ICCSCE '13)*, pp. 264–269, December 2013.

[10] Q. Yu, C. Chen, and Z. Pan, "Parallel genetic algorithms on programmable graphics hardware," in *Proceeding of the International Conference on Advances in Natural Computation*, vol. 3612, pp. 1051–1059, Springer-Verlag.

[11] S. Tsutsui and N. Fujimoto, "Solving quadratic assignment problems by genetic algorithms with GPU computation," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '09)*, pp. 2523–2530, Montreal, Canada, July 2009.

[12] P. Pospíchal, "GPU-based acceleration of the genetic algorithm," *Gecco Competition*, 2009.

[13] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[14] S. Aridhi, P. Lacomme, L. Ren, and B. Vincent, "A MapReduce-based approach for shortest path problem in large-scale networks," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 151–165, 2015.

[15] T. G. Addair, D. A. Dodge, W. R. Walter, and S. D. Ruppert, "Large-scale seismic signal analysis with Hadoop," *Computers and Geosciences*, vol. 66, no. 2, pp. 145–154, 2014.

[16] X. Li, J. Song, F. Zhang, X. Ouyang, and S. U. Khan, "MapReduce-based fast fuzzy *c*-means algorithm for large-scale underwater image segmentation," *Future Generation Computer Systems*, vol. 65, pp. 90–101, 2016.

[17] F. Wang, X. Meng, and Y. Zhang, "An adaptive user preferences elicitation scheme for location recommendation," *Chinese Journal of Electronics*, vol. 25, no. 5, pp. 943–949, 2016.

[18] A. Verma, X. Llorà, D. E. Goldberg, and R. H. Campbell, "Scaling genetic algorithms using MapReduce," in *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications (ISDA '09)*, pp. 13–18, Pisa, Italy, December 2009.

[19] C. Jin, C. Vecchiola, and R. Buyya, "MRPGA: an extension of MapReduce for parallelizing genetic algorithms," in *Proceedings of the IEEE 4th International Conference on eScience (eScience '08)*, pp. 214–221, Indianapolis, Ind, USA, December 2008.

[20] X. Llorà, A. Verma, R. H. Campbell, and D. E. Goldberg, "When huge is routine: scaling genetic algorithms and estimation of distribution algorithms via data-intensive computing," *Studies in Computational Intelligence*, vol. 269, pp. 11–41, 2010.

[21] K. Tagawa and T. Ishimizu, "Concurrent differential evolution based on MapReduce," *International Journal of Computers*, vol. 4, no. 4, pp. 161–168, 2010.

[22] C. Zhou, "Fast parallelization of differential evolution algorithm using MapReduce," in *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, (GECCO '10)*, pp. 1113–1114, July 2010.

[23] B. Wu, G. Wu, and M. Yang, "A MapReduce based ant colony optimization approach to combinatorial optimization problems," in *Proceedings of the 8th International Conference on Natural Computation (ICNC '12)*, pp. 728–732, May 2012.

[24] A. W. McNabb, C. K. Monson, and K. D. Seppi, "Parallel PSO using MapReduce," in *Proceedings of the 2007 IEEE Congress on Evolutionary Computation, (CEC '07)*, pp. 7–14, September 2007.

[25] Y. Wang, Y. Li, Z. Chen, and Y. Xue, "Cooperative particle swarm optimization using MapReduce," *Soft Computing*, pp. 1–11, 2016.

[26] Y. Li, Z. Chen, Y. Wang, and L. Jiao, "Quantum-behaved particle swarm optimization using mapreduce," *Bio-Inspired Computing—Theories and Applications*, vol. 682, pp. 173–178, 2016.

[27] J. Li, J. Dang, and Y. Wang, "Medical image segmentation algorithm based on quantum clonal evolution and two-dimensional tsallis entropy," *Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao/Journal of Computer-Aided Design and Computer Graphics*, vol. 26, no. 3, pp. 465–471, 2014.

[28] O. P. Patel and A. Tiwari, "Novel quantum inspired binary neural network algorithm," *Sādhanā*, vol. 41, no. 11, pp. 1299–1309, 2016.

[29] L. Jiao, Y. Li, M. Gong, and X. Zhang, "Quantum-inspired immune clonal algorithm for global optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 5, pp. 1234–1253, 2008.

[30] Y. Li, R. Xiang, L. Jiao, and R. Liu, "An improved cooperative quantum-behaved particle swarm optimization," *Soft Computing*, vol. 16, no. 6, pp. 1061–1069, 2012.

[31] G. Zhang, "Quantum-inspired evolutionary algorithms: a survey and empirical study," *Journal of Heuristics*, vol. 17, no. 3, pp. 303–351, 2011.

[32] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the IEEE 2004 Congress on Evolutionary Computation*, pp. 325–331, 2004.

[33] J. Sun, W. Fang, X. Wu, V. Palade, and W. Xu, "Quantum-behaved particle swarm optimization: analysis of individual particle behavior and parameter selection," *Evolutionary Computation*, vol. 20, no. 3, pp. 349–393, 2012.

[34] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the International Conference on Neural Networks (ICNN '95)*, pp. 1942–1948, Perth, Australia, 1995.

[35] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 69–73, Anchorage, Alaska, USA, May, 1998.

[36] F. van den Bergh, *An analysis of particle swarm optimizers [Ph.D. dissertation]*, Department of Computer Science at the University of Pretoria, Pretoria, South Africa, 2002.

[37] J. J. Liang, B. Y. Qu, P. N. Suganthan, and G. Alfredo, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," Tech. Rep. 201212, Computational Intelligence Laboratory, Singapore; Zhengzhou University, Zhengzhou, China; Nanyang Technological University, January 2013.

[38] C. Hipp and M. Plum, "Optimal investment for insurers," *Insurance: Mathematics and Economics*, vol. 27, no. 2, pp. 215–228, 2000.

[39] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, 1999.

[40] C. Patrascioiu and C. Marinoiu, "The applications of the nonlinear equations systems algorithms for the heat transfer processes," in *Proceedings of the 12th WSEAS International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems (MAMECTIS '10)*, pp. 30–35, 2010.

[41] B. P. Mann and N. D. Sims, "Energy harvesting from the nonlinear oscillations of magnetic levitation," *Journal of Sound and Vibration*, vol. 319, no. 1-2, pp. 515–530, 2009.

[42] N. Filipovic, Z. Teng, M. Radovic, I. Saveljic, D. Fotiadis, and O. Parodi, "Computer simulation of three-dimensional plaque formation and progression in the carotid artery," *Medical & Biological Engineering & Computing*, vol. 51, no. 6, pp. 607–616, 2013.

[43] C. L. Collins, "Forward kinematics of planar parallel manipulators in the Clifford algebra of P2," *Mechanism and Machine Theory*, vol. 37, no. 8, pp. 799–813, 2002.

[44] W. Song, Y. Wang, H.-X. Li, and Z. Cai, "Locating multiple optimal solutions of nonlinear equation systems based on multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 414–431, 2015.

Hindawi
Journal of Optimization
Volume 2017, Article ID 4093973, 14 pages
https://doi.org/10.1155/2017/4093973

Research Article

The Research of Disease Spots Extraction Based on Evolutionary Algorithm

Kangshun Li,[1] Lu Xiong,[1] Dongbo Zhang,[2] Zhengping Liang,[3] and Yu Xue[4]

[1]*College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China*
[2]*School of Civil Engineering and Transportation, South China University of Technology, Guangzhou, China*
[3]*College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China*
[4]*School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China*

Correspondence should be addressed to Lu Xiong; luzi0822@163.com

Received 10 January 2017; Accepted 20 February 2017; Published 3 May 2017

Academic Editor: Maoguo Gong

According to the characteristics of maize disease spot performance in the image, this paper designs two-histogram segmentation method based on evolutionary algorithm, which combined with the analysis of image of maize diseases and insect pests, with full consideration of color and texture characteristic of the lesion of pests and diseases, the chroma and gray image, composed of two tuples to build a two-dimensional histogram, solves the problem of one-dimensional histograms that cannot be clearly divided into target and background bimodal distribution and improved the traditional two-dimensional histogram application in pest damage lesion extraction. The chromosome coding suitable for the characteristics of lesion image is designed based on second segmentation of the genetic algorithm Otsu. Determining initial population with analysis results of lesion image, parallel selection, optimal preservation strategy, and adaptive mutation operator are used to improve the search efficiency. Finally, by setting the fluctuation threshold, we continue to search for the best threshold in the range of fluctuations for implementation of global search and local search.

1. Introduction

In recent years, as the country pays more attention to the problem of "agriculture, countryside, and farmers," "precision agriculture" concept is gradually on the rise. The new ideal of using information technology and computer technology to assist farmers to carry out agricultural production has been put on the agenda. In order to solve the problem of fuzziness and subjectivity in the detection and identification of crop diseases and insect pests, graphics and image processing technology are introduced into the field of agricultural pests identification and control. Image classification and recognition technology of crop pests and diseases is the application of image processing technology in the field of crop diseases and insect pests identification [1]. The image classification and recognition technology will be able to rely on computer image to obtain a large amount of information, as the basis for the diagnosis of diseases and insect pests can make up for the shortcomings of traditional diagnostic techniques, which has the characteristics of less input parameters, fast diagnosis, high accuracy, and good real-time performance, so it has important significance for the timely provision of the necessary information and the prevention and treatment of diseases and insect pests of agricultural workers.

Research on the diagnosis and image recognition of crop diseases and insect pests on machine vision in foreign countries should be carried out earlier [2–4]. As early as 1985, related researchers identified disease spot by the shape characteristic of grain. In recent years, with the development of computing power, especially the development of pattern recognition and image processing, computer vision technology has been developed. Then, computer vision technology has been applied to the identification and detection of pests and diseases of crops. Keagy, Zayas, and Christopher et al. studied the recognition technology of computer vision and applied it to the recognition of stored grain pests. The experimental results show that the research has high recognition efficiency. Gassoumi studied the identification of insects in cotton field and applied the computer image processing technology based on feature extraction to identify the pests. Gassoumi also proposed a fuzzy neural network recognition algorithm; the algorithm can identify 12 kinds of pests in the research content, and the recognition rate is above 90%. Murakami et al. sampled the leaves of cucumber and used a variety of methods such as gray level cooccurrence matrix to identify the pests. Where the thrips pest identified, recognition rate is as high as 90%. Shariff proposes a classification algorithm based on fuzzy logic, which is aimed at the 6 kinds of pests in rice. Experimental results show that the algorithm has good classification and recognition rate.

Relatively speaking, the research on the field of classification and identification of crop diseases and insect pests in China is more advanced than foreign countries. In spite of this, some universities and research institutes in China have accumulated a certain research foundation in this field. This is mainly due to the release of China's "food industry science and technology development plan"; the plan clearly shows that the increase in food production needs to have information technology support. At present, China's agricultural information construction is relatively backward, so there is a need to strengthen the agricultural information technology research and gradually the world's leading edge of Agricultural Technology. However, there are many research directions in the agricultural information construction, in which the intelligent identification and detection of diseases and insect pests is one of the key problems to be solved. Therefore, it has a long-term strategic significance to solve practical problems. In our country, many researchers have studied and solved this problem by using computer vision, and some practical problems have been solved [5, 6]. Among them, Qiu et al. and others used artificial neural network technology [7]. They developed a real-time detection system for 9 kinds of pests in the field of crops. With the rapid development of computer vision technology, the problem of population density calculation of micro insect pests has been solved. The laboratory of IPMIST of the China Agricultural University calculated the population density of the whitefly in the crops based on their research results. The experimental results show that the recognition accuracy can reach more than 90%.

Chen et al. used digital image processing technology and local threshold method to segment the target region and nontarget region, and the research results were applied to the identification of the cotton leaf worm hole area and the edge of the leaves of defect degree and pest level. The experimental results show that the algorithm is effective, and the error range is less than 0.05 [6]. Zhang et al. use neural network to determine the degree of breeding of tomato. The training of neural network is realized by genetic algorithm. Xu et al. are also studying tomato species, but their study is to identify the missing nutrients in the culture process. The feature extraction improved by using the percentage difference histogram after eliminating the influence of leaf shape and leaf environment. Therefore, the precision of extracting the texture characteristics of the missing nutrients is improved, and the recognition rate is as high as 80%. Wang also used the digital image processing and neural network [8], but he combined expert knowledge and information technology, so as to play a long distance identification and detection of crop diseases and pests. Cheng et al. set the color and texture of the leaves as a starting point. It is done through the use of color theory, based on these two features to select the appropriate parameters, and the use of fuzzy K-nearest neighbor classification. The image of rice pests and diseases was collected by Wang and Zhou and others [9], who take borer as object based on neural network to complete the identification of the experimental results, with accuracy of up to 90%.

Liang et al. collected different kinds of insect images [10] to obtain the characteristics of the image. They showed the characteristics obtained based on neural network in mathematical form to join the rough set. The experimental results show that the classification accuracy is high. In 2009, Tian et al., Niu et al. and others propose a method for the identification of plant diseases and insect pests based on the SVM. The method is based on the linear kernel function and radial basis function in the SVM based on the identification method for cucumber disease and insect pests. Experimental results show that the proposed algorithm has better recognition accuracy than other support vector machines in the field of cucumber and show that the algorithm has better classification accuracy for small sample data sets. Based on the experiment results, the method proposed in this study provides a classification accuracy of up to 94.12% in training set and 81.82% in testing set with the SVM classifier [11, 11–14]. The extracted features are input into some classifiers including the SVM (support vector machine) [11, 13, 15]. Classification is a classical data mining method, which have been applied in many fields [1, 16–21]. In 2011, Liu et al. also used support vector machine to identify plant diseases and insect pests. They took the traces that the insects left on the persimmon surface as research object, and then the texture and color features were extracted and used as parameters. Support vector machine uses Sigmoid as the kernel function for classification and recognition. The experimental results show that the algorithm can better identify fruit diseases and insect pests. Qiu et al. studied the classification and recognition based on fuzzy decision classifier, and the research results will be applied to the recognition of stored grain pests. In the form of insect pests, the complexity of the recognition feature, through the classification of the classifier, and the accuracy are of up to 95.2%. The experimental results show that this algorithm is aimed at the effect of pest classification. In addition, Qiu et al. also realized the monitoring of field pests and diseases; the experimental results show that the monitoring technology has high practicality. Chen et al. study wheat pests. In addition to the study of the classification of pests, the image segmentation technology is applied to the study. They take the aphids as the target; the experimental results show that the algorithm is effective, and the recognition accuracy is up to 90.7%.

2. Evolutionary Algorithm

Evolutionary algorithm is an iterative search algorithm proposed by Holland in 1975. Evolutionary algorithm [22, 23] is based on Darwin's theory of biological evolution, which is the nature of biological evolution and the nature of the population to be integrated into the algorithm, so that the algorithm is intelligent and parallel. At the same time, the reasonable design evolution algorithm [24] can effectively avoid the local optimization problem in the process of solving the problem, and the algorithm has strong robustness. Evolutionary algorithm essentially is a kind of probability of random search, but with adaptive, it refers to the "survival of the fittest" evolutionary ideas. The algorithm gives a fitness value of each individual in the population, which represents the survival probability of the individual in the process of biological evolution [25]. The higher the fitness of the individual is, the higher the probability of its entry into the next generation of reproduction is. It is this kind of selection strategy that makes the evolutionary algorithm have self-adaptability in the process of solving practical problems and produce better offspring. Evolutionary algorithm for the problem is not directly solving the problem of the solution space for optimal solution search, but the problem of the solution of gene coding and the establishment of the mapping between problem solving and gene coding. Each potential candidate solution has a corresponding gene encoding. In the process of the algorithm, the solution of the code is searched. After determining the coding method for the potential candidate solution, the initial population is generated. Initial population is the starting point of the search algorithm, which is the first generation of the population multiplication and scientific and reasonable setting of initial population, which helps the algorithm to get the optimal solution of the problem faster and more efficiently. In the process of population evolution, evolutionary algorithm is used as the only index to evaluate the quality of the population. The algorithm is analyzed by the model of the problem. The fitness function is designed to solve the problem, and the fitness value is used to evaluate the probability of survival to the next generation in the evolution process of the population and the survival of the fittest. As well as the process of biological evolution, evolutionary algorithms need to experience the three processes of selection, crossover, and mutation. Selection, crossover, and mutation are referred to as genetic operations of evolutionary algorithms. It is because of the genetic operation, which makes the population of individual reproduction of more diversity, we can produce different solutions, to avoid falling into the local optimal dilemma. Evolutionary algorithm, for the whole process as natural biological evolution generally, continues to produce better offspring and algorithm convergence in the last population. The optimal solution of the individual with the highest fitness value is obtained by genetic decoding.

3. Analysis of Diseases and Pest of Crop

Because of the diversity of plant diseases, many difficulties are brought to the diagnosis of machine vision. Field crop pests and diseases have a wide variety of categories and complex and diverse symptoms, which makes the field of crop pests classification and identification more difficult. The research and analysis of crop diseases and insect pests showed that because the leaf diseases and insect pests have the same occurrence, some symptoms are similar; it is difficult to distinguish. Leaf diseases and insect pests are difficult to distinguish because the part of the occurrence is the same, and some symptoms are similar. In this section, maize disease and insect pest image will be used as the research object, study on leaf plant diseases and insect pests, and analysis the lesion area of image visual features and statistical features. Using image analysis and combined machine vision technology and converting some of the information in the image that is difficult to describe to a target that the computer can recognize and deal with have become an important basis for the identification of leaf disease and disease in the classification and identification of diseases and pests.

3.1. Analysis of Image Gray. The gray images of crop diseases and insect pests include many features of the image. Among them, the gray histogram can be generalized to represent the gray statistical information of the image and can get the description information for a particular type of image. As shown in Figure 1, it takes the maize big spot disease image as the analysis target, carries on histogram the original image to 256 gray level, and uses the gray histogram to carry on the analysis. From the visual angle of analysis, leaf disease has obvious characteristics of the external position through the observation. The gray values of disease spot are significantly higher than those of no symptoms around the lesion area. From the point of view of statistical information, it can be found that the gray value of the image of the maize spot disease showed a single peak distribution, and in the 100 areas of gray value changes significantly, as well as the frequency of rapid decline.

The infection process of the leaf of crops is the pathogen invading the local side and gradually spreads outward to form the green spots which differ in size, color, and shape. The lesion and healthy leaf area vary in the color difference, and this difference appears in gray image performance as gray inhomogeneity. As shown in Figure 2, the histogram showed that the gray value of lesion location broadly distributed in the area more than 100, and the gray value of good leaf location broadly distributed below 100. For a particular type of pest disease images, the gray value of lesion and nondisease area is in different gray range. This is consistent with the observed appearance characteristics of the diseased and nondiseased leaves; it is also consistent with the rule of formation of lesion. It is obvious that gray mean value of pest lesion position in the image is higher than that of the disease-free area from Tables 1 and 2. The gray value of variance in the image spot area is large, the distribution is not uniform, and the gray value of no symptoms area convergence and its distribution is uniform. In the histogram, the lesion of the gray value distribution is more average. In the image, the distribution of gray value is scattered, and the distribution of the health area is concentrated.

3.2. Analysis of HSI Chroma. HSI color space is a model of image color which is described by using three kinds of indexes, the chroma value, the saturation value, and the

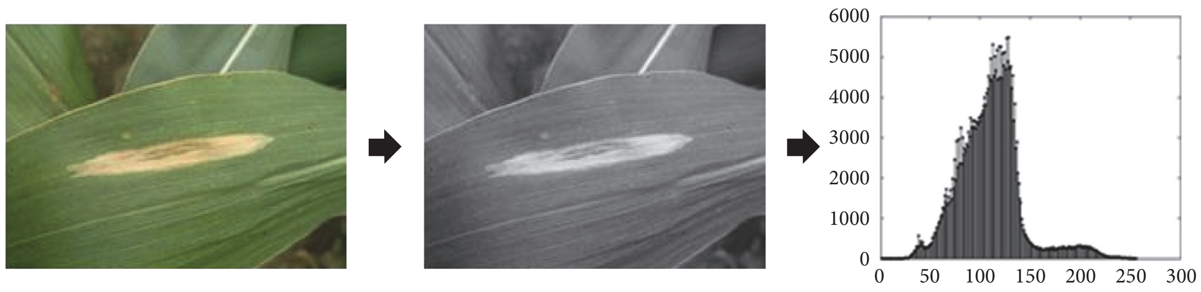


FIGURE 1: Analysis of gray big spot disease image.

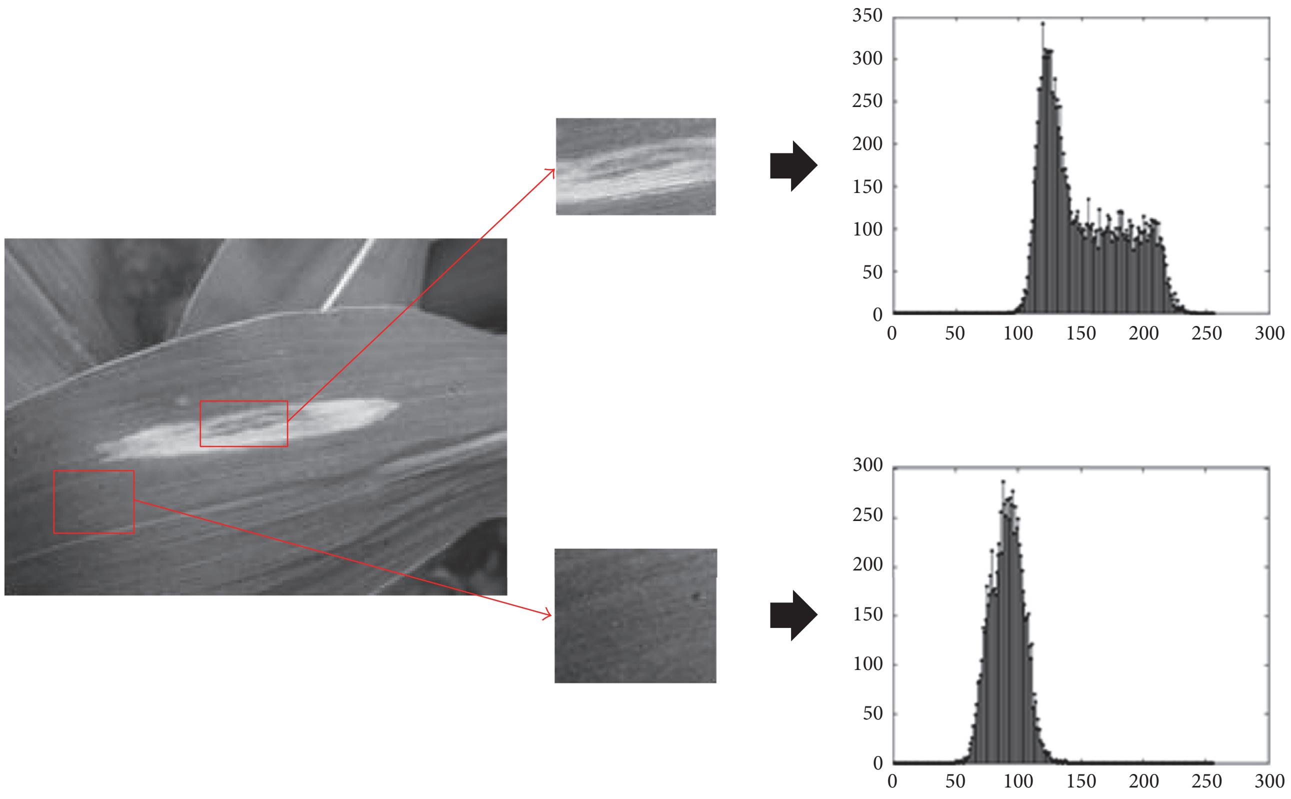


FIGURE 2: Gray histogram distribution characteristics of large spot disease image.

brightness value [26]. The chroma value is defined as the color wavelength, which reflects the color degree of color image. The infection process of the leaf of crops is the pathogen invading the local side and gradually spreads outward to form the green spots which differ in size, color, and shape. Therefore, the color value of HSI color space can be more clearly reflected in the disease and insect pests image of the color difference. General pest spots will appear yellow, red brown, brown, and other strange colors. Therefore, it can be more effective to reflect the visual characteristics and statistical characteristics of the diseased leaves by analyzing the color value of the image of plant diseases and insect pests in the HSI color space. As shown in Figure 3 the big spot disease image of maize is converted to HSI color space from the RGB color space for analysis. In order to facilitate the observation and analysis and to obtain the color value H in the HSI color space, calculate the conversion image I which is divided into 256 levels. The position of disease spot in converting image I has obvious external dissimilarity compared to disease-free position, and the bimodal phenomenon can be observed in the statistical histogram.

The position of big spot disease of conversion image I and leaf position without symptoms were analyzed, as shown in Figure 4. Analysis results show that the stationarity of lesion location chroma values of H and disease-free position without obvious difference and chroma values of H are concentrated in a certain interval, and distribution is uniform. In Tables 3 and 4, it can be found that, in HSI color space chroma, value is an important index to evaluate the object surface color, while the leaf color changed significantly after

Table 1: Gray information of plant diseases and insect pests image.

Image type	Mean value	Mean square error	Entropy	Energy
No symptoms of image	147.67	13.87	5.82	0.019
Big spot disease	109.30	30.27	6.80	0.012
Cochliobolus heterostrophus	125.11	43.41	7.01	0.010
Gray leaf spot	127.28	32.63	6.85	0.011
Cercospora leaf spot	146.01	26.39	6.64	0.012
Anthrax	147.14	32.11	6.90	0.009

Table 2: Gray information of pest disease image.

Types of spot	Mean value	Mean square error	Entropy	Energy
Cochliobolus heterostrophus	183.76	35.94	6.83	0.009
Gray leaf spot	170.04	32.03	6.45	0.013
Cercospora leaf spot	140.95	23.34	6.50	0.014
Anthrax	149.67	19.32	6.24	0.014

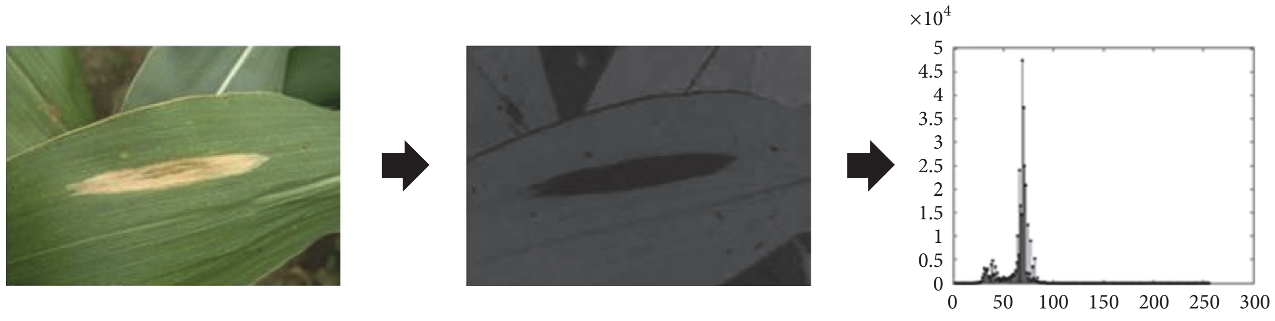


Figure 3: Analysis of HSI chroma of big spot disease.

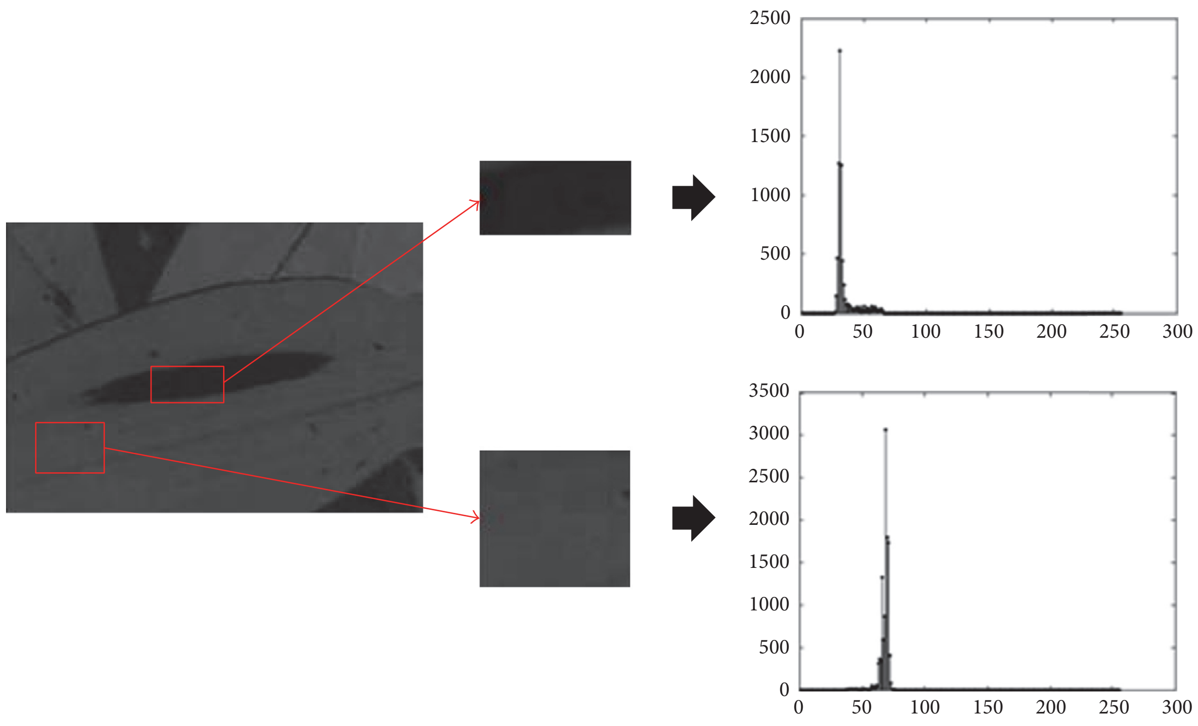


Figure 4: Distribution characteristics of HSI color in the image of the large spot disease.

Table 3: Chroma information of image of plant diseases and insect pests.

Types of spot	Mean value	Mean square error	Entropy	Energy
No symptoms image	67.34	2.76	3.11	0.153
Big spot disease	63.77	11.94	4.58	0.069
Cochliobolus heterostrophus	55.16	10.46	5.01	0.041
Gray leaf spot	75.98	21.85	6.13	0.017
Cercospora leaf spot	58.54	9.22	4.87	0.044
Anthrax	44.74	10.99	5.04	0.038

Table 4: Chroma information of image of plant diseases and insect pests.

Types of spot	Mean value	Mean square error	Entropy	Energy
Big spot disease	32.44	6.90	3.32	0.170
Cochliobolus heterostrophus	43.06	8.84	4.88	0.038
Gray leaf spot	41.39	20.15	5.45	0.034
Cercospora leaf spot	55.91	9.18	4.94	0.040
Anthrax	37.68	5.11	4.33	0.056

the occurrence of the disease. And this change can reflect the use of chroma, as an important basis for judging the location of spot disease.

4. The Segmentation Scheme of Spot Disease Based on Genetic Algorithm

4.1. The Main Steps of Segmentation of Spot Disease Extraction Based on Genetic Algorithm. Digital image processing, image matching, image classification, and recognition technology, are the bases of the work on which the image segmentation is generally based on [27]. Image segmentation can be used to extract meaningful features in the image, which can be used to obtain the required image information. The object of image segmentation is to decompose the modules with the same or similar features in the image and get some basic elements with different characteristics. The basic element which is obtained by image segmentation is called image element. The pest images can be more easily and quickly processed after extracting the image element from pest spot image. Therefore, the realization of image classification and recognition play a fundamental role, according to the characteristics of crop diseases and insect pests image, to design an effective image segmentation algorithm, extraction plant diseases, and insect pests in the lesion area. The image features of plant diseases and insect pests can be divided into visual features and statistical features [28]. The visual features of the image of pests and diseases refer to the characteristics which can be identified by human visual system, such as image color, leaf lesion brightness, texture, and shape. Statistical characteristics of the image of plant diseases and insect pests are obtained by means of a certain image transformation, such as color moment, statistical histogram, and image spectrum.

Threshold segmentation based on image gray value is a very effective method, which can reduce the computational cost and can be used in special real-time applications. Therefore, the threshold method is widely used. In this section, we present a threshold segmentation algorithm based on evolutionary algorithm whose fusion guides filtering algorithm to set the weights matrix to achieve the extraction of diseased spots. The specific process is as shown in Figure 5.

The following are details about the major steps of the processing flow:

(1) Construct two-dimensional histogram, using color and gray information to establish two-dimensional histogram after conversing original pest damage image to gray and color image. Plant diseases and insect pests lesion image design for a target class; the other parts are set as the background which makes the histogram able to reflect more pest image information. From one-dimensional to two-dimensional transformation, it can more clearly reflect the spot information distribution and information clustering.

(2) Get two valued matrix marked spot diseases based on application of genetic algorithm. Threshold search using a genetic algorithm is based on the design of the chroma and gray histogram, and threshold value is determined by the second optimization. Design chromosome coding suitable for the image threshold characteristic of diseases and insect pests combined with the results of the image analysis; the initial population is selected by the empirical value combined with the random value, and the optimization efficiency is improved. Measure function of spot disease segmentation designed to evaluate the merits of individual, according to the measurement method of the target and background in the combination of one-dimensional histogram and two-dimensional histogram. Combined with the characteristics of the

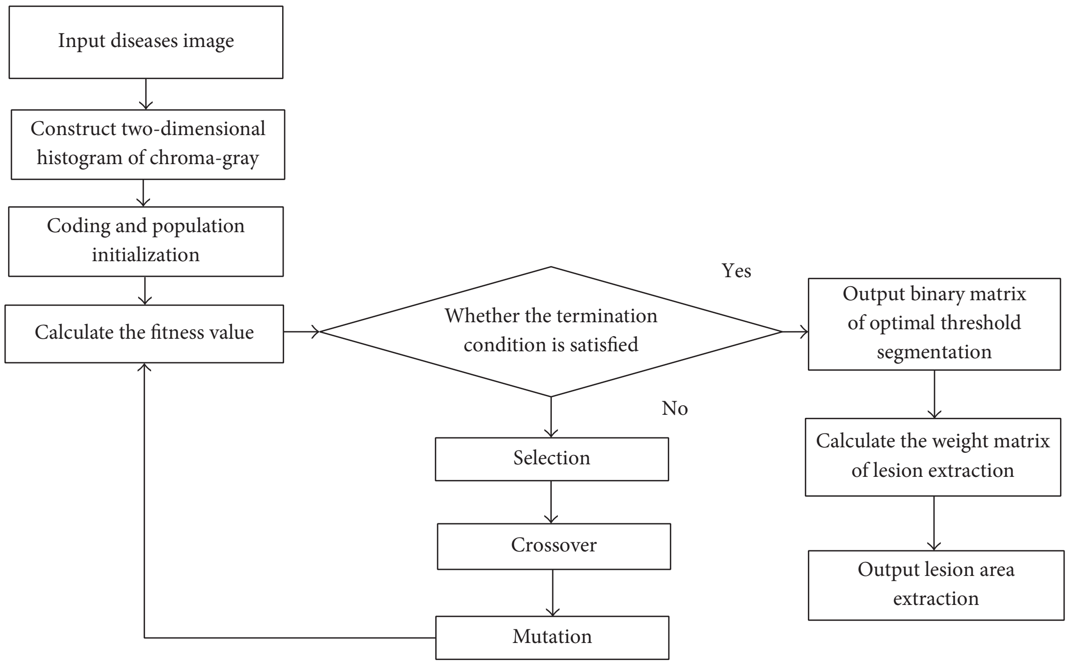

Input diseases image
Construct two-dimensional histogram of chroma-gray
Coding and population initialization
Calculate the fitness value
Whether the termination condition is satisfied
Yes
No
Output binary matrix of optimal threshold segmentation
Calculate the weight matrix of lesion extraction
Output lesion area extraction
Selection
Crossover
Mutation

Figure 5: Flow chart of algorithm implementation of extraction of disease spot.

image of pests and diseases, the strategy of genetic operation is designed to improve the search efficiency and to speed up the convergence of the algorithm. Finally, use genetic algorithm to derive the optimal threshold to obtain binary matrix of spot disease segmentation, labeled pest damage in the image of the spot disease.

(3) The weight matrix of spot extraction is calculated by using directed graph. Simple binary segmentation results are used to extract the disease region; being easy to lose the original lesion edge, shape, texture, and other characteristics, it is not conducive to the classification and identification of plant diseases and insect pests. Therefore, the idea of guiding filtering algorithm is introduced. The filtering operation on the image segmentation is based on the gray orientation guide. Calculate the weight matrix of lesion region extraction and optimize results of lesion extraction. Restore the segmented lesion edge, texture fuzziness, and rough location; it is better to keep pest lesion image edge, shape, texture, and characteristics.

4.2. The Combination of Genetic Algorithm and Chroma-Gray Histogram. Threshold segmentation is an effective method of image segmentation, and the ideal situation of threshold segmentation is the statistical histogram of the image to be bimodal, and the valley in the histogram is used as the segmentation threshold to separate the object and background [29]. But in most cases, the image histogram is not the bimodal form; it is more of a single peak or multipeak shape; this will make the selection of the threshold of the histogram segmentation difficult. Two-dimensional histogram segmentation method is developed from the traditional one-dimensional histogram segmentation, which is based on the threshold value of image segmentation. Two-dimensional histograms, i, j, respectively, represent the image of the two indicators. Take the frequency of two tuples (i, j) as the vertical coordinate to establish 3D information coordinate of image. Set the frequency of two tuples $(i, j) f_{ij}$, respectively, to be corresponding to the gray value of the pixel and the right pixel gray value; the joint probability density is defined as follows:

$$p_{ij} = \frac{f_{ij}}{M \times N}, \quad (1)$$

where $M \times N$ is the size of image. In order to form the gray level cooccurrence matrix, which represents the gray spatial variation, two-dimensional histogram is divided into four regions by using two thresholds (t, s), as shown in Figure 6, where areas 0 and 1 represent background and target, respectively, and region 3 and region 2 represent the information of boundary texture and noise.

In the previous section of pest damage image analysis, we can know lesion location in the image gray value characteristics and that no symptom location gray value feature has an obvious gap, but single gray-scale information cannot well determine the location of the lesion in the image; also it is prone to regional misjudgment of the situation. Therefore, this study proposed a segmentation method based on chroma-gray histogram. At the same time, this method uses the image gray level information and takes full account of the chroma information of image to determine the spot area. Images are converted to HSI chroma space and HSI gray space for extracting gray value and chroma value, respectively. The chroma value is divided into K and the chroma

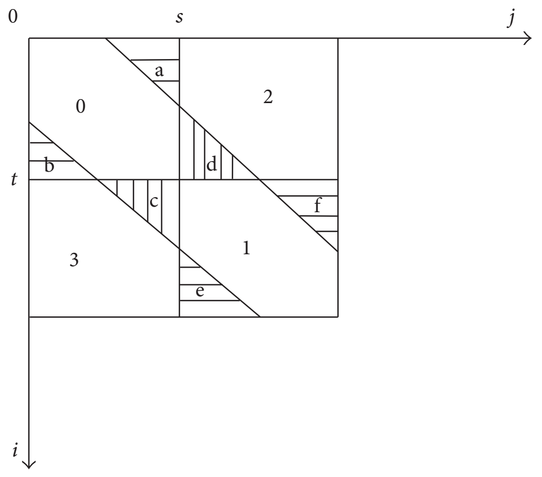


Figure 6: Region division of two-dimensional histogram.

of nongreen area, which is divided in $[t, K-1]$, and the gray value is divided into L. Count the frequency of two tuples (i, j) which are consisting of chroma value and gray value. The distribution probability of region 0 and region 1 is represented by P_0 and P_1 after threshold (t, s) segmentation. Formula (2) and formula (3) are used to represent the distribution probability of the two regions.

$$p_0 = \sum_{i=0}^{t} \sum_{j=0}^{s} p_{ij}, \tag{2}$$

$$P_1 = \sum_{i=t+1}^{K-1} \sum_{j=s+1}^{L-1} P_{ij}. \tag{3}$$

Area 0 is defined as the background region and region 1 is the target area. Region 2 and region 3 are negligible noise points and scattered points, expressed as follows:

$$P_0 + P_1 \approx 1. \tag{4}$$

Thus, in a two-dimensional histogram, the distance measure defined by region 0 and region 1 can be expressed as follows:

$$\delta_B(t, s) = P_0 (\xi_0 - \xi)^2 + P_1 (\xi_1 - \xi)^2, \tag{5}$$

where ξ_0, ξ_1 are mean values of element of regions 0 and 1. ξ is population mean value. Figure out the value of (t^*, s^*) as the threshold of image segmentation when δ_B is the largest value.

As shown in Figure 7, taking the gray leaf spot of maize as an example, the disease and insect pests image is transformed into the two-dimensional histogram of the combination of the image of gray value and chroma value, and the image of gray and chroma value is set to 256 levels, as in $[0, 255]$. Joint probability density of two tuples (i, j) is P_{ij}.

If one uses the two-dimensional histogram of the Otsu method as in formula (5) for segmentation, then the amount of computation is exponential growth. At any t and s, P_0, P_1, ξ_0, and ξ_1 need to execute cumulative calculation, perform 6 times' division, and multiply. The time complexity of arbitrary image processing is $O(N^4)$. In the segmentation of the chroma-gray two-dimensional histogram, in this paper, genetic algorithm is used to find the optimal threshold; combining the goal of one-dimensional and two-dimensional histogram and the background of the measurement method, two-time optimization is performed based on histogram segmentation threshold values t, s. Using genetic algorithm in threshold optimization is faster than the exhaustive search to realize threshold optimization, and the optimal solution of the corresponding image segmentation performance is good and stable.

4.3. Coding Design and Population Initialization

4.3.1. Coding Design. Using the genetic algorithm optimization to segment two-dimensional histogram of crop diseases, the main steps are as follows: in the process of image threshold optimization, according to the characteristics of the diseases image, the genetic algorithm is used to solve the latent solution in the solution space. This key step will also affect the optimization efficiency, convergence speed, and so on. In the analysis of Section 3.2, we can use the statistical analysis of the data obtained by the different diseases image gray and chroma distribution and characteristics to coding. Using binary encoding, the gray and chroma of the image are divided into 256 levels, with 8-binary number coding, that is, from 00000000 to 11111111, corresponding to the diseases image gray and chroma value of the 256 decimal number interval of 0 to 255.

4.3.2. Population Initialization. The initial population is the starting point of the optimal threshold search. According to the gray and chroma features of the pest image, we give the initial population in the gray space and chroma space. In this paper, we adopt the same strategy for gray and chroma image; 5 empirical values with good effect on the experiment and 5 random values randomly generated from 0 to 255 were selected as the initial population.

4.4. The Design of Adaptive Value Function. The genetic algorithm is used to evaluate the fitness of individuals, and based on this, it can produce better offspring. Fitness function is a function of calculating the fitness value of the population, and it is an important function that directly affects the search efficiency and convergence of the algorithm. Therefore, the fitness function that is designed for the two-dimensional histogram threshold segmentation is very important. Fitness function design, usually achieved by solving the problem of modeling and computing the objective function, in the transformation process formed by some specific practical problems of the fitness function, reflects the individual merits.

Set a given image chroma level to be L, segmentation threshold value of chroma to be t, gray level to be L, and segmentation threshold value of gray to be s. For the first search, suppose $s = L - 1$, that is, $P_1 = 0$. The distribution

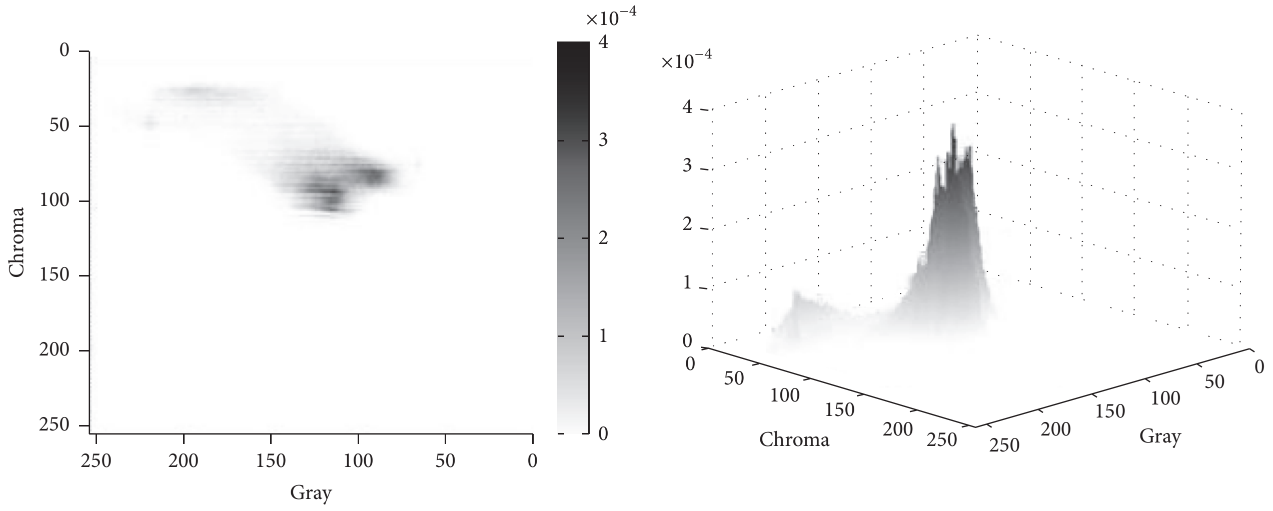


FIGURE 7: Two-dimensional histogram of chroma-gray.

probability of area 3 cannot be ignored, and the distance measure of the first optimization is expressed as follows:

$$\sigma_{B,1}(t) = P_0(\xi_0 - \xi)^2 + P_3(\xi_3 - \xi)^2, \quad (6)$$

where P_3 is the distribution probability of area 3 and ξ_3 is the mean of elements in area 3. Seek t^* as the threshold of the first search when $\sigma_{B,1}$ is the maximum.

On the second search for optimization, put $t = t^*$ into formula (5); then formula (7) is

$$\sigma_{B,2}(t) = \sigma_B(t^*, s) = P_0(\xi_0 - \xi)^2 + P_1(\xi_1 - \xi)^2. \quad (7)$$

Seeking s^* as the threshold of the second search when $\sigma_{B,2}$ is the maximum, the result of the two optimizations (t^*, s^*) is taken as the final threshold.

Therefore, formula (6) and formula (7) are used as the fitness function of the two-segmentation method that this paper adopted, respectively.

4.5. The Design of Genetic Strategy

4.5.1. Parallel Selection. Selection is the first step of genetic manipulation; the purpose is to select the population of individuals for the crossover, mutation, and other operations. Individuals with better fitness values can produce a new generation of populations by selection. In order to guarantee the multiplicity of the population in the process of reproduction and to keep pace with the convergence speed of the algorithm, this paper adopts the method of combining fitness value selection with optimal preservation strategy.

The specific design is as follows: first, the individual selection probability is given to each individual according to the proportion distribution method of population fitness. After the crossover and mutation operation, the individuals of the parent population and the offspring population are sorted according to the fitness value, respectively. The highest individual adopts a conservation strategy, substituting the optimal individuals in the parent population directly for the non-optimal individuals in the offspring population to form new offspring populations. The ensemble selection method can guarantee the diversity of population by adapting the fitness value proportion, and, at the same time, it can save the optimal individuals accelerating the convergence speed and improve the solution precision.

In the algorithm, the evolutionary computation process is divided into three stages: initial, middle, and final stage, and different optimal preservation strategies are adopted for different periods. In the early stage of evolutionary computation, the individuals with the best fitness (the largest fitness) of the previous generation are substituted for the individuals with the fifth-highest fitness value in the generation, so as to avoid the local convergence of the search results and to enlarge the optimization interval of the division threshold. In the middle period of evolutionary computation, the best individuals of the previous generation are substituted for the individuals whose fitness is ranked last, so that the optimal individuals are preserved in the next generation, and the convergence of the algorithm is guaranteed. At the end of the evolutionary computation, the individuals with the first two ranks of the previous fitness values are substituted for the individuals with the second-order of fitness in the population, so that the algorithm converges quickly in the optimal solution neighborhood and improves the efficiency of optimization.

4.5.2. Single Point Crossover. Crossover [30] is the process of generating new individuals. Through the structural exchange of the individuals, the new individuals are different from the fathers, which ensure the diversity of the population in the process of evolution. Through the intersection operation, the searching space of the genetic algorithm is expanded, and the algorithm has powerful searching ability. In this paper, a single point crossover method is adopted, in which the paired individuals selected for crossover operation are assigned a cross bit in the binary coded bit string, and the crossover bits are the structural demarcation points for the exchange and reorganization of the pairing individuals. The latter part of the encoded bit string structure is to exchange the formation of a new population of individuals. Generate random number by random function to determine the intersection point and set the crossover probability to be 0.7.

4.5.3. Adaptive Mutation. Mutation operator [31] is also an important part of the genetic algorithm. It can effectively increase the local search ability of the algorithm by mutation operator to produce an individual different from the parental property; at the same time, it can increase the diversity of individuals and avoid premature phenomenon. The selection of mutation probability is particularly important; the value of pm is too large and is easy to lose outstanding individual; otherwise it is difficult to jump out of local optimum. In this paper, using adaptive mutation operator, pm will change with the individual fitness value; the larger fitness of individuals is with smaller mutation probability, while the smaller fitness of individuals is with a large probability of mutation.

The calculation formula of mutation probability is as follows:

$$p_m = \begin{cases} \dfrac{k_1\left(f_{\max} - f\right)}{f_{\max} - f_{\text{avg}}} & f \geq f_{\text{avg}} \\ k_2 & f < f_{\text{avg}}, \end{cases} \tag{8}$$

where f is the fitness value of the individual to be mutated, $f_{\max}$ is the largest fitness value in the population, f_{avg} is the average fitness value in the population, and k_1, k_2 are the coefficients, representing the change of p_m, $k_1 = 0.2$, and $k_2 = 0.2$.

4.6. Optimization of Extracting Effect Based on Gray-Oriented Graph. In order to make the binary image segmentation have a more delicate edge texture, this paper proposed a new method for calculating the weight of lesion extraction by using the idea of oriented filtering [32, 33] that performs the filtering operation on the binarization matrix obtained by image segmentation and calculates the weight matrix by transforming to the lesion region. Guided filtering is a smoothing operation that preserves the edges of an image under the guidance of a graph. Definition p is the input image to be processed, I is the guided graph, and q is the output image after filtering operation. Guided filtering is assumed to be in a window w_k centered on pixel k, where q is a linear transformation of the guided graph I and is expressed by (9) as follows:

$$q_i = a_k I_i + b_k, \quad \forall i \in w_k, \tag{9}$$

where coefficients a_k and b_k are constants in window w_k.

In this paper, we define a weight matrix H in the process of lesion area extraction and let D be the original diseases image and J is the lesion extraction image; then the region extraction process is expressed as follows:

$$J_i = H_i D_i, \quad H_i \in [0, 1]. \tag{10}$$

Gray image has a fine edge texture; therefore, the gray image is taken as the guide graph I, introducing the idea of directed filtering; the binary matrix of the image segmentation result is defined as A, in a window w_k centered at the pixel k. Convert H to a linear transformation of the graph I, expressed as follows:

$$H_i = a_k I_i + b_k, \quad \forall i \in w_k. \tag{11}$$

The guidance filtering requires coefficients a_k and b_k to minimize the difference between the output image q and the input image p. Similarly, coefficients a_k and b_k are required to minimize the difference between the weight matrix H and the binarization matrix A. In the window w_k, we consider the cost function as follows:

$$E\left(a_k, b_k\right) = \sum_{i \in w_k} \left(\left(a_k I_i + b_k - A_i\right)^2 + \varepsilon {a_k}^2\right), \tag{12}$$

where ε is a regularization matrix to prevent a_k from being too large and keeping the filtering computationally stable. According to the linear regression analysis, the optimal solution expression of a_k, b_k can be expressed as follows:

$$a_k = \frac{(1/w) \sum_{i \in w_k} I_i A_i - \xi_k \overline{A_k}}{\delta_k^2 + \varepsilon}, \tag{13}$$

$$b_k = \overline{A_k} - \xi_k a_k, \tag{14}$$

where δ_k^2 and ξ are the variance and mean of gray-guided graph I in window w_k and $|w|$ is the number of elements in window w_k. At the same time, $\overline{A_k}$ is the mean of all the elements in window w_k. Since element i can be contained in multiple windows and the value of a_k, b_k changes due to the change of the window center k, it is necessary to calculate the mean value of a_k and b_k in the window centered on element i. The leading filter output of the final weight matrix H is the result of the following formula:

$$H_i = \frac{1}{|w|} \sum \sum_{i \in w_k} \left(a_k I_i + b_k\right) = \overline{b_i} I_i + \overline{b_i}, \tag{15}$$

where $\overline{a_i} = (1/|w|) \sum_{k \in w_i} a_k$, $\overline{b_i} = (1/|w|) \sum_{k \in w_i} a = b_k$. In order to reduce the noise point, the impact of scattered points, here set a coefficient θ, the weight matrix into the formula (16).

$$H_i = \begin{cases} 0, & H_i < 0 \\ H_i, & H_i. \end{cases} \tag{16}$$

It can be seen that the weight matrix H has a detail texture and edge effect similar to that of the gray-scale guided graph I while having features similar to the binarization matrix A. And guiding the filter with a fast implementation, only the time complexity, is a good performance and can quickly achieve the filtering algorithm.

5. Experimental Results and Analysis

5.1. Experimental Results. Figure 8 shows the effect of the spot extraction algorithm. The image test is focused on the image of corn diseases at 200 × 200 pixels, which includes the common leaf blight, *Cochliobolus heterostrophus*, and gray

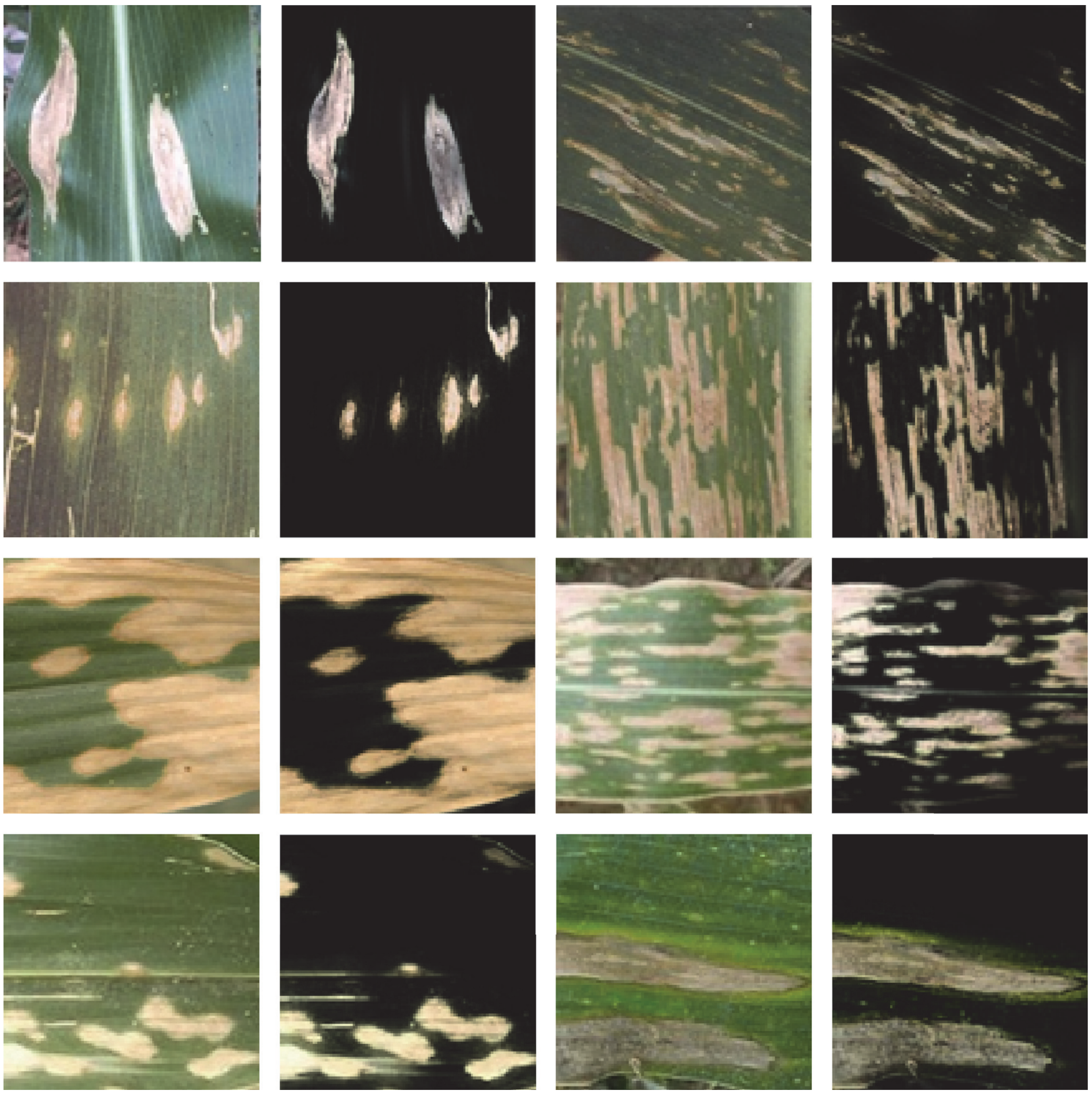


Figure 8: Results of lesion extraction.

spot disease. The experimental results show that the algorithm can effectively segment lesion location, lessen error points, and retain the original lesion images in color, edge, and texture features.

Figure 9 shows the effect of setting different values of on the extraction of lesion area in weight matrix calculation. Experiments show that, the lesion extraction area retains a fine and complete edge and texture but contains a lot of scattered points. With the increase of value, the scattered points are gradually reduced, and the lesion edge texture gradually becomes blurred and rough. Therefore, the value is set to 0.2.

5.2. Comparison and Analysis. Figure 10 is the comparison of the algorithm of this paper with one-dimensional Otsu threshold segmentation method and EM clustering segmentation method. The experimental results show that there are some misjudgment points when the Otsu method and the EM method are used in the segmentation, and the nonlesion location is misdiagnosed as the lesion and the lesion location is misjudged as nonlesion. The algorithm can effectively determine the location of the lesion, the less misjudgment, the region with fine texture edge extraction, and the original image of the lesion area to maintain good consistency.

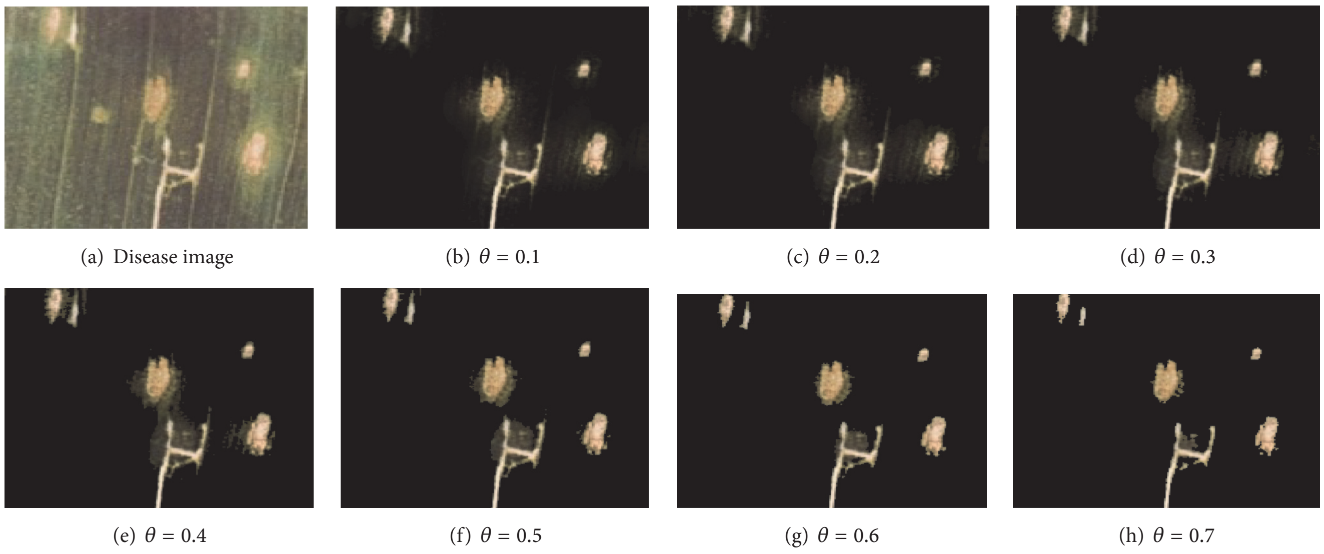

(a) Disease image (b) $\theta = 0.1$ (c) $\theta = 0.2$ (d) $\theta = 0.3$ (e) $\theta = 0.4$ (f) $\theta = 0.5$ (g) $\theta = 0.6$ (h) $\theta = 0.7$

Figure 9: The influence of different θ on the experimental results.

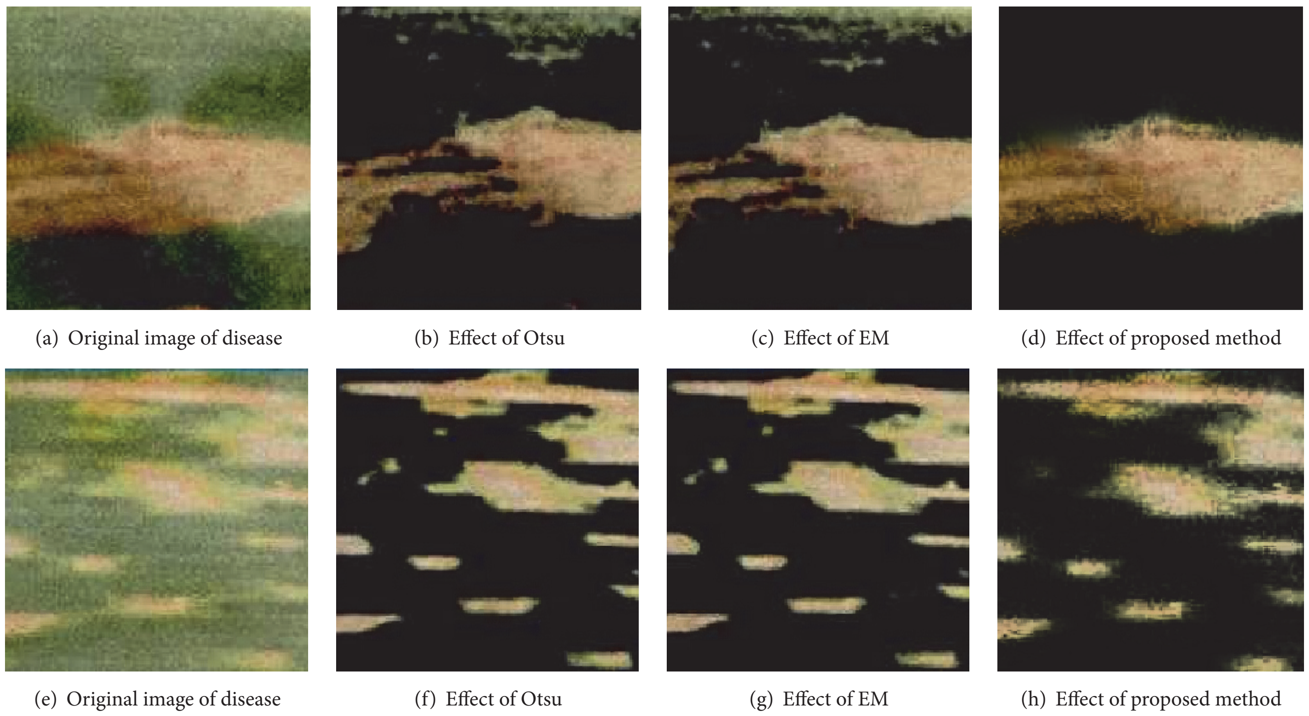
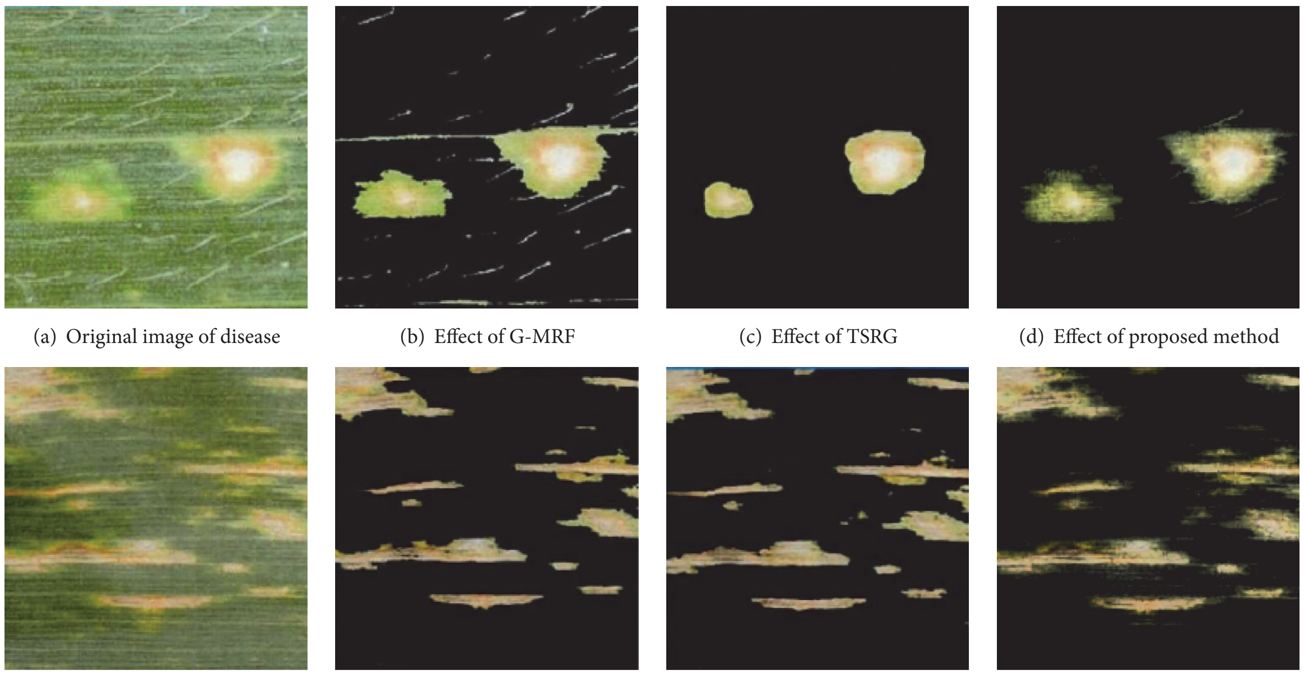

(a) Original image of disease (b) Effect of Otsu (c) Effect of EM (d) Effect of proposed method (e) Original image of disease (f) Effect of Otsu (g) Effect of EM (h) Effect of proposed method

Figure 10: Comparison of the proposed algorithm with Otsu and EM algorithm.

The G-MRF [34] and TSRG algorithm [35] are the lesion segmentation algorithms for corn pests and diseases. Figure 11 shows the experimental comparison results of this algorithm with GMRF and TSRG algorithm. The experimental results show that the G-MRF algorithm can preserve the shape and texture of the lesion to be more intact but is susceptible to other scattered points, resulting in a small number of false points. While the TSRG algorithm can avoid the effect of scattered points of other nonlesion areas, its lesion area cannot retain the original shape of the lesion well. The proposed algorithm can not only preserve the original features of the lesion texture, but also avoid the influence of the scattered points and has high extraction quality.

6. Summary

First of all, this paper analyzes the data of maize diseases and pests damage image, in terms of both gray and chroma space of image in global and local statistical characteristics of the data analysis, taking differences in lesion location



(a) Original image of disease (b) Effect of G-MRF (c) Effect of TSRG (d) Effect of proposed method

(e) Original image of disease (f) Effect of G-MRF (g) Effect of TSRG (h) Effect of proposed method

Figure 11: Comparison result of proposed algorithm with G-MRF and TSRG algorithm.

into account to obtain disease-free position of the color, texture, and so on, as for pest image segmentation and extraction of the lesion. According to the analysis of plant diseases and insect pests image, gray image and the HSI color space of color image are composed of two tuples to build a two-dimensional histogram, to better describe the distribution of pixel, to solve the one-dimensional histograms that cannot be clearly divided into target and background bimodal distribution situation, and to improve the use of traditional two-dimensional histogram in pest damage lesion extraction. For two-dimensional histogram threshold optimization problems, in this paper, the design of threshold optimization process is based on genetic algorithm; design for plant diseases and insect pests image data features of chromosome coding, combined with image analysis, results in the selection of initial population, using parallel selection, optimal preservation strategy, and adaptive mutation operator, to improve the searching efficiency; at the same time by setting the threshold fluctuations, the optimal threshold field fluctuation range continues to search and to achieve combined global search and local search. For binary segmentation of the lesion extraction results, this paper introduces the guidance filtering algorithm, using gray-oriented graph to compute the guiding and the filtering operation on the image segmentation, calculates the weight matrix, in the segmented lesion edge, texture fuzziness and rough position for restoration, to optimize the lesion extraction results and for better retention of plant diseases and insect pests in the edge of the lesion images and texture feature.

Finally, experimental results show that the algorithm of lesion region extraction is effective and discuss the weight matrix parameters setting effect of experimental results. Through comparison with the traditional Otsu algorithm, EM clustering algorithm indicates the applicability of the algorithm of crop disease, insect, and pest image, at the same time, through comparative tests with Gauss MRF and tsrg algorithm; the algorithm in the spot region extraction affects the superiority.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Foundation of China with Grant no. 61573157, the Fund of Natural Science Foundation of Guangdong Province of China with Grant no. 2014A030313454, the Department of Education of Guangdong Province under Grant 2015KQNCX193, and Innovation Foundation for Higher Education of Guangdong, China (Grant no. 2016KTSCX121).

References

[1] J. Wang, W. Zhang, L. Liu, and S. Huang, "Summary of crop diseases and pests image recognition technology," *Computer Engineering & Science*, vol. 36, no. 7, pp. 1363–1370, 2014.

[2] P. M. Keagy and T. F. Schatzki, "Machine recognition of weevil damage in wheat radiographs," in *Optics in Agriculture and Forestry*, vol. 1836 of *Proceedings of SPIE*, pp. 108–119, The International Society for Optical Engineering, 1993.

[3] C. Ridgway, R. Davies, and J. Chambers, "Imaging for the high-speed detection of pest insects and other contaminants in cereal grain in *Transit*," Sacramento, Calif, USA, July 29-August 1 2001.

[4] I. Y. Zayas and P. W. Flinn, "Detection of insects in bulk wheat samples with machine vision," *Transactions of the American Society of Agricultural Engineers*, vol. 41, no. 3, pp. 883–888, 1998.

[5] B. Chen, X. Guo, and X. Li, "Image diagnosis algorithm of diseased wheat," *Transactions of the Chinese Society for Agricultural Machinery*, vol. 40, no. 12, pp. 190–195, 2009.

[6] X. Zou, "Research status of crop pest recognition over computer vision," *Computer Systems and Applications*, vol. 20, no. 6, pp. 238–242, 2011.

[7] D. Qiu, H. Zhang, X. Liu, and Y. Liu, "Design of detection system for agriculture field pests based on machine vision," *Transactions of the Chinese Society for Agricultural Machinery*, vol. 38, no. 1, pp. 120–122, 2007.

[8] K. Wang, "Diagnosis of crop disease, insect pest and weed based on image recognition," *Chinese Academy of Agricultural Science*, 2005.

[9] J. Wang and G. Zhou, "The research of yellow stem borer identification system based on neural network," *Agriculture Network Information*, vol. 2, pp. 39–41, 2006.

[10] Z. Liang, F. Liu, Q. Zhao, and R. Du, "Application of mathematical morphology to superfamily level in insect taxonomy," *Zoological Systematics*, vol. 32, no. 1, pp. 142–152, 2007.

[11] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 7, pp. 1403–1416, 2015.

[12] B. Gu and V. S. Sheng, "A robust regularization path algorithm for ν-support vector classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 1, pp. 1–8, 2016.

[13] Z. Xia, X. Wang, X. Sun, Q. Liu, and N. Xiong, "Steganalysis of LSB matching using differences between nonadjacent pixels," *Multimedia Tools and Applications*, vol. 75, no. 4, pp. 1947–1962, 2016.

[14] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, 2016.

[15] Z. Xia, X. Wang, X. Sun, and B. Wang, "Steganalysis of least significant bit matching using multi-order differences," *Security and Communication Networks*, vol. 7, no. 8, pp. 1283–1291, 2014.

[16] H. Chang, M. Yang, and J. Yang, "Learning a structure adaptive dictionary for sparse representation based classification," *Neurocomputing*, vol. 190, pp. 124–131, 2016.

[17] X. Qi, G. Zhao, L. Shen, Q. Li, and M. Pietikäinen, "LOAD: local orientation adaptive descriptor for texture and material classification," *Neurocomputing*, vol. 184, pp. 28–35, 2016.

[18] X. Song, J. Zhang, Y. Han, and J. Jiang, "Semi-supervised feature selection via hierarchical regression for web image classification," *Multimedia Systems*, vol. 22, no. 1, pp. 41–49, 2016.

[19] Z. Liang, J. Sun, Q. Lin, Z. Du, J. Chen, and Z. Ming, "A novel multiple rule sets data classification algorithm based on ant colony algorithm," *Applied Soft Computing Journal*, vol. 38, pp. 1000–1011, 2016.

[20] Y. Lu, Z. Lai, Z. Fan, J. Cui, and Q. Zhu, "Manifold discriminant regression learning for image classification," *Neurocomputing*, vol. 166, pp. 475–486, 2015.

[21] R. Yang and Z. Wang, "Cross-oriented choquet integrals and their applications on data classification," *Journal of Intelligent and Fuzzy Systems*, vol. 28, no. 1, pp. 205–216, 2015.

[22] J. Zhang, Z.-H. Zhang, Y. Lin et al., "Evolutionary computation meets machine learning: a survey," *IEEE Computational Intelligence Magazine*, vol. 6, no. 4, pp. 68–75, 2011.

[23] M. J. Kavetha, "Coevolution evolutionary algorithm: a survey," *International Journal of Advanced Research in Computer Science*, vol. 4, no. 4, 2013.

[24] X. Jiang, J. Ma, and C. Lei, "Kernel Evolutionary Algorithm for Clustering," in *Bio-inspired Computing—Theories and Applications. BIC-TA*, M. Gong, L. Pan, T. Song, and G. Zhang, Eds., vol. 682 of *Communications in Computer and Information Science*, Springer, Singapore, Singapore, 2016.

[25] C. Huo, R. Zeng, Y. Wang, and M. Shang, "A Multi-Parent Crossover Based Genetic Algorithm for Bi-Objective Unconstrained Binary Quadratic Programming Problem," in *Bio-inspired Computing—Theories and Applications. BIC-TA*, M. Gong, L. Pan, T. Song, and G. Zhang, Eds., vol. 682 of *Communications in Computer and Information Science*, Springer, Singapore, Singapore, 2016.

[26] J. Yang and C. Liu, "Research on color space and its transformation in digital image processing," *Journal of Shangqiu Vocational and Technical College*, vol. 8, no. 2, pp. 25–27, 2009.

[27] C. Lei, J. Ma, and X. Jiang, "Unsupervised Image Segmentation Based on Watershed and Kernel Evolutionary Clustering Algorithm," in *Bio-inspired Computing—Theories and Applications. BIC-TA*, M. Gong, L. Pan, T. Song, and G. Zhang, Eds., vol. 682 of *Communications in Computer and Information Science*, Springer, Singapore, Singapore, 2016.

[28] Y. Wang, *Research on Key Technology of The Image Segemention*, Jiangnan University, Beijing, China, 2008.

[29] Y. Liang, R. Pang, and Y. Zhu, "Two dimensional Otsu line segmentation method for gray level images," *Computer Engineering and Applications*, vol. 48, no. 33, pp. 178–182, 2012.

[30] Z. Chen, Y. Jiang, and R. Wang, "Ant Colony Optimization with Different Crossover Schemes for Continuous Optimization," in *Bio-Inspired Computing—Theories and Applications*, M. Gong, L. Pan, T. Song, K. Tang, and X. Zhang, Eds., vol. 562 of *Communications in Computer and Information Science*, Springer, Berlin, Germany, 2015.

[31] K. Li and L. Xiong, "Community detection based on an improved genetic algorithm," *Communications in Computer and Information Science*, vol. 575, pp. 32–39, 2016.

[32] Z. Li, J. Zheng, Z. Zhu, W. Yao, and S. Wu, "Weighted guided image filtering," *IEEE Transactions on Image Processing*, vol. 24, no. 1, pp. 120–129, 2015.

[33] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 35, no. 6, pp. 1397–1409, 2013.

[34] S. Li, *Research on Maize Diseases Intelligent Diagnosis Based on Disease Image*, Shihezi University, Beijing, China, 2010.

[35] J. Li, *Research and Implementation of Intelligent Image Processing Algorithm on Maize Leaf Ill Spots*, Beijing University of Posts and Telecommunications, Beijing, China, 2010.

Hindawi
Journal of Optimization
Volume 2017, Article ID 3259140, 14 pages
https://doi.org/10.1155/2017/3259140

Research Article

A Multiple Core Execution for Multiobjective Binary Particle Swarm Optimization Feature Selection Method with the Kernel P System Framework

Naeimeh Elkhani and Ravie Chandren Muniyandi

Center for Software Technology and Management, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

Correspondence should be addressed to Ravie Chandren Muniyandi; ravie@ukm.edu.my

Received 22 October 2016; Revised 26 January 2017; Accepted 12 March 2017; Published 19 April 2017

Academic Editor: Gexiang Zhang

Membrane computing is a theoretical model of computation inspired by the structure and functioning of cells. Membrane computing models naturally have parallel structure, and this fact is generally for all variants of membrane computing like kernel P system. Most of the simulations of membrane computing have been done in a serial way on a machine with a central processing unit (CPU). This has neglected the advantage of parallelism in membrane computing. This paper uses multiple cores processing tools in MATLAB as a parallel tool to implement proposed feature selection method based on kernel P system-multiobjective binary particle swarm optimization to identify marker genes for cancer classification. Through this implementation, the proposed feature selection model will involve all the features of a P system including communication rule, division rule, parallelism, and nondeterminism.

1. Introduction

The concept of membrane computing (MC) pursues the basics processes taking place in living cell based on its compartmental structure [1]. In general, the so-called P system represents the MC structure in mimicking reactions and rules. This system mimics the basics of cell-like variants such as cell division, cell death, the transformation of objects via rules, and halt process just when there are no more applicable rules. The nature of MC and its tools as P systems are conducted based on three important features including communication rule, parallelism, and nondeterminism. According to the nature of P system, it is quite suitable to represent biological systems, and all molecular interactions can take place in different locations of living cells [2]. Various variants of P system are considered to solve a wide range of problems. Some of the recent studies are [3] which has used the enzymatic numerical P system (EN P system) as the variant of P system mix with active membrane to solve subset sum problem, [4] which is the application of enzymatic numerical P systems discussed in the area of mobile robots control, [5] that used P system as framework to give a better understanding of the pattern in regenerative biology, [6] which applies membrane computing in google earth application, [7] which applies membrane computing in obtaining optimal values of the thresholds, and [8] which introduces various application of membrane computing in real life.

Recently, new models of P systems have been explored. A kernel P system (KP system) based on the tissue P system (graph-based) has been defined, which consists of a low-level specification language that uses established features of existing P system variants and also includes some new elements. KP system, first because of its coherent set of rules, second flexibility of updating rules in different part of modelling, and third its tissue based model which is more adaptable with particle modelling of optimization algorithms, is selected among other variants of P systems to model the multiple-objective binary particle swarm optimization (MObPSO) model in parallel execution. Importantly, KP systems offer a coherent way of integrating these elements into the same formalism [9]. Artificial Intelligence (AI) refers to a machine or algorithm which tries to mimic a cognitive function that human uses for learning or problem-solving. Inspired by AI algorithms—the intelligence that a machine or program demonstrates to address a problem—for feature selection of cancer microarray data, here, we propose a membrane-inspired feature selection method to use the potentials of membrane computing, such as decentralization, nondeterminism, and maximal parallel computing, to address the limitations of AI-feature selection.

Particle swarm optimization has been used to develop feature selection methods in microarray gene expression studies [10–12]. Graph-based MObPSO [13] was modelled through kernel P system for the following reasons: (1) it has ability to model genes (nodes) and define relationships between them (edges); (2) it has a higher accuracy as compared with flat (filter and wrapper) methods, sequential backward elimination (SBE), correlation-based feature selection (CFS), minimum redundancy maximum relevance (mRMR), and sequential forward search (SFS).

Based on the interaction on membrane computing and evolutionary computation, the field of membrane-inspired evolutionary algorithms (MIEAs) has been introduced in the study [14]. After that, more studies have been done to mix the evolutionary algorithms and P systems (e.g., [2, 15, 16]). Some other studies also focused on hybridization of P system with optimization algorithms: for example, [17] combined P system with particle swarm optimization for the aim of minimizing nonlinear optimization problem, [18] combined P system with particle swarm optimization for the objective of enhancing accuracy of particle swarm optimization as well as overcoming the premature convergence, and [19] proposed a particle swarm optimization P system, the so-called PSOPS, and examined the model on seven-bench function optimization problem and concluded that effectiveness of method improved compared to PSO. Moreover, there are other combinations of PSO with membrane computing, for example, [2, 20–22].

These reviews, of previous studies, prove the usefulness of introducing the P systems into EAs and particularly optimization algorithms to improve pure EAs and optimization algorithms. To the best of our knowledge, there is not any work focusing on the use of a kernel P system into optimization algorithms using all characteristics of a membrane system including parallel implementation and particularly in the scope of cancer dataset's feature selection. In this paper, a hybrid model of a kernel P system with multiobjective particle swarm optimization is proposed to improve the performance measures to compare with pure multiobjective particle swarm optimization. The main contribution of this paper is to parallelize the implementation of the proposed KP-MObPSO model that will lead to building a membrane-inspired optimization model taking full advantage of membrane computing.

This article can be summarized as follows. (1) in Section 2, a brief of basic concepts is brought regarding KP system and MObPSO approaches that are used to build the proposed model. (2) In Section 3, first an overview of the proposed KP-MObPSO is explained and then the four steps of building KP-MObPSO are displayed in detail, namely, initialization, evolution, interaction with client, and collecting the final result. The last two steps define how the proposed KP-MObPSO model is parallelized via multicore. (3) In Section 4, technical explanations of executing KP-MObPSO on multicore are brought to provide a clear overview of parallelizing particles for swarm optimization. (4) Section 5 displays the type of cancer dataset, the methodology of the preprocessing dataset, and simulation of a proposed model via real dataset. (5) In Section 6, the performance of proposed KP-MObPSO model compared with pure MObPSO regarding measures such as accuracy and ROC. Finally, a conclusion and suggestion for future work are given in Section 7.

2. Preliminary Approaches

The proposed models in this study are based on the preliminary concepts and algorithms in multiobjective binary particle swarm optimization and kernel P system. MObPSO is already updated and improved through the rules of the kernel P system, the so-called KP-MObPSO.

2.1. The Kernel P System. According to [9, 23], A KP system of degree n is a tuple, $k\Pi = (O, \mu, C_1, \ldots, C_n, i_0)$, where O is a finite set of objects, called an alphabet; μ defines the membrane structure, which is a graph, (V, E), where V represents vertices indicating compartments and belongs to a set of labels $L(l_i, \ldots)$, and E represents edges; $C_i = (t_i, w_i)$, $1 \leq i \leq n$, is a compartment of the system consisting of a compartment type from T and an initial multiset, w_i, over O; i_0 is the output compartment, where the result is obtained (this will not be used in this study). Each rule r may have a guard g, in which case r is applicable when g is evaluated to true. Its generic form is $r\{g\}$. KP systems use a graph-like structure (similar to that of tissue P systems) and two types of rules:

(1) Rules to process objects: these rules are used to transform objects or to move objects inside compartments or between compartments. These rules are called rewriting, communication, and input-output rules.

   (a) Rewriting and communication rule: $x \rightarrow y\{g\}$, where $x \in A^+$, $y \in A^*$, $g \in$ finite regular expressions FE over $(A \cup \overline{A})$; y at the right side is defined as $y = (a_1, t_1) \cdots (a_h, t_h)$, where $a_j \in A$ and $t_j \in L$, $1 \leq j \leq h$, a_j is an object, and t_j is a target, respectively.

   (b) The input-output rule: $(x/y)\{g\}$, where $x, y \in A^*$, $g \in$ finite regular expressions FE over $(A \cup \overline{A})$; it means that x can be sent from current compartment to the environment or y can be brought from the environment to the target compartment.

(2) System structure rules: these rules make a fundamental change in the topology of the membranes, for

example, with division rule on a compartment, dissolution rule on a specific compartment, and making a link between compartments or dissolving the link between them. These rules are described as follows:

(c1) Division rule: $[]_{l_i} \rightarrow []_{l_{i_1}} \cdots [\,]_{l_{i_h}} \{g\}$, where $g \in$ finite regular expressions FE over $(A \cup \overline{A})$; it means compartment l_i can be replaced with h number of compartments. All newly created compartments inherit objects and links of l_i.

(c2) Dissolution rule: $[]_{l_i} \rightarrow \lambda\{g\}$; it means compartment l_i does not exist anymore as well as all its links with other compartments.

(c3) Link-creation rule: $[]_{l_i}; []_{l_j} \rightarrow []_{l_i} - []_{l_j}\{cg\}$; it means a link will be created between compartment l_i and compartment l_j. If there is more than one compartment with the label l_j, one of them will have a connection with l_j nondeterministically.

(c4) Link-destruction rule: $[]_{l_i} - []_{l_j} \rightarrow []_{l_i}[]_{l_j}\{cg\}$; it means the existing link between l_i and l_j will be eliminated and there will not be any link between them anymore. The same as link creation, if there are more than one compartment which has a link with l_i then one of them will be selected nondeterministically to apply this rule.

2.2. The MObPSO Approaches. Optimization problems with multiple goals or objectives are referred to as multiobjective optimization (MOO) problems. Therefore, the objectives may estimate different aspects of solutions, which are partially or wholly in conflict. MOO can be defined as follows: optimize $Z = (f_1(x), f_2(x), \ldots, f_m(x))$, where $x = (x_1, x_2, \ldots, x_m) \in X$. A multiobjective searching concept is clearly described in [24].

Graph structure is one of the models of feature selection for classification. For example, a graph-based MObPSO algorithm [13] proposed to optimize average of node weights and edge weights at the same time through making different subgraphs. This algorithm is a feature selection model to highlight relevant and nonredundant genes in microarray datasets. The results of microarray datasets indicated that graph-based MObPSO produces better performance in comparison to SBE, CFS, mRMR, and SFS methods from the classification accuracy point of view. Multiobjective optimization has been vastly used in evolutionary algorithms [25]. Although execution time in such a technique, which is known as optimization technique, is not efficient, its time complexity is not much higher relative to other comparable methods [13]. MObPSO is designed for maximizing the dissimilarity (negative correlation) and signal-to-noise ratio (SNR); (1) and (2) are represented as edge weight and node weight, respectively. The population is initialized by arbitrarily selected features from the data matrix, and population-fitness values are calculated using dissimilarity and SNR average values. The archive, A, is initialized to the population value after nondominated sorting of the primary population. Velocity and position are updated using (3) and (4). The local *best P* is updated after comparing the current and previous fitness values of a particle, and the global *best G* is updated according to randomly picking a particle from the archive.

$$\text{Dissimilarity} = \left(1 - \frac{\operatorname{cov}(x, y)}{\sqrt{\operatorname{var}(x)\operatorname{var}(y)}}\right) \tag{1}$$

$$\text{SNR} = \left|\frac{\text{mean}(C1) - \text{mean}(C2)}{\text{s.d}(C1) - \text{s.d}(C2)}\right| \tag{2}$$

$$\begin{aligned} V(t+1) = {} & w * v(t) + c1 * r1 \\ & * (pbest(t) - x(t)) + c2 * r2 \\ & * (gbest(t) - x(t)), \end{aligned} \tag{3}$$

$$x(t+1) = x(t) + v(t+1). \tag{4}$$

3. Proposed KP-MObPSO Model and Multiple Core KP-MObPSO

The entire process of the proposed KP-MObPSO model is summarized in Algorithm 1. It consists of four main phases, including (i) initialization, (ii) evolution, (iii) selecting the minimum *gBestScore*, and (iv) collecting the marker gene. Each phase is built based on defined objects and rules.

To implement the exact proposed sequential KP-MObPSO feature selection method on multiple cores, we have defined four stages: initialization, evolution, interaction with client, and collecting the final result. Based on the assumption for the entire model, there are 25 number of particles ($p = 25$) divided into 4 compartments (compartment 1, compartment 2, compartment 3, and compartment 4) having 6 particles in compartments 1 to 3 and 7 particles in compartment 4. The number of compartments is chosen based on the number of logical cores in the machine used for execution of the model. A total number of iterations is 100 times in evolutionary step.

Objects are defined as P: number of particles, Max_c: maximum number of genes inside particles, $\text{position}_i^1 \cdots \text{position}^n$: n number of positions inside each particle, reserve, a: $a_1 \cdots a_{100}$: data source of all genes, Max_c: maximum number of genes inside particles, NGENES, NewNGENES, Q, c, C, sum_diss, sum_snr, FIT, Q: selected gene IDs, *pBestScore*. Rules are defined based on Table 1.

The symbol dictionary and explanation of rules in Table 1 are explained in the following before going through the steps. According to the MObPSO algorithm, random numbers will be generated first to choose a random number of genes for further process. Thus, in Table 1, $r1$, $r2$, and $r3$ are used to implement the first part of MObPSO via kp rules. In $r1$, we assume there is a so-called membrane position, inside the skin membrane which is always represented by the symbol $[\,]_0$. Inside membrane position, we have defined max_c and p which are two objects to present the maximum number of genes and a maximum number of particles, respectively. Through $r1$, p number of membranes will be generated inside the membrane so-called position. Through $r2$, multiple sets of

```
Begin
  (i) Initialize
      Run $r1 > r2 > r3 > r4 > r5 > r6 > r7$ once to initialize $pBestScore$
$it = 1$
  (ii) Evolution
      (a) Run the rules $r1 > r2 > r3 > r4 > r5 > r6 > r7 > r8 > r9 > r10 > r11 > r12 > r14 > r15 > r16$
          by priority to get fitness value and replace with $pBestScore$ in the case $fitness < pBestScore$
      (b) $Converge\ curve = \min(fitness)$
      (c) $w = pBestScore/Converge\ curve$
      (d) if $it \geq 3$ and Converge curve is repeated the same amount at least three times,
          update the binary position of genes from 0 to 1 and from 1 to 0
till $it = 100$
  (iii) $gBestScore = \min(Converge\ curve)$
  (iv) collect marker genes
```

Algorithm 1: The entire process of the proposed KP-MObPSO model.

Table 1: Rules of KP-MObPSO.

$r1$: rewriting

$r1 \equiv [[p, \mathrm{max\_}c]_{\mathrm{position}}]_0 \xrightarrow{\mathrm{pos}} [[(\mathrm{position}^1 \cdots \mathrm{position}^n)_1 \cdots (\mathrm{position}^1 \cdots \mathrm{position}^n)_p]_{\mathrm{position}}]_0$

$[[]_1[]_{\mathrm{position}}]_0$

$r2$: communication

$r2 \equiv [[(\mathrm{position}^1 \cdots \mathrm{position}^n)_1 \cdots (\mathrm{position}^1 \cdots \mathrm{position}^n)_p]_{\mathrm{position}}]_0 \rightarrow [[(\mathrm{position}^1 \cdots \mathrm{position}^n)_1 \cdots (\mathrm{position}^1 \cdots \mathrm{position}^n)_p]_1]_0$

$r3$: communication

$r3 \equiv [(\mathrm{position}^1 \cdots \mathrm{position}^n)_1 \cdots (\mathrm{position}^1 \cdots \mathrm{position}^n)_p]_1 \rightarrow [[(\mathrm{position}^1 \cdots \mathrm{position}^n)_1]_{p_1} \cdots [(\mathrm{position}^1 \cdots \mathrm{position}^n)_p]_{p_n}]_1$

Rules inside each p: $[\ ]_{P1} \cdots [\ ]_{Pn}$: $r4 > r5 > r6 > r7$

$r4$: rewriting

$r4 \equiv [\mathrm{position}, a, \mathrm{max\_}c, p]_{p1}^{pn} \xrightarrow{\text{subgraph 1}} [\mathrm{NGENES}]_{p1}^{pn}$, $[\mathrm{NGENES}]_{p1}^{pn} \xrightarrow{\text{subgraph 1}} [\mathrm{NewNGENES}, Q, c]_{p1}^{pn}$

$r5$: communication/rewriting

$r5 \equiv [\mathrm{NewNGENES}, c, p, a]_{p1}^{pn} \xrightarrow{\mathrm{MyCost}} [C]_{p1}^{pn}$, $[C]_{p1}^{pn} \xrightarrow{\mathrm{MyCost}} [\mathrm{sum\_diss}]_{p1}^{pn}$, $[a]_{p1}^{pn} \xrightarrow{\mathrm{MyCost}} [\mathrm{snr}]_{p1}^{pn}$

$[\mathrm{snr}]_{p1}^{pn} \xrightarrow{\mathrm{MyCost}} [\mathrm{sum\_snr}]_{p1}^{pn}$, $[\mathrm{sum\_diss}, \mathrm{sum\_snr}]_{p1}^{pn} \xrightarrow{\mathrm{MyCost}} [\mathrm{FIT}]_{p1}^{pn}$

$r6$: Link creation

$r6 \equiv [\ ]_{p1}^{pn}$----$[\ ]_{\mathrm{master}}$

$r7$: communication/rewriting

$r7 \equiv [\mathrm{FIT}^n]_{p1}^{pn} \rightarrow [pBestScore^n]_{\mathrm{master}}$, $[Q^n]_{p1}^{pn} \rightarrow [Q^n]_{\mathrm{master}}$

$r8$: division

$r8 \equiv [[[\ ]_{P1} \cdots [\ ]_{Pn}[pBestScore^n, Q^n]_{\mathrm{master}}]_1]_0 \rightarrow [[[\ ]_{P1} \cdots [\ ]_{Pn}[pBestScore^n, Q^n, gBestScore]_{\mathrm{master}}]_{11}[[\ ]_{P1} \cdots [\ ]_{Pn}[\mathrm{fitness}, pBest, gBest,$ $\mathrm{Velocity}, c1, c2, w, V\max, s]_{\mathrm{master}}]_{12}]_0$

$r9$: membrane dissolution

$[[[\ ]_{P1} \cdots [\ ]_{Pn}[\ ]_{\mathrm{master}}]_1]_0 \rightarrow \boldsymbol{\lambda}$

$R10$: link creation

$r10 \equiv [[[]_{P1} \cdots [\ ]_{Pn}[pBestScore^n]_{\mathrm{master}}]_{11}]_0$, $[[[\ ]_{P1} \cdots [\ ]_{Pn}[\mathrm{fitness}^n]_{\mathrm{master}}]_{12}]_0 \rightarrow [[[\ ]_{P1} \cdots [\ ]_{Pn}[pBestScore^n]_{\mathrm{master}}]_{11}]_0$-----
$[[[\ ]_{P1} \cdots [\ ]_{Pn}[\mathrm{fitness}^n]_{\mathrm{master}}]_{12}]_0$

$r11$: communication/rewriting

$r11 \equiv [[[pBestScore^n]_{\mathrm{master}}]_{11}]_0 \rightarrow [[[\mathrm{fitness}^n]_{\mathrm{master}}]_{11}]_0$, $[[[pBest^n]_{\mathrm{master}}]_{12}]_0 \rightarrow 1$

$\{[[[\mathrm{fitness}^n]_{\mathrm{master}}]_{12}]_0 < [[[pBestScore^n]_{\mathrm{master}}]_{11}]_0,\ 1 \leq n \leq p\}$

&

$[[[gBestScore]_{\mathrm{master}}]_{11}]_0 \rightarrow [[[pBestScore^n]_{\mathrm{master}}]_{11}]_0$, $[[[gBest^n]_{\mathrm{master}}]_{12}]_0 \rightarrow 1$

$\{[[[pBestScore^n]_{\mathrm{master}}]_{11}]_0 < [[[gBestScore]_{\mathrm{master}}]_{11}]_0,\ 1 \leq n \leq p\}$

&

Table 1: Continued.

$[[[\text{converge}]_{\text{master}}]_{11}]_0 \rightarrow \min[[[[pBestScore]_{p1}^{pn}]_{\text{master}}]_{11}]_0$
$r12$: communication
$r12 \equiv [[[\text{position}]_{p1}^{pn}]_{12}]_0 \rightarrow [[[[\text{position}]_{p1}^{pn}]_{\text{master}}]_{12}]_0$
r13: communication/rewriting
$[[[\text{position}^n, c1, c2, c, w, pBest, gBest, p, \text{max\_}c, \text{rand}]_{\text{master}}]_{12}]_0 \rightarrow [[[\text{Velocity}]_{\text{master}}]_{12}]_0$
$[[[\text{Velocity}]_{\text{master}}]_{12}]_0 \rightarrow [[V\text{max}]_{\text{master}}]_{12}\{\text{Velocity} > V\text{max}\}$
$[[[\text{Velocity}]_{\text{master}}]_{12}]_0 \rightarrow [[-V\text{max}]_{\text{master}}]_{12}\{\text{Velocity} < -V\text{max}\}$
$[[[[\text{position}]_{p1}^{pn}]_{\text{master}}]_{12}]_0 \rightarrow 1\{\text{rand} \le 1/(1+\exp(-2 * \text{Velocity})\}$
$[[[[\text{position}]_{p1}^{pn}]_{\text{master}}]_{12}]_0 \rightarrow 0\{\text{rand} > 1/(1+\exp(-2 * \text{Velocity})\}$
r14: output/link creation/communication & rewriting
$[[[[\text{position}]_{p1}^{pn}]_{\text{master}}]_{12}]_0 \rightarrow [[[\text{position}]_{p1}^{pn}]_{12}]_0$
$[[\,]_{12}]_0, [[\,]_{\text{position}}]_0 \rightarrow [[\,]_{12}]_0\text{----}[[\,]_{\text{position}}]_0$
$[[[\text{position}]_{p1}^{pn}]_{12}]_0 \rightarrow [[[\text{position}]_{p1}^{pn}]_{\text{position}}]_0$
$r15$: division rule
$[[]_{p1}\cdots[]_{pn}[]_{\text{master}}]_{12}[[]_{p1}\cdots[]_{pn}[]_{\text{master}}]_{121}[[]_{p1}\cdots[\,]_{pn}[\,]_{\text{master}}]_{122}$
$r16$: membrane dissolution
$[[\,]_{p1}\cdots[]_{pn}[]_{\text{master}}]_{12} \rightarrow \boldsymbol{\lambda}$

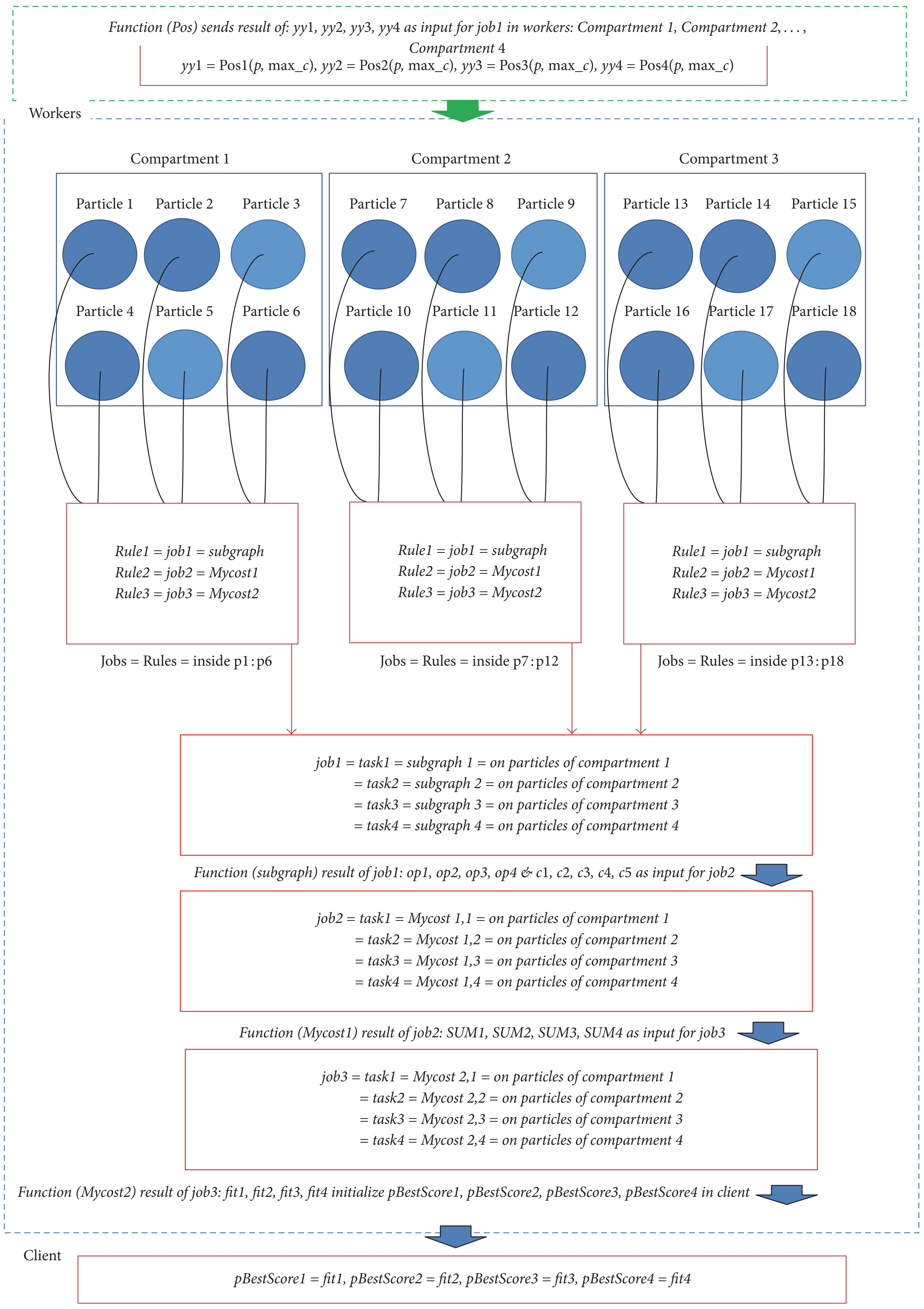
Dash lines means making link and making connection between two membranes.

random numbers which are p sets in maximal will be generated. Through $r3$, each set of random numbers will enter the independent membranes so-called Ps.

In $r4$, with having an array of sample data called "a" and the random numbers generated and saved in position objects in the last stage we will choose related gene of random number from the array "a"; this means we will look at the array "a" for the value of the gene which carries the random number "position." This processed gene is called NGENES. By adding a row at the end of the chosen genes, we can indicate to which sample each column belongs. This action makes another set, the so-called "NewGENES." Other objects, like "Q" or "c," will help to keep some value, for example, a total number of genes in each set; for example, "c" keeps the value of the total number of genes in each set and "Q" keeps the genes inside each compartment. $r5$ calculates dissimilarity and signal-to-noise value for each set of genes with the symbol indications of "sum_diss" and "snr," respectively. Then fitness value for each set will be calculated and shown by the symbol "FIT." To find the best fitness value among the sets of genes, $r6$ will create a link among the membranes through a master membrane, the so-called "master." Then through the $r7$, all the newly linked membranes will be able to communicate to extract the minimum fitness value, which will be called "*pBestScore*" for the first run and "*gBestScore*" for the second run, and the gene IDs have contributed to make this fitness value saved in the objects called "Q." Through $r8$, a division rule will create new particles called Ps for the evolving session of selected genes to other ones. According to the MObPSO itself, evolving the genes will produce objects such as *pBest, gBest,* Velocity, $c1$, $c2$, w, Vmax, and s. Then, $r9$ can dissolve (λ) previous Ps and all their connections with other membranes and $r10$ make new connections between the new Ps with the "master" membrane of the last session. In $r11$, after evolving the genes to new sets of genes inside each Ps then a calculation of "fitness" will be done, and it will compare with previous "*pBestScore*" and the value of "*pBest*" and "*gBest*" will initialize by "1." Moreover, in the second round, the values of "*pBestScore*" and "*gBestScore*" will compare with each other. At the end, the minimum value of "*pBestScore*" will be the value of the object called "converge." In $r12$, the position of the genes has been contributed in the particular process and will be saved in the membrane called "Master." $r13$ basically explains the typical formulation of swarm optimization through the objects of position, $c1$, $c2$, c, w, *pBest, gBest, p,* max_c, rand to produce new velocity and new positions. The new positions will pass to $r14$ to make particles from newly generation positions for random genes and the process will continue again by new genes.

The explanation of objects and how they have evolved based on the rules are defined in different steps as Step 1: initialization, and Step 2: evolution. Moreover, Figure 1 also explains Steps 1 and 2 plus Step 3: interaction with client, and Step 4: collecting the final result.

Step 1 (initialization). Rules from the proposed sequential KP-MObPSO translate into jobs in the proposed multiple core KP-MObPSO feature selection method. As Algorithm 1 and Figure 1, first, binary positions of genes should be randomly initialized by the client. Thus, rule numbers $r1$, $r2$, and $r3$ in sequential KP-MObPSO are defined as function F_n = Pos. To execute Fn = Pos on four compartments in multiple core KP-MObPSO, it is divided into four tasks including Pos1, Pos2, Pos3, and Pos4. Pos1 initializes the random position of genes inside 6 particles of compartment 1, Pos2 initializes random position of genes inside 6 particles



Function (Pos) sends result of: yy1, yy2, yy3, yy4 as input for job1 in workers: Compartment 1, Compartment 2, . . . , Compartment 4
yy1 = Pos1(p, max_c), yy2 = Pos2(p, max_c), yy3 = Pos3(p, max_c), yy4 = Pos4(p, max_c)
Workers
Compartment 1
Compartment 2
Compartment 3
Particle 1
Particle 2
Particle 3
Particle 4
Particle 5
Particle 6
Particle 7
Particle 8
Particle 9
Particle 10
Particle 11
Particle 12
Particle 13
Particle 14
Particle 15
Particle 16
Particle 17
Particle 18
Rule1 = job1 = subgraph
Rule2 = job2 = Mycost1
Rule3 = job3 = Mycost2
Jobs = Rules = inside p1 : p6
Jobs = Rules = inside p7 : p12
Jobs = Rules = inside p13 : p18
job1 = task1 = subgraph 1 = on particles of compartment 1
= task2 = subgraph 2 = on particles of compartment 2
= task3 = subgraph 3 = on particles of compartment 3
= task4 = subgraph 4 = on particles of compartment 4
Function (subgraph) result of job1: op1, op2, op3, op4 & c1, c2, c3, c4, c5 as input for job2
job2 = task1 = Mycost 1,1 = on particles of compartment 1
= task2 = Mycost 1,2 = on particles of compartment 2
= task3 = Mycost 1,3 = on particles of compartment 3
= task4 = Mycost 1,4 = on particles of compartment 4
Function (Mycost1) result of job2: SUM1, SUM2, SUM3, SUM4 as input for job3
job3 = task1 = Mycost 2,1 = on particles of compartment 1
= task2 = Mycost 2,2 = on particles of compartment 2
= task3 = Mycost 2,3 = on particles of compartment 3
= task4 = Mycost 2,4 = on particles of compartment 4
Function (Mycost2) result of job3: fit1, fit2, fit3, fit4 initialize pBestScore1, pBestScore2, pBestScore3, pBestScore4 in client
Client
pBestScore1 = fit1, pBestScore2 = fit2, pBestScore3 = fit3, pBestScore4 = fit4

Figure 1: Continued.

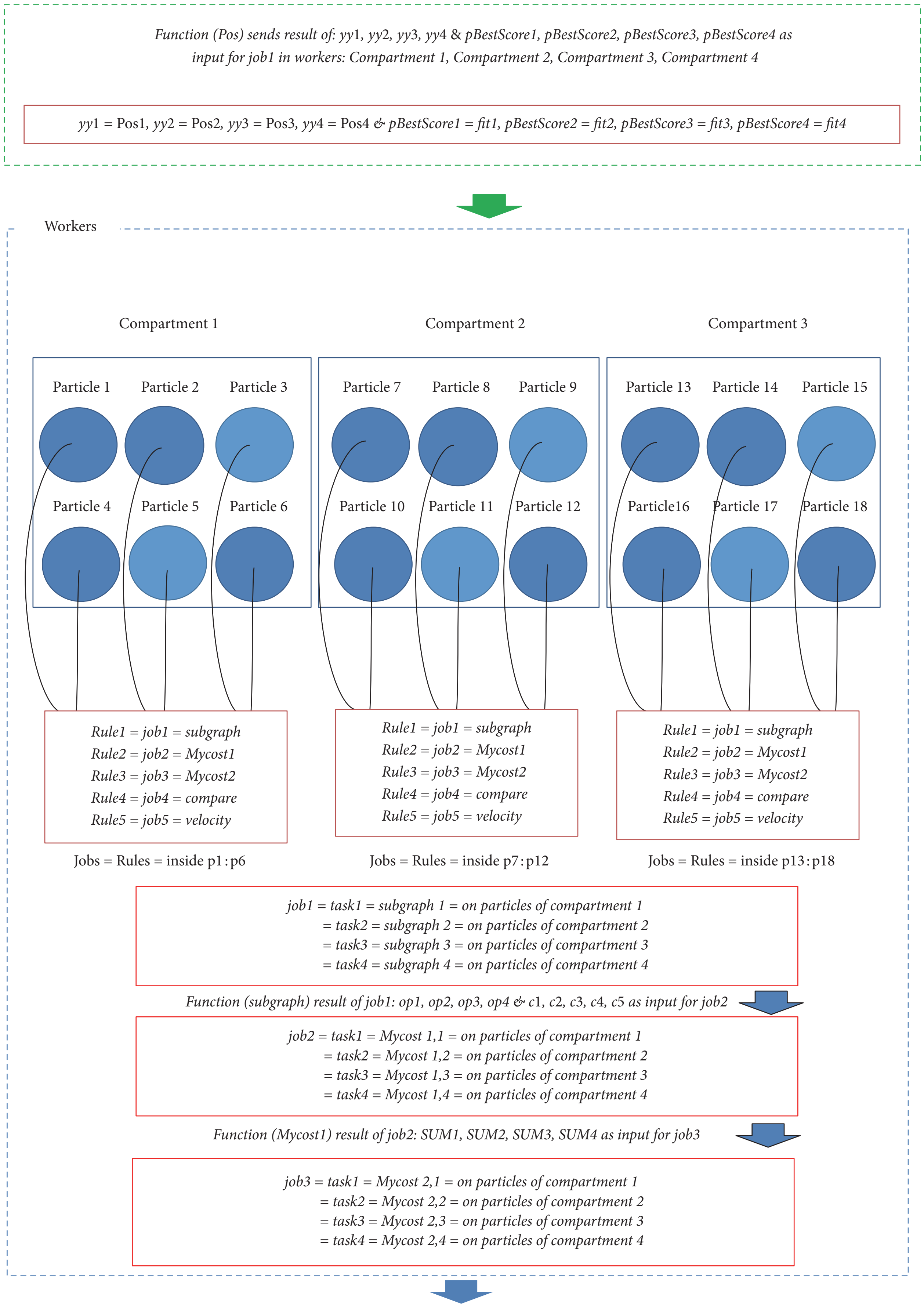
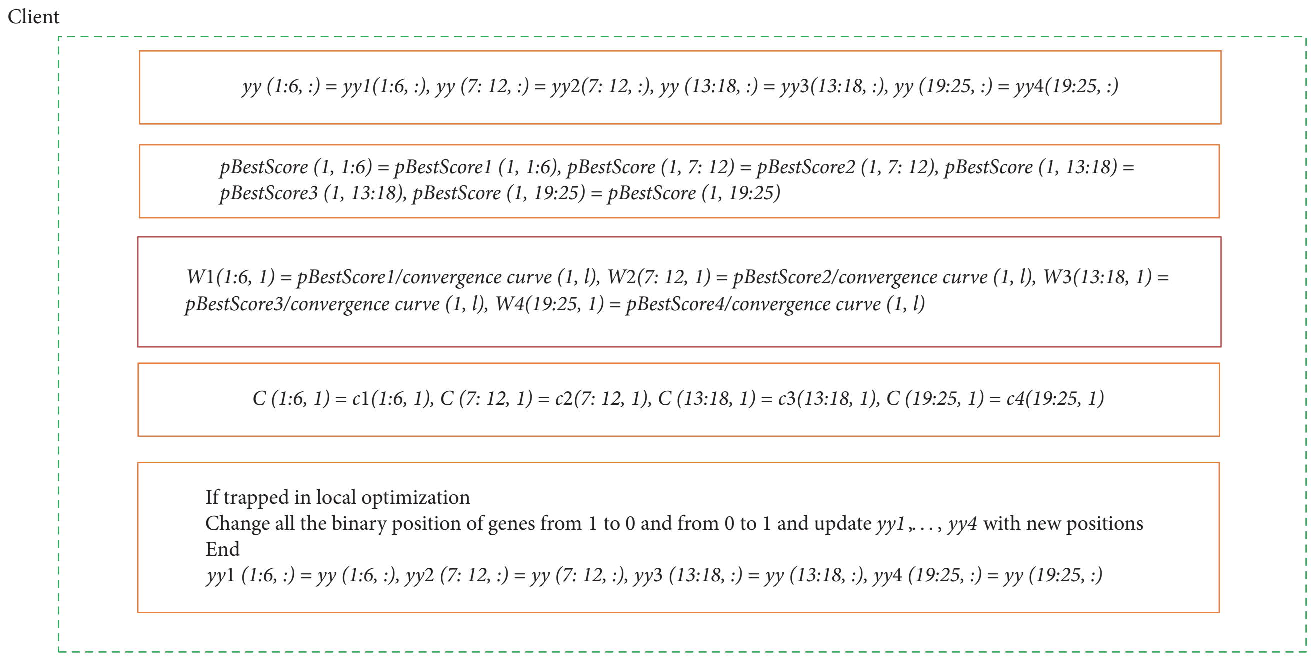

Function (Pos) sends result of: yy1, yy2, yy3, yy4 & pBestScore1, pBestScore2, pBestScore3, pBestScore4 as input for job1 in workers: Compartment 1, Compartment 2, Compartment 3, Compartment 4
yy1 = Pos1, yy2 = Pos2, yy3 = Pos3, yy4 = Pos4 & pBestScore1 = fit1, pBestScore2 = fit2, pBestScore3 = fit3, pBestScore4 = fit4
Workers
Compartment 1
Compartment 2
Compartment 3
Particle 1
Particle 2
Particle 3
Particle 4
Particle 5
Particle 6
Particle 7
Particle 8
Particle 9
Particle 10
Particle 11
Particle 12
Particle 13
Particle 14
Particle 15
Particle16
Particle 17
Particle 18
Rule1 = job1 = subgraph
Rule2 = job2 = Mycost1
Rule3 = job3 = Mycost2
Rule4 = job4 = compare
Rule5 = job5 = velocity
Jobs = Rules = inside p1 : p6
Jobs = Rules = inside p7 : p12
Jobs = Rules = inside p13 : p18
job1 = task1 = subgraph 1 = on particles of compartment 1
= task2 = subgraph 2 = on particles of compartment 2
= task3 = subgraph 3 = on particles of compartment 3
= task4 = subgraph 4 = on particles of compartment 4
Function (subgraph) result of job1: op1, op2, op3, op4 & c1, c2, c3, c4, c5 as input for job2
job2 = task1 = Mycost 1,1 = on particles of compartment 1
= task2 = Mycost 1,2 = on particles of compartment 2
= task3 = Mycost 1,3 = on particles of compartment 3
= task4 = Mycost 1,4 = on particles of compartment 4
Function (Mycost1) result of job2: SUM1, SUM2, SUM3, SUM4 as input for job3
job3 = task1 = Mycost 2,1 = on particles of compartment 1
= task2 = Mycost 2,2 = on particles of compartment 2
= task3 = Mycost 2,3 = on particles of compartment 3
= task4 = Mycost 2,4 = on particles of compartment 4

Figure 1: Continued.



```
yy (1:6, :) = yy1(1:6, :), yy (7: 12, :) = yy2(7: 12, :), yy (13:18, :) = yy3(13:18, :), yy (19:25, :) = yy4(19:25, :)

pBestScore (1, 1:6) = pBestScore1 (1, 1:6), pBestScore (1, 7: 12) = pBestScore2 (1, 7: 12), pBestScore (1, 13:18) =
pBestScore3 (1, 13:18), pBestScore (1, 19:25) = pBestScore (1, 19:25)

W1(1:6, 1) = pBestScore1/convergence curve (1, l), W2(7: 12, 1) = pBestScore2/convergence curve (1, l), W3(13:18, 1) =
pBestScore3/convergence curve (1, l), W4(19:25, 1) = pBestScore4/convergence curve (1, l)

C (1:6, 1) = c1(1:6, 1), C (7: 12, 1) = c2(7: 12, 1), C (13:18, 1) = c3(13:18, 1), C (19:25, 1) = c4(19:25, 1)

If trapped in local optimization
Change all the binary position of genes from 1 to 0 and from 0 to 1 and update yy1,. . . , yy4 with new positions
End
yy1 (1:6, :) = yy (1:6, :), yy2 (7: 12, :) = yy (7: 12, :), yy3 (13:18, :) = yy (13:18, :), yy4 (19:25, :) = yy (19:25, :)
```

Figure 1: KP-MObPSO on multicore processing.

of compartment 2, Pos3 initializes random position of genes inside 6 particles of compartment 3, and finally, Pos4 initializes the random position of genes inside 7 particles of compartment 4. Second, 4 workers are configured to execute 3 jobs on the initialized particles of 4 compartments. These 3 jobs are the rule number $r4$ for function subgraph, rule number $r5$, $r6$, $r7$ for functions Mycost1, and Mycost2 which are translated from sequential KP-MObPSO to multiple core KP-MObPSO. The outputs of job1 (Fun = subgraph) will be used as input arguments for job2 (Fun = Mycost1) and so on till the outputs of job3 (Fun = Mycost2) initialize the value of *pBestScore* in the client to be used in the second step named evolution step.

Step 2 (evolution). As Algorithm 1 and Figure 1, in this step function Pos produce new binary positions of genes again randomly. The new position of genes and the generated value of *pBestScore* on the previous step initialize 4 compartments of particles on 4 workers. Like the previous step, job numbers 1, 2, and 3 will be executed on 4 compartments to produce fitness value for each particle. Then, rules numbers $r8$, $r9$, $r10$, and $r11$ in sequential KP-MObPSO translate to job4 as Fn = compare. This job4 applies for all particles inside compartment 1 as task compare 1, all particles of compartment 2 as task compare 2, all particles of compartment 3 as task compare 3, and all particles of compartment 4 as task compare 4 which generally compare the fitness value of each particle with its previously produced *pBestScore* and update the *pBestScore* value for each particle with the fitness value if the fitness value of particle is less than its *pBestScore* value. The outputs of this job will enter job5 to update the velocity of gene positions. This job is defined based on the rule numbers $r12$, $r13$, and $r14$ in sequential KP-MObPSO. Making the 4 compartments of particles is the results of division rule, $r15$. The result of the updating gene position will transfer to the client section through the next step and membranes can dissolve as $r16$.

Step 3 (interaction with client). Totally evolutionary steps will be executed 100 times. However, after each iteration, results of workers should combine together in client section to initialize next iteration. To do so, according to Figure 1, in client section all the results of updated positions in 4 compartments combine as yy, all the results of updated *pBestScores* in 4 compartments combine as *pBestScore*, and all the results of a random number of genes inside each particle combine as C. By these combinations second and third iterations will launch. For iteration number 3 and above, the client checks whether the production of *pBestScores* has fallen in a local trap or not. If the value of the convergence curve variable which is the minimum of *pBestScores* in all 25 particles after each iteration repeatedly is the same value for three times, then all the positions of genes will reverse in client and new positions will launch new iteration continuously.

Step 4 (collecting final result). After 100 times the iteration, value of the convergence curve variable gathers a minimum of *pBestScores* in all 25 particles after each iteration. Minimum value in the convergence curve matrix is the record of 100 minimum *pBestScores* calls *gBestScore*. The set of the gene numbers has already produced such a minimum value of *gBestScore* which are marker genes. These marker genes are the final result of KP-MObPSO feature selection method.

4. Execution of KP-MObPSO on Multicore Processing

To implement multiple core processing of KP-MObPSO, we integrate the two types of parallel programming. First, all the

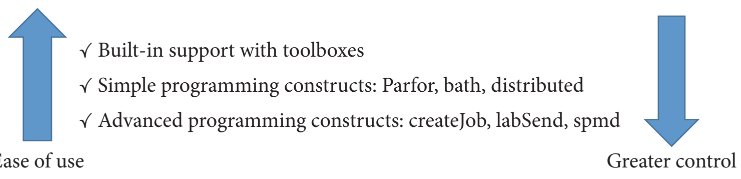

√ Built-in support with toolboxes
√ Simple programming constructs: Parfor, bath, distributed
√ Advanced programming constructs: createJob, labSend, spmd
Ease of use
Greater control

Figure 2

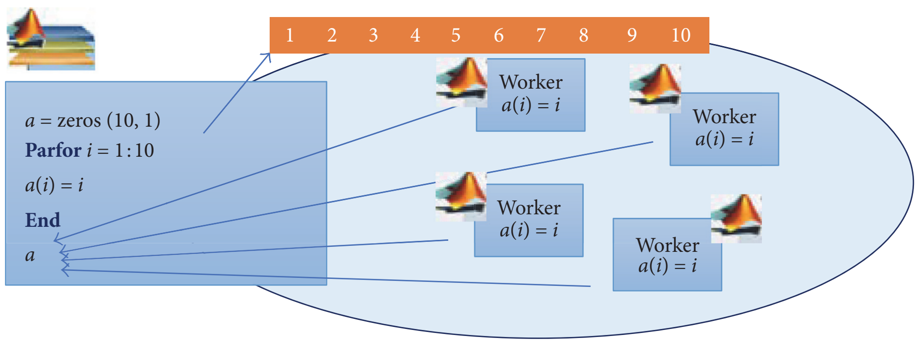


Figure 3: The mechanics of Parfor loops.

functions are defined in sequential MObPSO and rewritten based on Parfor independent loops. All the functions including Pos, subgraph, Mycost1, and Mycost2 are rewritten based on 25 independent Parfor loops. Second, each function is defined as a job based on createJob method to be executable on the particles. Particles are assumed to be located inside four compartments, and each job consists of four tasks each running on one compartment.

4.1. Parallel Computing. Computing in a parallel way is applicable through various methods such as clusters, multicores, and GPUs. The main aim in all parallel computing methods is to increase the speed of computing through using more processors and more memory compared with a single machine. From the programmers' point of view, languages such as C/C++/FORTRAN and MPI (Message Passing Interface) are hard to use, and languages such as C/FORTRAN and MPI are time consuming. That is the reason programming via MATLAB is highly preferred by programmers due to applying high-level language and use of mix features that can produce productive code. Using MATLAB for parallel computing support may lead to decrease in computing time through options such as parameter sweeping. In fact, parallel computing in MATLAB can take advantage of two types of parallelism, first implicit multiprocessing which uses built-in multithreading and second explicit multiprocessing which uses toolboxes and distributed computing on clusters. The explicit multiprocessing method is employed in this paper. According to Figure 2 that categorized parallel computing options based on ease of use and greater control opportunity, we will use Parfor and createJob to take advantages of greater control simultaneous with having ease of use.

Parfor: A Parallel FOR Loop. The key word, "Parfor," as in Figure 3, is defined to use instead of "for" keyword where a parallel loop is needed to parallelize an instruction. The difference is that each iteration in Parfor divided between different "workers" and the execution order of iterations is not known, although the results of different executions will be gathered and will be sent to MATLAB. Parfor is the simplest way to parallelize a loop statement. As usual execution starts by executing client codes and when it reaches Parfor, the iterations automatically divide between workers, and after all executions are done, the result will be collected in client again. As a prerequirement, it is necessary that all iterations run as an independent loop due to some limitation in access to the stored data. Thus, every loop in parallel computing makes an opportunity to distribute a series of independent tasks on a series of workers. Detection of workers and exchanging whatever necessary data between client and workers will be done automatically through MATLAB itself.

CreateJob: Life Cycle of a Job. The process of executing job from client side to workers is illustrated in Figure 4, which is called job's life cycle. As it is shown in Figure 4, the life cycle of a job is a process that starts with creating job on client side and finishes by running a job on workers. Between starting and finishing point there are two other stages, the so-called pending phase and queuing phase. In pending phase, the job is already created on scheduler through the command *createJob* function and tasks are already added to the jobs. In queuing phase, the pending job is already submitted to the queue and according to the sequence of submitting scheduler will send jobs for execution until all queued jobs are finished. The last stage for a job is when it reaches the top of the queue and will be sent for execution phase. In running phase scheduler will distribute job's tasks to different workers to run.

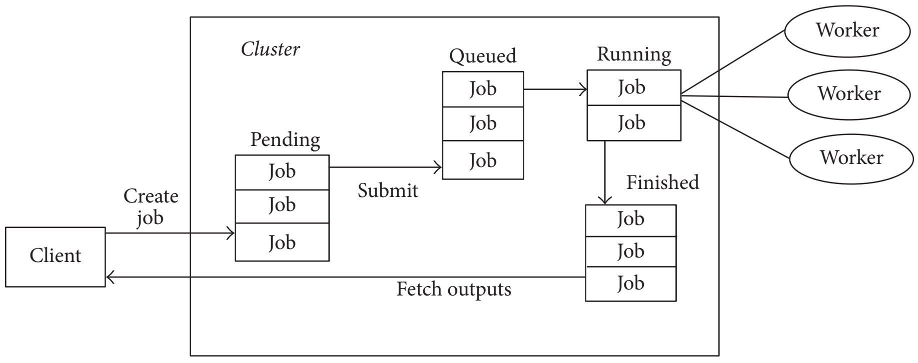


FIGURE 4: Life cycle of a job.

In this case, more workers are available compared with the number of existing job's tasks; the scheduler will send another job from the queue to running phase. It means more than one job will be in running phase at a time. When the job's execution finishes, the result will be gathered with the function "*fetchOutputs.*" Any problem occurs when the scheduler attempts to run a job's tasks will lead to a fail error; moreover, in the case of deleting a job, its status will be updated to delete one and its state will be available all the time the job objects are still in the client.

Configuration. We define parallel configuration on MATLAB by cluster, and we have defined jobs and job's tasks in the MATLAB client. As predefinitions, the number of physical cores on system indicates the maximum number of workers in MATLABPOOL. In this study, there are 4 cores in our machine; therefore, we change the configuration as follows:

> Parallel > Manage Cluster Profiles > local > Edit and change the value of "Number of workers to start on your local machine" to 4

The proposed KP-MObPSO feature selection methods assume to have 25 particles. To configure 25 particles on 4 cores of the system, we assume to have 4 compartments including 6 particles, 6 particles, 6 particles, and 7 particles, respectively. Each particle consists of a random number of gene objects from 1 to 100 and sets of rules will be executed on objects. Since then, we will assign each rule to a specific job and each job will assign to each compartment through a set of 4 tasks repeating execution of rules on each object inside a particle. Thus, the multicore KP-MObPSO model will be initialized by

(i) 4 compartments including 6, 6, 6, and 7 particles,

(ii) 4 jobs/rules inside each particle,

(iii) 4 tasks assigned to each rule/jobs to be executed on particles inside the 4 compartments,

(iv) priority of rules Pos > subgraph > Mycost1 > Mycost2 > compare > velocity being effective in the priority of tasks.

5. Material and Method

In this paper, cell line datasets of colorectal cancer and breast cancer are downloaded from publicly available datasets in the Gene Expression Omnibus (GEO) repository (https://www.ncbi.nlm.nih.gov/gds).

5.1. Methods and Data. Totally, six samples of colorectal cancer are used according to Tables 1 and 2, respectively, explaining the specification of samples. When samples of a microarray dataset represent normal (benign) and cancer (malignant) tissue, classifying such samples is referred to as binary classification. Otherwise, when samples represent various subtypes of cancer, classification is known as multiclass cancer classification. In both cases, genes with significantly different expression in the two categories (normal and tumor or two different subtypes of cancer) are labeled as differentially expressed genes. In this paper, our aim was to find these indicator genes or marker genes, to distinguish normal and tumor genes (binary classification) in colorectal dataset through multicore KP-MObPSO method faster than its counterpart sequential KP-MObPSO method.

5.2. Colorectal Data. According to Table 2, samples were provided from two subtypes including subtype 1.1 and 2.1 with platform IDs: GPL570 (Affymetrix U133 Plus 2.0) and series GSE35896 to compare between 62 colorectal cancers. First three samples, including GSM877130, GSM877141, and GSM877142, were from the subtype 1.1, and the last three samples, GSM877127, GSM877138, and GSM877140, were from subtype 2.1. Before evaluating the model, we preprocessed the data. The number of genes in the raw data was 54676. A list of significant genes, including normal and tumor genes, was highlighted according to the results of a study [8] where the clinical, pathological, and biological features of different subtypes of colorectal tumors were compared. As a result, 382 tumor genes were selected according to the subtypes 1.1 and 2.1 in the initial cell line dataset. The SNR was calculated according to (5) for both normal and tumor genes. For each gene, the first three sets of samples were categorized as $C1$ (subtype 1.1), and the rest were classified as $C2$ (subtypes 2.1). The SNR values were sorted in descending order, and 100

Table 2: Specification of colorectal cancer dataset.

Data Set Record GDS4379				
Title: Colorectal Cancer Tumors				
Summary: analysis of primary colorectal cancer (CRC) tumors. CRC is a heterogeneous disease. Results provide insight into stratifying CRC tumor samples into subtypes and tailoring treatments for the CRC subtypes. Organism: *Homo sapiens*				
Platform: GPL570: [HG-U133_Plus_2] Affymetrix Human Genome U133 Plus 2.0 Array				
Citation: Schlicker A, Beran G, Chresta CM, McWalter G, et al. Subtypes of primary colorectal tumors correlate with response to targeted treatment in colorectal cell lines. BMC Med Genomics 2012 Dec 31; 5:66.				
PMID: 23272949				
Platform ID: GPL570				
Series: GSE35896, gene expression data from 62 colorectal cancer cases				
Samples	Cell line	Tissue	Disease state	Genotype/variation
		Subtype: 1.1		
GSM877130	CRC_42	Large intestine	Adenocarcinoma	microsatellite.status: MSS
GSM877141	CRC_45	Large intestine	Adenocarcinoma	microsatellite.status: MSS
GSM877142	CRC_61	Large intestine	Adenocarcinoma	microsatellite.status: MSS
		Subtype: 2.1		
GSM877127	CRC_01	Large intestine	Adenocarcinoma	microsatellite.status: MSI
GSM877138	CRC_02	Large intestine	Adenocarcinoma	microsatellite.status: MSI
GSM877140	CRC_19	Large Intestine	Adenocarcinoma	microsatellite.status: MSI

maximum SNR values (50 maximum SNR values for normal genes and 50 maximum SNR values for tumor genes) were selected for further analysis. Then the signal-to-noise ratio (SNR) value (node weight) corresponding to each feature is calculated using mean and standard deviation (s.d.) of class 1 ($c1$) and class 2 ($c2$).

During the final stage of preprocessing, the 100 genes were normalized according to (6). For normalization, minimum and the maximum value of each gene (column) are calculated first. Then normalization is done, where g_{ij} presents gene expression value of the ith sample of the jth gene and j presents gene expression values of j-gth gene. Therefore, all the values of the dataset (100 rows of genes times six sets of samples) were transformed to a value between 0 and 1 $[0, 1]$.

$$|\text{SNR}| = \left|\frac{\text{mean}\,(c1) - \text{mean}\,(c2)}{\text{s.d.}\,(c1) + \text{s.d.}\,(c2)}\right| \tag{5}$$

$$\text{Normalize}\left(g_{ij}\right) = \frac{g_{ij} - \text{minimum}\left(g_{ij}\right)}{\text{maximum}\left(g_{ij}\right) - \text{minimum}\left(g_{ij}\right)}. \tag{6}$$

The prepared data for colorectal dataset are evaluated by KP-MObPSO and MObPSO to search for gene markers. MATLAB R2014a was used for object-oriented coding of the entire proposed model. Then, Weka 3.6.9 is used to classify the marker genes resulting from KP-MObPSO and MObPSO. Classification accuracy and ROC by Weka are used for comparison. Using the Weka software, SMO (Implements, John Platt's sequential minimal optimization algorithm for training a support vector classifier) works based on support vector machine theory. A linear SMO ($K(x, y) = \langle x, y\rangle$) works to find the largest margin for separating hyperplane. It is defined according to the aggregation of distances from hyperplane to the most nearest positive or negative points. It is projected that as the margin gets larger the classifier will be more general. When the case is not separable, linear support vector machine looks for a trade-off that maximizes the margin and on the other hand minimizes errors.

6. Evaluation and Discussion

The aim of our proposed model (KP-MObPSO) was to build a more robust feature selection method about MObPSO. Therefore, to evaluate the performance of our proposed feature selection method for classification, we needed to consider the classification accuracy, which is the number of correct predictions made divided by the total number of predictions, multiplied by 100 (7). The numbers of true negatives (TN), false negatives (FN), true positives (TP), and false positives (FP) are used to compute the efficiency of the classifier.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \times 100\%. \tag{7}$$

To determine whether KP-MObPSO is adequate, classification accuracy alone is typically not sufficient. Therefore, we assessed the F-score based on precision and recall measures

Table 3: Performance metrics for classification accuracy for KP-MObPSO and MObPSO.

Accuracy	ROC area	F-measure	Sensitivity	Specificity
		KP-MObPSO		
87.50%	0.90	0.877	0.875	0.906
		MObPSO		
78.57%	0.72	0.785	0.786	0.792

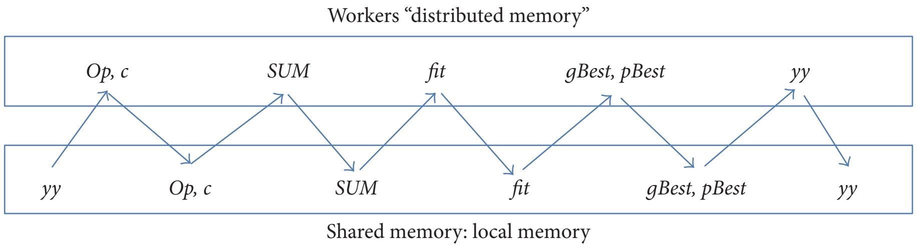


Figure 5: Architectural differences between distributed and shared memory.

(8) to evaluate the performance of our proposed feature selection model for binary classification.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$
$$F = 2 * \text{Precision} * \text{Recall Precision} + \text{Recall}.$$

To evaluate the accuracy of the proposed feature selection model, we use support vector machines. We divided 70% of the dataset into training and testing sets, and the proposed approach was applied to the training set. A candidate solution was obtained, with which the corresponding testing set was used for validation.

According to Table 3, regarding classification accuracy according to (7), there were 87.50% correctly classified genes from the KP-MObPSO method, which was higher than the 78.57% reported from the MObPSO method. Measures like ROC area, F-measure, sensitivity, and specificity of KP-MObPSO which are greater than MObPSO indicate much more liable model.

The proposed KP-MObPSO model is parallelized on multicore. In general, the main memory in parallel computer hardware can be classified into two kinds: shared memory and distributed memory. Shared memory is all processors interconnections with significant logical memory. Distributed memory refers to the fact that each processor has its own local memory. Indeed, accesses to local memory are typically faster than accesses to nonlocal memory [26, 27]. Figure 5 illustrates architectural differences between distributed and shared memory in the trace of objects based on the process of rules in KP-MObPSO as discussed before in Table 1.

Multicore chips do more work per clock cycle, are able to run at a lower frequency, and may enhance the performance of computation time. MATLAB includes the tic and toc functions which set a starting time and then return the elapsed time. The same command can be used in parallel. The MATLAB pool command itself can take some time. Typically, we want to ignore that and focus on the parallel computation. In the proposed multicore KP-MObPSO method, after launching workers through "*yy*," the outputs of jobs like "op, *c*, SUM, fit, *gBest*, *pBest*, and *yy*" need to be saved in client to send to the next jobs as input arguments. This frequent interaction between clients and workers takes time and does not lead to more improvement in timely execution of multicore KP-MObPSO.

7. Conclusion and Future Work

Due to the inherent large-scale parallelism feature of membrane computing, any membrane computing inspired model can fully represent this computation model only in the case of using the parallel platform. From the beginning of introducing this model, it was a big concern in all membrane related studies. For instance, to fully implement parallelism of such membrane computing model and to support an efficient execution [28] used a platform based on reconfigurable hardware. Without parallelism, all subsequent studies face a challenge of how to make rules available in all steps of computation. In [29] with sequential computing of membrane computing, the authors just had an option of using one membrane and made the rules periodically available based on time-varying sequential P system. Even by using minimal parallelism of rules more efficient model of membrane computing is achievable. This is because of the trading space for time. For example, in the study of [30] better performance has already achieved which is due to the parallel implementation of at least one rule from a set of rules to solve NP-complete problems in polynomial time.

Recently, several studies attempted to utilize membrane computing to improve intelligent algorithms. For instance, multicore processing used in the study of [31] utilized a membrane computing inspired genetic algorithm and [32] has highlighted parallelism in membrane computing in the case of solving the N-queens problem.

Regarding the modelling of particle swarm optimization via kernel P system, in the one hand, useful features of KP system, for example, its capability for changing the structure within modelling, and the rules such as division, dissolution, and link creation make variation of a P system more powerful to compare with those having only rewriting and communication rules. However, on the other hand, the most challenging part is the implementation of division rule when new membrane born to facilitate particle creation in PSO and even the multiple membranes born through division rule to facilitate multi-PSO.

The kernel P system as a variant of a P system is used to apply various rules of the kernel P system including communication/rewriting, division, input/output, link creation, and link dissolution. Thus, the model is not static; it generates compartments by division rule and makes links between compartments and dissolves them time by time. The MObPSO model is built based on kernel P system rules and the proposed model called KP-MObPSO. To examine the performance of hybrid KP-MObPSO model to compare with pure MObPSO, a real dataset of colorectal cancer dataset is tested on both models including the hybrid and the pure one. According to the methodology, preprocessing steps include filtering the initial dataset based on SNR factor and then normalization performed on the raw dataset. The results came out from executing preprocessed datasets on the proposed KP-MObPSO model. The results of performance measures indicate that empowering the MObPSO by kernel P system leads to improving accuracy, recall, and F-measure. It proves KP-MObPSO is a more robust model in comparison with pure MObPSO and the first objective of the paper is fulfilled through that.

The second objective of the paper is taking full advantage of membrane computing via parallel implementation of kernel P system rules. The proposed KP-MObPSO model is implemented on multicore and it made all the rules available at the same time for n number of particles and swarm optimization for all the objects inside the n number of particles can run in parallel. The architectural differences between CPUs and GPUs cause CPUs to perform better on latency-sensitive, partially sequential, single sets of tasks. In contrast, GPUs performs better with latency-tolerant, highly parallel, and independent tasks. Executing KP-MObPSO on multicore GPU will be considered as a solution for time improvement in future work.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper. The mentioned received funding in "Acknowledgments" did not lead to any conflicts of interest regarding the publication of this manuscript.

Acknowledgments

This work has been supported by the Science Fund of the Ministry of Science, Technology, and Innovation (MOSTI) (Malaysia; Grant code: 01-01-02-SF1104) and FRGS/1/2015/ICT04/UKM/02/3.

References

[1] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[2] G. Zhang, H. Rong, F. Neri, and M. J. Pérez-Jiménez, "An optimization spiking neural P system for approximately solving combinatorial optimization problems," *International Journal of Neural Systems*, vol. 24, no. 5, Article ID 1440006, 2014.

[3] T. Shiiba and A. Fujiwara, "Solving subset sum problem using EN P system with active membranes," in *Proceedings of the Joint 8th International Conference on Soft Computing and Intelligent Systems (SCIS '16) and 17th International Symposium on Advanced Intelligent Systems (ISIS '16)*, pp. 886–891, IEEE, Sapporo, Japan, August 2016.

[4] X. Wang, G. Zhang, F. Neri et al., "Design and implementation of membrane controllers for trajectory tracking of nonholonomic wheeled mobile robots," *Integrated Computer-Aided Engineering*, vol. 23, no. 1, pp. 15–30, 2016.

[5] M. García-Quismondo, M. Levin, and D. Lobo, "Modeling regenerative processes with membrane computing," *Information Sciences*, vol. 381, pp. 229–249, 2017.

[6] V. P. Singh, T. Prakash, N. S. Rathore, D. P. S. Chauhan, and S. P. Singh, "Multilevel thresholding with membrane computing inspired TLBO," *International Journal on Artificial Intelligence Tools*, vol. 25, no. 6, Article ID 1650030, 2016.

[7] D. Zhao, X. Liu, and W. Sun, "Power lines optimization algorithm for membrane computing based on google earth and ACIS," in *Proceedings of the International Conference on Human Centered Computing*, vol. 9567, pp. 960–965, Springer, 2016.

[8] G. Zhang, M. Gheorghe, and M. J. Pérez-Jiménez, *Real-Life Applications with Membrane Computing*, Springer, Berlin, Germany, 2016.

[9] M. Gheorghe, F. Ipate, C. Dragomir et al., "Kernel P systems—version I. Membrane computing," in *Proceedings of the 11th Brainstorming Week on Membrane Computing (BWMC '13)*, pp. 97–124, 2013.

[10] P. Mohapatra and S. Chakravarty, "Modified PSO based feature selection for Microarray data classification," in *Proceedings of the IEEE Power, Communication and Information Technology Conference (PCITC '15)*, pp. 703–709, IEEE, Bhubaneswar, India, October 2015.

[11] S. Kar, K. D. Sharma, and M. Maitra, "Gene selection from microarray gene expression data for classification of cancer subgroups employing PSO and adaptive K-nearest neighborhood technique," *Expert Systems with Applications*, vol. 42, no. 1, pp. 612–627, 2015.

[12] A. Chinnaswamy and R. Srinivasan, "Hybrid feature selection using correlation coefficient and particle swarm optimization on microarray gene expression data," in *Innovations in Bio-Inspired Computing and Applications*, pp. 229–239, Springer, 2016.

[13] M. Mandal and A. Mukhopadhyay, "A graph-theoretic approach for identifying non-redundant and relevant gene markers from microarray data using multiobjective binary PSO," *PLoS ONE*, vol. 9, no. 3, Article ID e90949, 2014.

[14] G. Zhang, M. Gheorghe, L. Pan, and M. J. Pérez-Jiménez, "Evolutionary membrane computing: a comprehensive survey and new results," *Information Sciences*, vol. 279, pp. 528–551, 2014.

[15] T. Y. Nishida, "An application of P system: a new algorithm for NP-complete optimization problems," in *Proceedings of the 8th World Multi Conference on Systems, Cybernetics and Informatics*, N. Callaos, W. Lesso, and B. Sanchez, Eds., pp. 109–112, 2004.

[16] G. Zhang, J. Cheng, M. Gheorghe, and Q. Meng, "A hybrid approach based on differential evolution and tissue membrane systems for solving constrained manufacturing parameter optimization problems," *Applied Soft Computing*, vol. 13, no. 3, pp. 1528–1542, 2013.

[17] G. Singh and K. Deep, "Hybridization of P systems and particle swarm optimization for function optimization," in *Proceedings of the 3rd International Conference on Soft Computing for Problem Solving*, pp. 258–395, Springer India, 2014.

[18] Q. Du, L. Xiang, and X. Liu, "P system based particle swarm optimization algorithm," in *Frontier and Future Development of Information Technology in Medicine and Education*, vol. 269 of *Lecture Notes in Electrical Engineering*, pp. 553–563, Springer, Dordrecht, The Netherlands, 2014.

[19] F. Zhou, G. Zhang, H. Rong et al., "A particle swarm optimization based on P systems," in *Proceedings of the 6th International Conference on Natural Computation (ICNC '10)*, IEEE, Yantai, China, August 2010.

[20] X. Y. Wang, G. X. Zhang, J. B. Zhao, H. N. Rong, F. Ipate, and R. Lefticaru, "A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning," *International Journal of Computers, Communications & Control*, vol. 10, no. 5, pp. 732–745, 2015.

[21] J. Cheng, G. Zhang, and T. Wang, "A membrane-inspired evolutionary algorithm based on population P systems and differential evolution for multi-objective optimization," *Journal of Computational and Theoretical Nanoscience*, vol. 12, no. 7, pp. 1150–1160, 2015.

[22] G. Zhang, F. Zhou, X. Huang et al., "A novel membrane algorithm based on particle swarm optimization for solving broadcasting problems," *Journal of Universal Computer Science*, vol. 18, no. 13, pp. 1821–1841, 2012.

[23] M. Gheorghe, F. Ipate, R. Lefticaru et al., "3-Col problem modelling using simple kernel P systems," *International Journal of Computer Mathematics*, vol. 90, no. 4, pp. 816–830, 2013.

[24] S.-W. Lin, K.-C. Ying, S.-C. Chen, and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1817–1824, 2008.

[25] X. Zhang, Y. Tian, and Y. Jin, "A knee point-driven evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 761–776, 2015.

[26] Introduction to Parallel Computing, https://computing.llnl.gov/tutorials/parallel_comp/#Whatis.

[27] Cuda Programming, http://cuda.ce.rit.edu/cuda_overview/cuda_overview.htm.

[28] V. Nguyen, D. Kearney, and G. Gioiosa, "An implementation of membrane computing using reconfigurable hardware," *Computing and Informatics*, vol. 27, no. 3, pp. 551–569, 2008.

[29] A. Alhazov, R. Freund, H. Heikenwälder, M. Oswald, Y. Rogozhin, and S. Verlan, "Sequential P systems with regular control," in *Membrane Computing: 13th International Conference, CMC 2012, Budapest, Hungary, August 28–31, 2012, Revised Selected Papers*, vol. 7762 of *Lecture Notes in Computer Science*, pp. 112–127, Springer, Berlin, Germany, 2013.

[30] G. Ciobanu, L. Pan, G. Păun, and M. J. Pérez-Jiménez, "P systems with minimal parallelism," *Theoretical Computer Science*, vol. 378, no. 1, pp. 117–130, 2007.

[31] A. Maroosi and R. C. Muniyandi, "Membrane computing inspired genetic algorithm on multi-core processors," *Journal of Computer Science*, vol. 9, no. 2, pp. 264–270, 2013.

[32] A. Maroosi and R. C. Muniyandi, "Accelerated simulation of membrane computing to solve the n-queens problem on multicore," in *Proceedings of the International Conference on Swarm, Evolutionary, and Memetic Computing*, pp. 257–267, Springer, 2013.

Hindawi
Journal of Optimization
Volume 2017, Article ID 8063767, 12 pages
https://doi.org/10.1155/2017/8063767

Research Article

A Hybrid Multiobjective Discrete Particle Swarm Optimization Algorithm for Cooperative Air Combat DWTA

Guang Peng, Yangwang Fang, Shaohua Chen, Weishi Peng, and Dandan Yang

School of Aeronautics and Astronautics Engineering, Air Force Engineering University, Xi'an, Shaanxi 710038, China

Correspondence should be addressed to Guang Peng; pg1445334307@163.com

Received 30 December 2016; Accepted 20 March 2017; Published 11 April 2017

Academic Editor: Maoguo Gong

A hybrid multiobjective discrete particle swarm optimization (HMODPSO) algorithm is proposed to solve cooperative air combat dynamic weapon target assignment (DWTA). First, based on the threshold of damage probability and time window constraints, a new cooperative air combat DWTA multiobjective optimization model is presented, which employs the maximum of the target damage efficiency and minimum of ammunition consumption as two competitive objective functions. Second, in order to tackle the DWTA problem, a mixed MODPSO and neighborhood search algorithm is proposed. Furthermore, the repairing operator is introduced into the mixed algorithm, which not only can repair infeasible solutions but also can improve the quality of feasible solutions. Besides, the Cauchy mutation is adopted to keep the diversity of the Pareto optimal solutions. Finally, a typical two-stage DWTA scenario is performed by HMODPSO and compared with three other state-of-the-art algorithms. Simulation results verify the effectiveness of the new model and the superiority of the proposed algorithm.

1. Introduction

The weapon target assignment (WTA) is a typical NP-complete constrained combinatorial optimization problem [1], which can be classified into two categories: static WTA (SWTA) and dynamic WTA (DWTA) [2, 3]. In SWTA, all the weapons attack targets in a single stage. In contrast, DWTA is much more complicated than SWTA, which takes the time window and resource constraints into account [4]. Besides, DWTA needs to deal with the new incoming targets and assesses the outcome of each engagement.

Most of the previous researches on WTA are focused on SWTA [4]. However, DWTA has begun to gain more attention of researchers since it was put forward by Hosein and Athans in 1990 [3]. Cai et al. [4] provided a survey of the research on DWTA problem and introduced some basic concepts on DWTA. Khosla [5] proposed a hybrid approach, which combines genetic algorithm (GA) with simulated annealing (SA) to solve a target-based DWTA problem. Liu et al. [6] analyzed the time and space restriction of the mathematical model of DWTA and proposed an adaptive memetic algorithm to obtain the suboptimum solution step by step. Chen et al. [7] established a generic asset-based DWTA model which incorporates four categories of constraints, namely, capability constraints, strategy constraints, resource constraints, and engagement feasibility constraints. Based on the asset-based DWTA model, Xin et al. [8] proposed a new technique for constraint handling. Moreover, Xin et al. [9] proposed an efficient rule-based heuristic, which uses the domain knowledge of DWTA in the form of three crucial rules, to solve asset-based DWTA problems. The proposed method has obvious advantage over the Monte Carlo method (MCM) with regards to solution quality and computation time. Wang et al. [10] applied intuitionistic fuzzy entropy of discrete particle swarm optimization (IFDPSO) algorithm to solve DWTA problem. Wang et al. [11] firstly established a multicombat step DWTA game model of UAV aerial combat and then presented a clonal selection optimization algorithm to solve the model.

However, the above WTA problems are focused on one objective (i.e., operational effects), ignoring the operational cost, while in actual combat situations, apart from considering the maximum of the damage to targets, the ammunition consumption should be also taken into account. Clearly, the two competitive objectives are conflicting, which implies that the WTA problem is a multiobjective optimization problem.

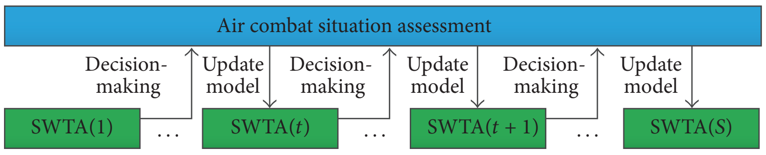


FIGURE 1: The schematic diagram of cooperative air combat DWTA model.

Up to now, there have been several studies on multiobjective optimization for DWTA problems. Liu et al. [12] and Zhou et al. [13] proposed an improved MOPSO algorithm to solve the multiobjective programming model of SWTA, respectively. Lötter and Van Vuuren [14] used NSGA-II to solve a triobjective DWTA model for surface-based air defense. Li et al. [15, 16] established deterministic and uncertain multiobjective optimization models of multistage WTA (MWTA) problem and modified two multiobjective optimizers, NSGA-II and MOEA/D, by adding an adaptive mechanism for solving the NWTA models. However, MOPSO easily falls into the local optimum; NSGA-II and MOEA/D have complex computation for DWTA. DWTA problem has higher requirements for the real-time performance and the convergence. In order to meet the real-time performance and the convergence accuracy simultaneously, this paper proposed an efficient HMODPSO algorithm to solve the DWTA multiobjective optimization problem. The proposed HMODPSO algorithm can generate obviously better DWTA decisions without the cost of overmuch extra computation time, which can improve the cooperative air combat effectiveness.

The rest of this paper is organized as follows. In Section 2, the cooperative air combat DWTA multiobjective optimization model based on the threshold of damage probability is formulated. Section 3 presents the structure of HMODPSO algorithm. The simulation results based on the proposed algorithm for a typical two-stage DWTA scenario are discussed in Section 4, and also the comparisons with three other state-of-the-art algorithms are conducted in this section. Conclusions and future work will be drawn in Section 5.

2. The Cooperative Air Combat Multiobjective Optimization Model for DWTA

Definition 1 (time window of target). The time window of target (t^T) is the exposure time of a target which the weapon can attack efficiently.

Definition 2 (time window of algorithm). The time window of algorithm (t^A) is the running time for solving the DWTA.

A complete cooperative air combat is a multistage offensive and defensive process. So the DWTA model can be regarded as the repetition of the SWTA model with the damage assessment. There are at most S stages of DWTA, which means no targets or weapons left after final stage assignment. In the cooperative air combat DWTA model, the "shoot-look-shoot" engagement policy is adopted. When entering the $t+1$ stage, it is necessary to determine a set of alive targets and remaining weapons. Therefore, it needs to observe the outcome of the t stage engagement and reformulate air combat situation assessment. The schematic diagram of cooperative air combat DWTA model is shown in Figure 1.

2.1. The DWTA Multiobjective Optimization Model. Assuming in the cooperative air combat that there are F flights in blue formation and each flight carries M_f $(f = 1, 2, \ldots, F)$ missiles, the total number of weapons is $M = \sum_{f=1}^{F} M_f$. At a certain time, the formation detects N targets, which can be attacked by the weapons. p_{ij} is the damage probability that the i-th missile attacks j-th target. w_j is the threat value of target j. Obviously, p_{ij} and w_j can be obtained by C^{4}I system according to the weapons' performance and air combat situation.

In order to describe the DWTA problem, a Boolean type decision matrix is introduced:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ & & \vdots & \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix}, \qquad (1)$$

where $x_{ij} = 1$ if the weapon i is assigned to the target j and $x_{ij} = 0$ otherwise. After all the weapons attack the target j cooperatively, the joint damage probability of the target j can be expressed as

$$P_j = 1 - \prod_{i=1}^{M} \left(1 - p_{ij}\right)^{x_{ij}}. \qquad (2)$$

In order to avoid wasting weapon resources in the single WTA optimization model, we define the maximum of the target damage efficiency and minimum of using weapon units as two objective functions. Additionally, aiming at targets with different threat values, the new model should guarantee the threshold of damage probability of each target. Hence, the formulation of the objective functions for the stage t is constructed:

$$\begin{aligned} \max \quad & f(t) = \sum_{j=1}^{N(t)} w_j(t) \left[1 - \prod_{i=1}^{M(t)} \left(1 - p_{ij}(t)\right)^{x_{ij}(t)}\right] \\ \min \quad & g(t) = \sum_{j=1}^{N(t)} \sum_{i=1}^{M(t)} x_{ij}(t), \end{aligned} \qquad (3)$$

where t is the stage index; $M(t)$ and $N(t)$ are the numbers of existing weapons and targets at the stage t, respectively; $p_{ij}(t)$

is the damage probability that the i-th missile attacks the j-th target at the stage t; $x_{ij}(t)$ is the Boolean type decision variable at the stage t.

The following three categories of constraints are incorporated in the cooperative air combat DWTA multiobjective optimization model:

$$P_j(t) \geq P_{dj}(t), \quad \forall t \in \{1, 2, \ldots, S\}, \ \forall j \in \{1, 2, \ldots, N\} \tag{4}$$

$$\sum_{j=1}^{N(t)} x_{ij}(t) \leq 1, \quad \forall t \in \{1, 2, \ldots, S\}, \ \forall i \in \{1, 2, \ldots, M\} \tag{5}$$

$$t_j^T(t) \geq t^A(t), \quad \forall t \in \{1, 2, \ldots, S\}, \ \forall j \in (1, 2, \ldots, N). \tag{6}$$

Constraint set (4) represents the threshold of damage probability of each target, the value of which depends on the decision-makers or the command system based on the current air combat situation. If one of the targets does not satisfy the threshold, the weapon target assignment is regarded as invalid assignment. Constraint set (5) reflects the capability of weapons attacking targets at the same time. In fact, a missile can only shoot one target each time. Constraint set (6) is very important for DWTA model, which takes the time windows of target and algorithm into account, influencing the engagement feasibility of weapons. In this case, it makes particularly high requirements for operational efficiency of the algorithm.

2.2. Constraints Handling. The DWTA multiobjective optimization model is a typical constrained nonlinear combinational optimization problem. To ensure the feasibility of solutions generated by the proposed algorithm, we make some preliminary treatments to constraints (4)–(6). First, we utilize the penalty function method to handle the constraint set (4). The function is defined as

$$h_j(t) = \min\left(0, P_j(t) - P_{dj}(t)\right). \tag{7}$$

For constraint set (5), it can be satisfied by solution encoding. And the constraint set (6) is satisfied by adopting the proposed algorithm.

Above all, the multiobjective optimization model for the cooperative air combat DWTA problem aforementioned can be formulated as

$$\begin{aligned} \max \quad & f(t) \\ & = \sum_{j=1}^{N(t)} w_j(t) \left[1 - \prod_{i=1}^{M(t)} \left(1 - p_{ij}(t)\right)^{x_{ij}(t)} \right] \\ & \quad - \sigma \sum_{j=1}^{N} h_j(t)^2 \end{aligned} \tag{8}$$

$$\min \quad g(t) = \sum_{j=1}^{N(t)} \sum_{i=1}^{M(t)} x_{ij}(t),$$

where σ is the penalty parameter and $\sigma = 1000$ is selected in this paper.

3. HMODPSO Algorithm for DWTA

3.1. Particle Position Encoding. Decimal encoding for particles is adopted in this paper. The length of the particle position encoding (denoted by D) is the total number of weapons. Each weapon is treated as a dimension of particle position encoding, and the value of each dimension indicates the number of the targets to which the weapon is assigned. Figure 2 provides an example to explain the encoding method ($M = 6$, $N = 4$, $t = 1$), and X is the corresponding 0-1 decision matrix. Obviously, such a particle position encoding method can guarantee that all the generated solutions satisfy the constraint set (5).

3.2. The Leader Particle Selecting. In PSO [17], each particle in the swarm corresponds to a potential solution of the optimization problem. In a D-dimensional search space, each particle has a position $X_i(k) = [X_{i,1}(k), X_{i,2}(k), \ldots, X_{i,D}(k)]$ and a velocity $V_i(k) = [V_{i,1}(k), V_{i,2}(k), \ldots, V_{i,D}(k)]$. During the movement, the personal best position (*Pbest*) for each particle is recorded as $P_i(k) = [P_{i,1}(k), P_{i,2}(k), \ldots, P_{i,D}(k)]$, and the global best particle (*Gbest*) of the swarm is referred to as $G(k) = [G_1(k), G_2(k), \ldots, G_D(k)]$. For specific DWTA problem, the velocity and position of the particle are updated by the following equations, respectively:

$$\begin{aligned} V_{ij}(k+1) & \\ = w V_{ij}(k) &+ c_1 r_1 \left(P_{ij}(k) - X_{ij}(k)\right) \\ &+ c_2 r_2 \left(G_j(k) - X_{ij}(k)\right) \end{aligned} \tag{9}$$

$$X_{ij}(k+1) = \text{round}\left(X_{ij}(k) + V_{ij}(k+1)\right) \tag{10}$$

$$V_{ij}(k+1) \longrightarrow \max\left(\min\left(V_{ij}(k+1), V_{\max}\right), V_{\min}\right) \tag{11}$$

$$X_{ij}(k+1) = \max\left(\min\left(X_{ij}(k+1), X_{\max}\right), X_{\min}\right), \tag{12}$$

where i represents the i-th particle of the swarm, j represents the j-th dimension in the search space, k is the number of current iterations, w is the inertia weight, c_1, c_2 are the acceleration coefficients, and $r_1, r_2 \in [0, 1]$ are uniformly distributed random variables. $X_{\max}$ and $X_{\min}$ are the lower and upper boundaries of the position, respectively; $V_{\max}$ and $V_{\min}$ are the lower and upper boundaries of the velocity, respectively. round() is said to be an integer operator. In MOPSO, selecting the global best position or the leader (*Gbest*) randomly from the external archive is a popular way [12]. However, the roulette method may damage evolution direction of the particles. Considering the weakness, a new method of selecting the leader particle is presented. First, the square root distance (SRD) [18] of particles X_1 and X_2 is defined as follows:

$$\text{SRD}\left(X_1, X_2\right) = \sum_{i=1}^{m} \sqrt{\left| f_i\left(X_1\right) - f_i\left(X_2\right) \right|}, \tag{13}$$

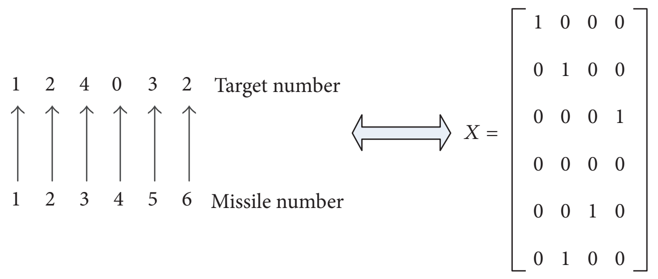

1 2 4 0 3 2 Target number
1 2 3 4 5 6 Missile number
$X = \begin{bmatrix} 1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&0&0\\0&0&1&0\\0&1&0&0\end{bmatrix}$

Figure 2: Particle position encoding.

where m represents the number of the objective functions. During the process of iteration, first, calculate all the particles' SRD with the nondominated solutions in the external archive; then, select the nondominated solution which has the minimum SRD as the leader particle; that is,

$$\min\left(\sum_{i=1}^{m}\sqrt{\left|f_i(Y_1)-f_i(X)\right|},\ldots,\sum_{i=1}^{L}\sqrt{\left|f_i(Y_L)-f_i(X)\right|}\right), \quad (14)$$

where $\{Y_1, Y_2, \ldots, Y_L\}$ represents the L nondominated solutions in the external archive and X represents the current position of particle. Particles can choose closer nondominated solutions as their leader particles through the new method, which can help the algorithm overcome the uncertainty of random selection and improve the convergence accuracy.

3.3. Repairing Operator. Clearly, unfeasible solutions which do not satisfy the constraint set (4) may be generated for solving the DWTA. To cope with this issue, a repairing operator is introduced [19], which includes two parts.

(1) Deleting the Redundant Allocation. First, select the unfeasible solutions and mark the targets satisfying the threshold of the damage probability. Then, try to delete the missile attacking the target with minimum damage probability. After deleting, if the target still satisfies the threshold, confirm the deletion; otherwise, retain the missile.

(2) Supplementing the Insufficient Allocation. First, select the targets which do not satisfy threshold of damage probability. Then mark the target with the minimum joint damage probability, and select one missile attacking the target with the maximum damage probability. Repeat the operation until no weapons are left or all the targets satisfy the threshold of damage probability.

The corresponding pseudocode of the repairing operator is shown in Algorithm 1.

```
Input. Unfeasible solutions
Output. New solutions
(1) Deleting the redundant allocation
  (1) for $j = 1:1:N$
  (2)     $Flag_j := \{i \mid x_{ij} = 1,\ i = 1,2,\ldots,M\}$
  (3)     while $P_j > P_{dj}$
  (4)         $k := \arg\min_{i\in Flag}\{P_{ij}\}$
  (5)         $P'_j = (P_j - P_{kj})/(1 - P_{kj})$
  (6)         if $P'_j > P_{dj}$
  (7)             $Flag_j : x_{kj} = 0$
  (8)             $P_j = P'_j$
  (9)         end if
  (10)    end while
  (11) end for
(2) Supplementing the insufficient allocation
  (12) $M' := \{i \mid \sum x_{ij} = 0,\ i = 1,2,\ldots,M\}$
  (13) $T' := \{j \mid P_j < P_{dj},\ j = 1,2,\ldots,N\}$
  (14) $P'' = \arg\min_{j\in T'}\{P_j\}$
  (15) while $M' > 0$ && $P'' > P_{dj}$
  (16)     $l := \arg\min_{j\in T'}\{P_j\}$
  (17)     $k := \arg\max_{i\in M'}\{P_{il}\}$
  (18)     $x_{kl} = 1$
  (19)     $M' = M' - 1$
  (20)     $P'' = \arg\min_{j\in T'}\{P_j\}$
  (21) end while
```

Algorithm 1: The pseudocode of the repairing operator.

3.4. Cauchy Mutation Operator. In order to further maintain the diversity of particles, the Cauchy mutation [20] is introduced into HMODPSO algorithm. When the current particle is dominated by *Pbest*, the Cauchy mutation is applied to disturb the current particle. The Cauchy mutation operator is defined as

$$X_{ij} = \begin{cases} X_{ij} + \operatorname{round}\left(\left(X_{\max} - X_{\min}\right) \times \operatorname{Cauchy}(0,1)\right), & \text{rand} \le p_b \\ X_{ij}, & \text{rand} > p_b \end{cases} \quad (15)$$

$$\operatorname{Cauchy}(0,1) = \tan\left(\left(\text{rand} - 0.5\right) \times \pi\right) \quad (16)$$

$$X_{ij} = \begin{cases} X_{\min}, & X_{ij} < X_{\min} \\ X_{\max}, & X_{ij} > X_{\max} \\ X_{ij}, & \text{elsewhile,} \end{cases} \quad (17)$$

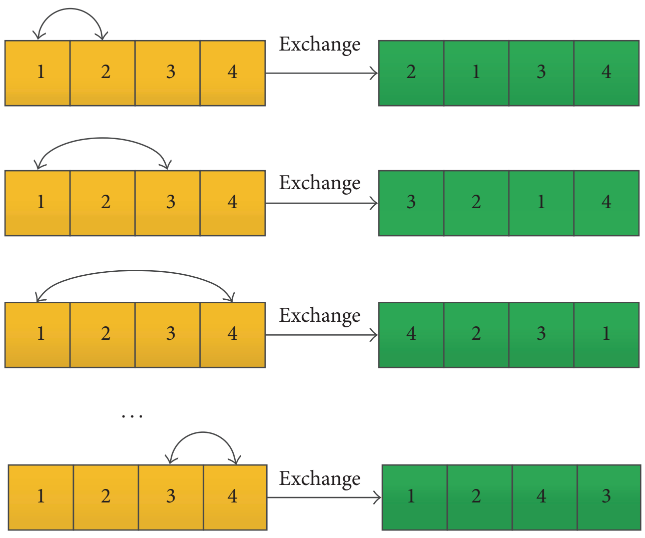


Figure 3: Neighborhood exchange criterion.

where rand $\in [0,1]$ is a uniformly distributed random value, Cauchy$(0,1)$ is a standard Cauchy-distributed random value, and the adapting mutation rate P_b is computed as

$$P_b = 1 - \sqrt{\frac{t}{\max It}}. \tag{18}$$

3.5. Neighborhood Searching Operator. Neighborhood search (NS) algorithm begins with an initial solution and searches the better solution in its neighborhood range [21]. At present, define a solution $X = \{1,2,3,4\}$, and a new vector can be obtained by exchanging any two positions of the solution. A collection of all the new vectors is referred to as exchange neighborhood shown in Figure 3.

NS algorithm is usually carried out at the end of the global search for individual local optimization, which can improve the convergence accuracy of algorithm. In MOPSO, the nondominated solutions in the external archive are regarded as leader particles to guide the swarm to fly. Additionally, the nondominated solutions are evolved into the Pareto optimal solutions set. In view of the particularity of the external archive, NS algorithm is introduced into the external archive. This paper proposed three kinds of operations: NS-1, NS-2, and NS-3. The procedures are described as follows.

For NS-1, do the following steps.

Step 1. Take each nondominated solution in the external archive as the initial solution.

Step 2. Exchange randomly two positions of the initial solution to get a new solution. If the new solution is better than the initial solution, then replace it with the new solution; otherwise, keep the initial solution.

For NS-2, do the following steps.

Step 1. Take each nondominated solution in the external archive as the initial solution.

```
NS-1
(1) for $i = 1 : 1 : L$
(2)     $NX = crossover(X, X)$
(3)     $evaluatefitness(NX)$
(4)     $X = Compare(X, NX)$
(5) end for
NS-2
(1) for $i = 1 : 1 : L$
(2)     for $j = 1 : 1 : D - 1$
(3)         for $k = j + 1 : 1 : D$
(4)         $NX = exchange(X(j,k))$
(5)         $evaluatefitness(NX)$
(6)         end for
(7)     end for
(8)     $X = Compare(X, \max(NX))$
(9) end for
NS-3
(1) for $i = 1 : 1 : L$
(2)     for $j = 1 : 1 : D - 1$
(3)         for $k = j + 1 : 1 : D$
(4)         $NX = exchange(X(j,k))$
(5)         $evaluatefitness(NX)$
(6)         $X = Compare(X, NX)$
(7)         end for
(8)     end for
(9) end for
```

Algorithm 2: The pseudocode of the local search operations.

Step 2. Select the best solution from the neighborhood range of the initial solution.

Step 3. If the best solution is better than the initial solution, then replace it with the new solution; otherwise, keep the initial solution.

For NS-3, do the following steps.

Step 1. Take each nondominated solution in the external archive as the initial solution.

Step 2. Exchange two positions of the initial solution in sequence to get a new solution; if the new solution is better than the initial solution, replace it with the new solution; then serve the new solution as the initial solution and repeat the above operation; otherwise, keep the initial solution.

The corresponding pseudocode of the local search operations is shown in Algorithm 2.

3.6. The Procedure of HMODPSO Algorithm. Summarizing the above procedures, we obtain the following pseudocode of the HMODPSO algorithm in Algorithm 3, where *Check-Boundaries* validates the particles to search in the solution space and max It is the maximum iterations. Furthermore, use the crowding distance sorting strategy [22] to maintain the external archive. Figure 4 illustrates the entire flowchart of HMODPSO.

4. Simulations and Results

In the experimental scenario, set $F = 5$, $M_f = 4$ $(f = 1, 2, 3, 4, 5)$, and $N = 10$, and obtain $M = 20$. When $t = 1$, the time window of each target t_j^T $(j = 1, 2, \ldots, 10)$ is $[8.3, 11.4, 16.2, 13.8, 20.1, 25.4, 19.6, 13.5, 17.2, 22.8]$; and each target's threat coefficient w_j $(j = 1, 2, \ldots, 10)$ is $[0.6, 0.7, 0.3, 0.5, 0.6, 0.35, 0.65, 0.55, 0.4, 0.75]$. The damage probability threshold of each target can be set to 0.9. The weapon's damage probability p_{ij} $(i = 1, 2, \ldots, 20,\ j = 1, 2, \ldots, 10)$ is

$$P(1) = \begin{bmatrix} 0.54 & 0.83 & 0.90 & 0.88 & 0.75 & 0.66 & 0.86 & 0.81 & 0.77 & 0.63 \\ 0.73 & 0.80 & 0.78 & 0.65 & 0.56 & 0.77 & 0.82 & 0.84 & 0.71 & 0.85 \\ 0.75 & 0.82 & 0.90 & 0.77 & 0.90 & 0.79 & 0.81 & 0.65 & 0.59 & 0.82 \\ 0.68 & 0.73 & 0.71 & 0.87 & 0.76 & 0.64 & 0.83 & 0.64 & 0.81 & 0.84 \\ 0.81 & 0.75 & 0.87 & 0.53 & 0.80 & 0.90 & 0.61 & 0.80 & 0.73 & 0.81 \\ 0.85 & 0.80 & 0.91 & 0.85 & 0.85 & 0.84 & 0.59 & 0.77 & 0.64 & 0.66 \\ 0.70 & 0.72 & 0.73 & 0.89 & 0.77 & 0.67 & 0.85 & 0.68 & 0.85 & 0.83 \\ 0.81 & 0.83 & 0.62 & 0.75 & 0.83 & 0.80 & 0.75 & 0.68 & 0.60 & 0.76 \\ 0.83 & 0.59 & 0.55 & 0.91 & 0.56 & 0.73 & 0.63 & 0.88 & 0.77 & 0.65 \\ 0.85 & 0.84 & 0.59 & 0.77 & 0.86 & 0.85 & 0.78 & 0.66 & 0.61 & 0.77 \\ 0.80 & 0.73 & 0.61 & 0.90 & 0.89 & 0.66 & 0.78 & 0.91 & 0.84 & 0.57 \\ 0.67 & 0.64 & 0.85 & 0.88 & 0.56 & 0.73 & 0.89 & 0.77 & 0.92 & 0.66 \\ 0.83 & 0.85 & 0.89 & 0.81 & 0.72 & 0.73 & 0.89 & 0.77 & 0.92 & 0.66 \\ 0.84 & 0.89 & 0.77 & 0.88 & 0.57 & 0.77 & 0.82 & 0.79 & 0.81 & 0.65 \\ 0.86 & 0.85 & 0.88 & 0.66 & 0.85 & 0.88 & 0.56 & 0.73 & 0.63 & 0.84 \\ 0.66 & 0.65 & 0.83 & 0.82 & 0.60 & 0.71 & 0.65 & 0.84 & 0.83 & 0.58 \\ 0.65 & 0.71 & 0.81 & 0.56 & 0.77 & 0.86 & 0.85 & 0.71 & 0.66 & 0.61 \\ 0.89 & 0.86 & 0.78 & 0.66 & 0.85 & 0.80 & 0.64 & 0.85 & 0.85 & 0.84 \\ 0.56 & 0.73 & 0.63 & 0.88 & 0.59 & 0.77 & 0.86 & 0.85 & 0.71 & 0.87 \\ 0.90 & 0.67 & 0.82 & 0.68 & 0.61 & 0.80 & 0.89 & 0.88 & 0.78 & 0.66 \end{bmatrix}. \quad (19)$$

To verify the efficiency of HMODPSO algorithm, three different kinds of algorithms were proposed: HMODPSO-1 (adopting NS-1 operation), HMODPSO-2 (adopting NS-2 operation), and HMODPSO-3 (adopting NS-3 operation). At the same time, compare the proposed algorithms with NSGA-II [23], MODPSO [12], and MODPSO-GSA [24] to show their potential competences. In the experiment, population size is 60, 100 iterations are carried out, the external archive size is chosen as 30, the crossover probability of NSGA-II is 0.8, and the mutation rate of NSGA-II is 0.1. All the simulations were performed under the same environment (Matlab) on Intel Core i5-4590 3.3 GHz CPU with 4 GB RAM.

The Pareto fronts produced by six algorithms are shown in Figure 5; it can be seen that all the solutions using HMODPSO are better than the other three algorithms. MODPSO-GSA can also get good solutions. However, MODPSO and NSGA-II easily fall into the local optimum. The different Pareto fronts are compared in Figure 6.

For each algorithm, 30 independent runs were executed. The computational time of algorithm is represented by T. The average results of the Pareto optimal solutions attained by six algorithms are shown in Table 1.

Owing to the time window constraint of the DWTA multiobjective optimization model, the computational time of algorithm must satisfy requirements of the targets' time window. From Table 1, NSGA-II, HMODPSO-2, and HMODPSO-3 algorithms do not meet the requirements. MODPSO has the fastest convergence speed, but it easily falls into the local optimum. Compared with MODPSO-GSA, HMODPSO-1 has better comprehensive performance. So, in the $t = 1$ stage, the assignment $(f = 5.0524,\ g = 13)$ which was obtained once by HMODPSO-1 can be selected for the engagement. Figure 7 shows the damage probability of each target and the scheme of the assignment $(f = 5.0524,\ g = 13)$.

After the first engagement, the parameters $M(t+1)$, $N(t+1)$, $w_j(t+1)$, $p_{ij}(t+1)$, and $t_j^T(t+1)$ need to be updated in the next stage according to the target damage assessment and the new air combat situation. Assume that, in the $t = 2$ stage, $M(2) = 7$ and $N(2) = 4$. The damage probability threshold of each target can be set to 0.9. The time window of each target $t_j^T(2)$ is $[6.5, 8.1, 5.9, 0.2]$; and each target's threat coefficient $w_j(2)$ is $[0.6, 0.8, 0.5, 0.7]$. The weapon's damage probability $p_{ij}(2)$ is

$$P(2) = \begin{bmatrix} 0.65 & 0.74 & 0.85 & 0.81 \\ 0.72 & 0.77 & 0.92 & 0.59 \\ 0.86 & 0.63 & 0.71 & 0.78 \\ 0.53 & 0.80 & 0.73 & 0.65 \\ 0.90 & 0.62 & 0.79 & 0.74 \\ 0.82 & 0.91 & 0.75 & 0.84 \\ 0.56 & 0.65 & 0.73 & 0.94 \end{bmatrix}. \quad (20)$$

Since $t_4^T(2) = 0.2\,\mathrm{s}$ is so small that it is difficult to satisfy the time window of the special target, a new priority selecting method is efficiently adopted to deal with several special targets whose time windows do not satisfy the time window constraint of the proposed algorithm. From the existing weapons at a certain stage, select the missile with the maximum damage probability to attack the special target; repeat this operation until the special target's joint damage probability satisfies the threshold. As seen from $P(2)$, the 7-th missile attacking the target can satisfy the threshold of damage probability. The assignment of the remaining targets can be solved by HMODPSO-1 algorithm. Figure 8 shows the Pareto front.

HMODPSO-1 was independently run for 30 times. The average results of the Pareto optimal solutions attained by HMODPSO-1 are shown in Table 2.

In the $t = 2$ stage, the assignment $(f = 1.7280,\ g = 3)$ can be selected for the engagement. Figure 9 shows the damage probability of each target and the scheme of the assignment $(f = 1.7280,\ g = 3)$. HMODPSO-1 can efficiently solve the

```
(1)  Initialize the swarm with $H$ particles
(2)  Initialize the external archive
(3)  for $k = 1 : 1 : \max It$
(4)      for $j = 1 : 1 : H$
(5)          Select leader for each particle
(6)          Update velocity (Eq. (9))
(7)          Update position (Eq. (10))
(8)          CheckBoundaries (Eq. (12))
(9)          Repairing operator
(10)           Evaluate the particle
(11)           Update Pbest
(12)           Cauchy mutation operator (Eq. (15))
(13)           CheckBoundaries (Eq. (17))
(14)           Update Pbest
(15)           Update external archive
(16)           Neighborhood searching operator
(17)     end for
(18)     if archive_size > max_archive
(19)         Maintain the external archive by crowding distance sorting strategy
(20)     end if
(21) end for
(22) Return the external archive
```

Algorithm 3: The pseudocode of the HMODPSO algorithm.

Table 1: The average results of the Pareto optimal solutions.

Pareto solutions	NSGA-II	MODPSO	MODPSO-GSA	HMODPSO-1	HMODPSO-2	HMODPSO-3
13	*	*	5.0500	5.0524	5.0524	5.0524
14	*	*	5.0779	5.0998	5.1052	5.1054
15	5.0551	5.0550	5.1516	5.1510	5.1536	5.1576
16	5.1220	5.0635	5.1971	5.1975	5.1976	5.1980
17	5.1346	5.1306	5.2380	5.2372	5.2382	5.2291
18	5.2158	5.2031	5.2682	5.2639	5.2680	5.2682
19	5.2375	5.2283	5.2894	5.2876	5.2899	5.2904
20	5.2637	5.2614	5.3038	5.3095	5.3135	5.3183
T/s	56.23	3.1364	6.8735	4.4328	8.7426	12.5279

Table 2: The average results of the Pareto optimal solutions.

Pareto solutions	HMODPSO-1
3	1.7280
4	1.7856
5	1.8372
6	1.8712
T/s	1.1155

DWTA until no targets or weapons are left in the cooperative air combat.

5. Conclusions

This paper has presented a new hybrid multiobjective DPSO called HMODPSO algorithm to solve the cooperative air combat DWTA multiobjective optimization problems. The proposed algorithm has three advantages. (a) The leader particle selecting operator and neighborhood searching operator can improve the search ability and the rate of convergence. (b) The repairing operator can enhance the efficiency of generating feasible solutions. (c) The Cauchy mutation operator can boost the diversity and distribution of Pareto optimal solutions. HMODPSO can find solutions with good accuracy, convergence, and diversity, which can generate obviously better DWTA decisions without the cost of overmuch extra computation time compared to NSGA-II, MODPSO, MODPSO-GSA, and MOEA/D. HMODPSO can also improve the cooperative air combat effectiveness efficiently.

Future research will focus on optimizing DWTA instances under larger scales. In addition, more new mechanisms on reducing the time complexity of the proposed algorithm will also be investigated.

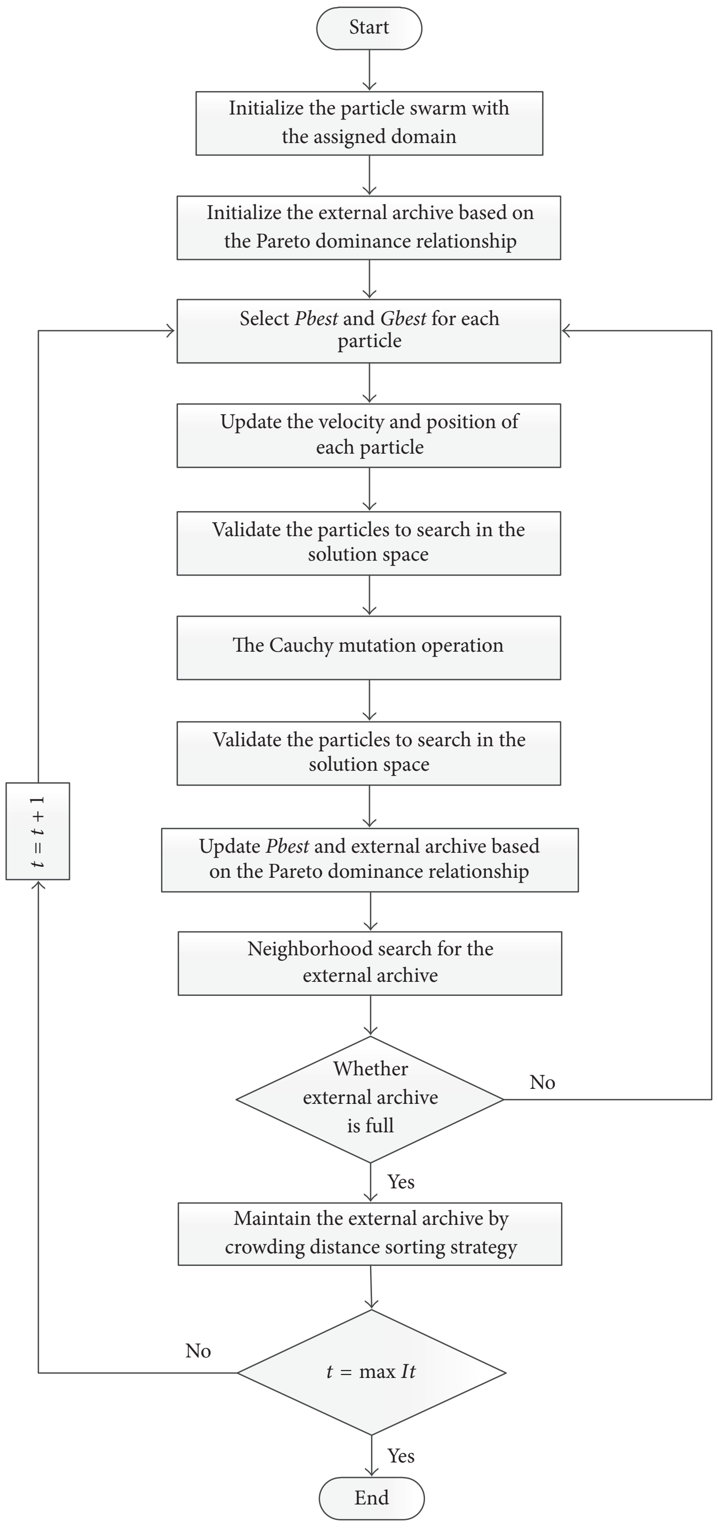
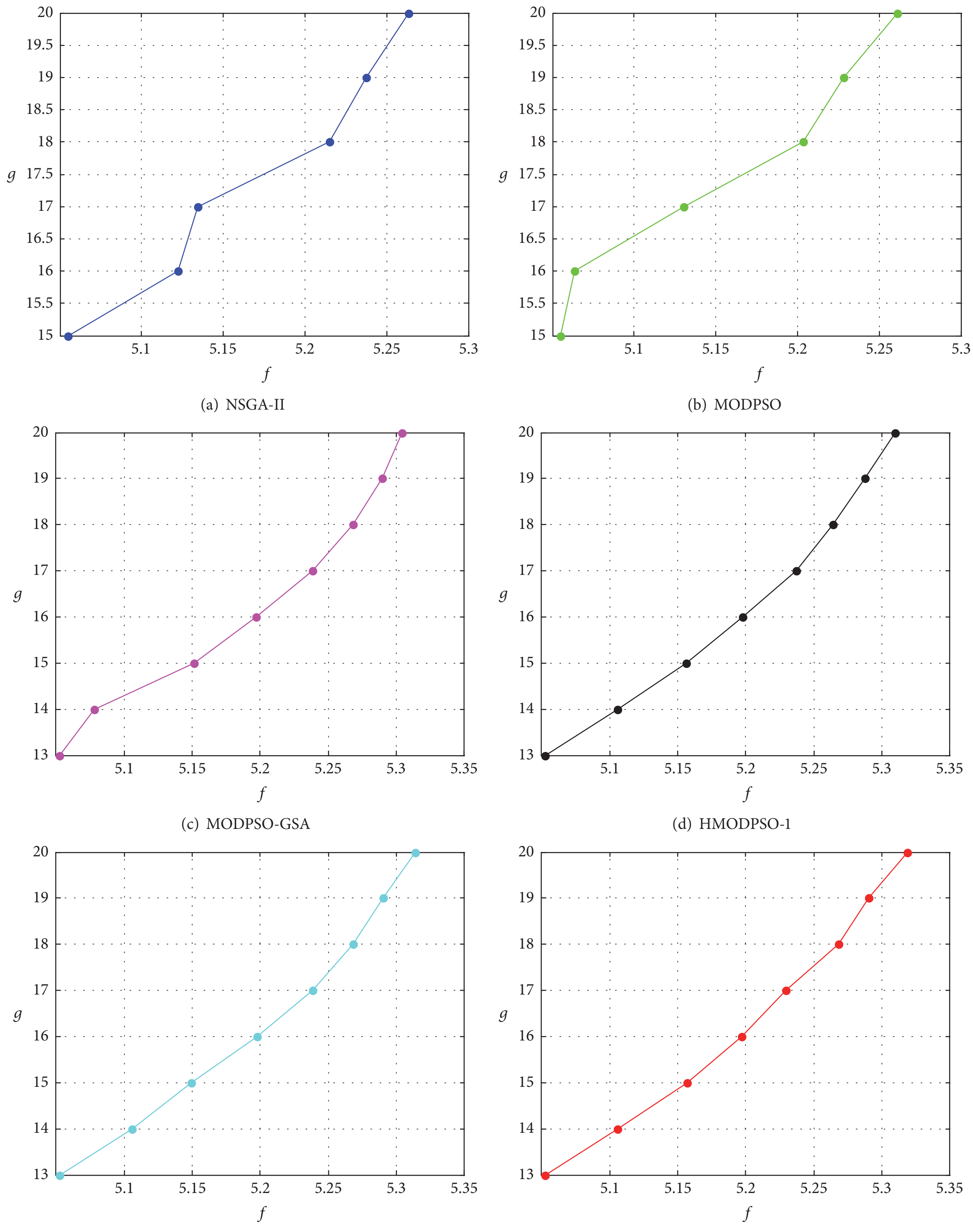

Start
Initialize the particle swarm with the assigned domain
Initialize the external archive based on the Pareto dominance relationship
Select Pbest and Gbest for each particle
Update the velocity and position of each particle
Validate the particles to search in the solution space
The Cauchy mutation operation
Validate the particles to search in the solution space
Update Pbest and external archive based on the Pareto dominance relationship
Neighborhood search for the external archive
Whether external archive is full
No
Yes
Maintain the external archive by crowding distance sorting strategy
t = max It
No
Yes
t = t + 1
End

Figure 4: The entire flowchart of HMODPSO.



(a) NSGA-II (b) MODPSO (c) MODPSO-GSA (d) HMODPSO-1 (e) HMODPSO-2 (f) HMODPSO-3

Figure 5: The Pareto fronts obtained by six algorithms.

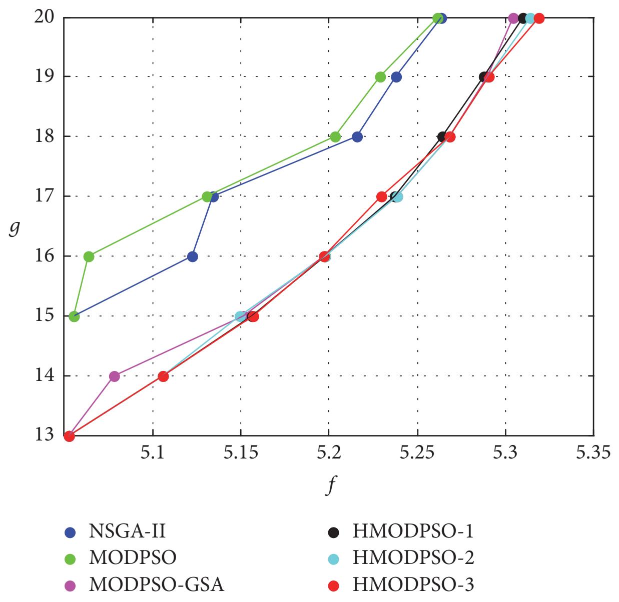


Figure 6: The compared Pareto fronts.

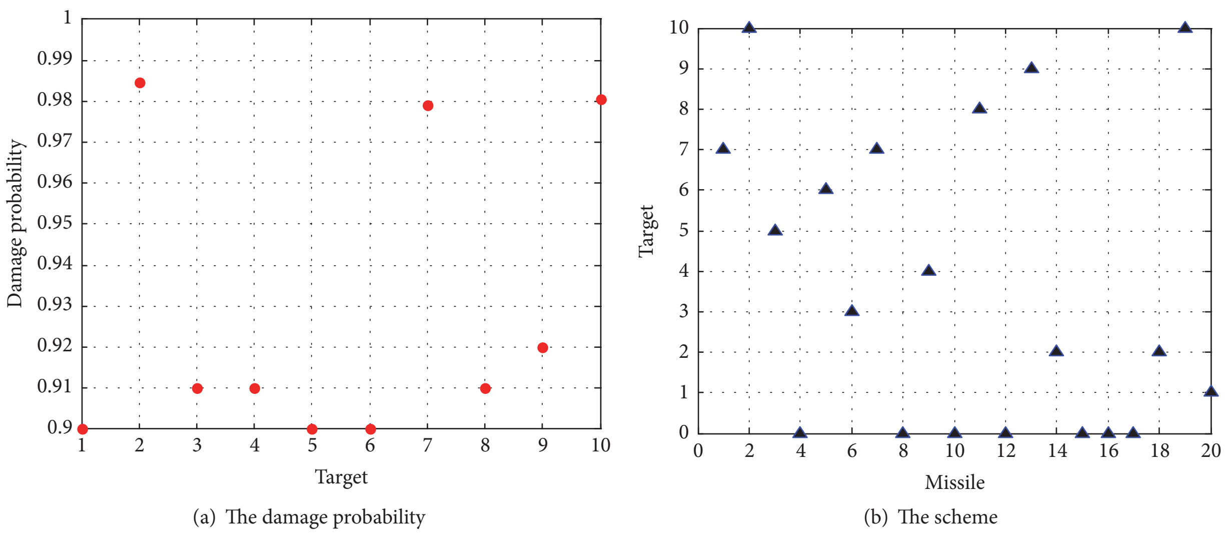


Figure 7: The WTA scheme obtained by HMODPSO-1.

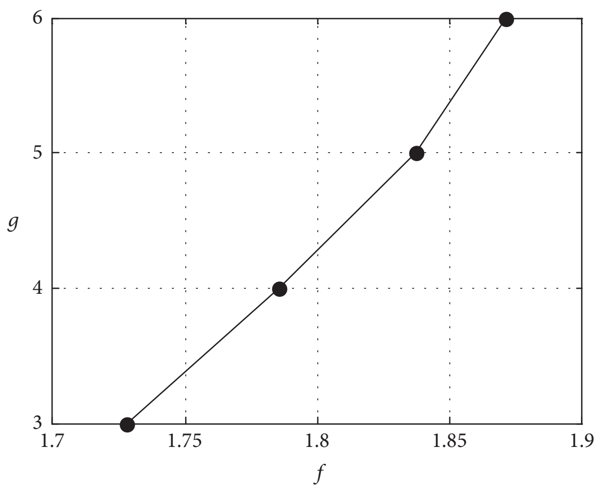
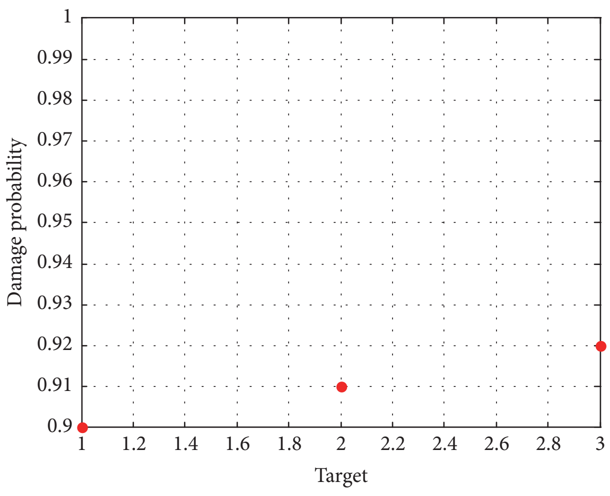
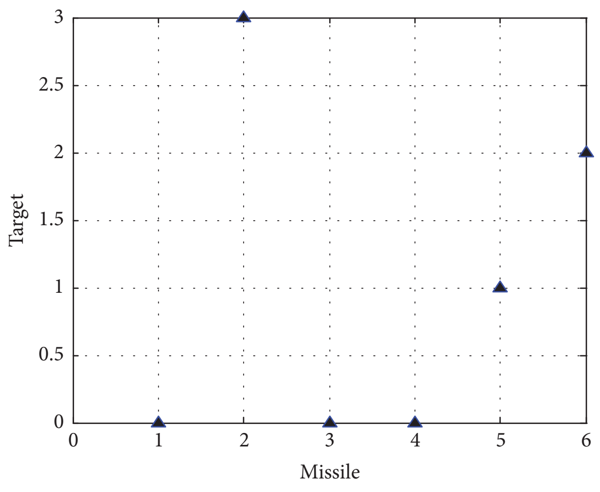


Figure 8: The Pareto front obtained by HMODPSO-1.



(a) The damage probability



(b) The scheme

Figure 9: The WTA scheme obtained by HMODPSO-1.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] S. P. Lloyd and H. S. Witsenhausen, "Weapons allocation is NP-complete," in *Proceedings of the IEEE Summer Simulation Conference*, pp. 1054–1058, Reno, Nev, USA, 1986.

[2] P. A. Hosein and M. Athans, "Preferential defense strategies. Part I: the static case," Tech. Rep. LIPS-P-2002, 1900.

[3] P. A. Hosein and M. Athans, "Preferential defense strategies. Part II: the dynamic case," Tech. Rep. LIPS-P-2003, 1900.

[4] H. Cai, J. Liu, Y. Chen, and H. Wang, "Survey of the research on dynamic weapon-target assignment problem," *Journal of Systems Engineering and Electronics*, vol. 17, no. 3, pp. 559–565, 2006.

[5] D. Khosla, "Hybrid genetic approach for the dynamic weapon-target allocation problem," in *Battlespace Digitization and Network-Centric Warfare*, vol. 4396 of *Proceedings of SPIE*, pp. 244–259, August 2001.

[6] C. B. Liu, H. Y. Wang, and Z. M. Qiu, "An adaptive Memetic algorithm solving dynamic weapon target assignment problem," in *Proceedings of the in 2nd International Conference on Information Engineering and Computer Science*, pp. 1–4, Wuhan, China, 2010.

[7] J. Chen, B. Xin, Z. H. Peng, L. H. Dou, and J. Zhang, "Evolutionary decision-makings for the dynamic weapon-target assignment problem," *Science in China Series F: Information Sciences*, vol. 52, no. 11, pp. 2006–2018, 2009.

[8] B. Xin, J. Chen, J. Zhang, L. H. Dou, and Z. H. Peng, "Efficient decision-makings for dynamic weapon-target assignment by virtual permutation and tabu search heuristics," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 40, no. 6, pp. 649–662, 2010.

[9] B. Xin, J. Chen, Z. Peng, L. Dou, and J. Zhang, "An efficient rule-based constructive heuristic to solve dynamic weapon-target assignment problem," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 41, no. 3, pp. 598–606, 2011.

[10] Y. Wang, J. Li, W. Huang, and T. Wen, "Dynamic weapon target assignment based on intuitionistic fuzzy entropy of discrete particle swarm," *China Communications*, vol. 14, no. 1, pp. 169–179, 2017.

[11] Y. Wang, W. G. Zhang, and Y. Li, "An efficient clonal selection algorithm to solve dynamic weapon-target assignment game model in UAV cooperative aerial combat," in *Proceedings of the 35th Chinese Control Conference (CCC '16)*, pp. 9578–9581, IEEE, Chengdu, China, July 2016.

[12] X. Liu, Z. Liu, W.-S. Hou, and J.-H. Xu, "Improved MOPSO algorithm for multi-objective programming model of weapon-target assignment," *Systems Engineering and Electronics*, vol. 35, no. 2, pp. 326–330, 2013.

[13] D. Zhou, X. Li, Q. Pan, K. Zhang, and L. Zeng, "Multiobjective weapon-target assignment problem by two-stage evolutionary multiobjective particle swarm optimization," in *Proceedings of the IEEE International Conference on Information and Automation (ICIA '16)*, pp. 921–926, Ningbo, China, August 2016.

[14] D. P. Lötter and J. H. Van Vuuren, "A tri-objective, dynamic weapon assignment model for surface-based air defence," *ORiON*, vol. 32, no. 1, pp. 1–22, 2016.

[15] J. Li, J. Chen, B. Xin, and L. Dou, "Solving multi-objective multi-stage weapon target assignment problem via adaptive NSGAII and adaptive MOEA/D: a comparison study," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '15)*, pp. 3132–3139, Sendai, Japan, May 2015.

[16] J. Li, J. Chen, B. Xin, L. H. Dou, and Z. H. Peng, "Solving the uncertain multi-objective multi-stage weapon target assignment problem via MOEA/D-AWA," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 4934–4941, Vancouver, Canada, 2016.

[17] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, 1995.

[18] M.-F. Leung, S.-C. Ng, C.-C. Cheung, and A. K. Lui, "A new strategy for finding good local guides in MOPSO," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 1990–1997, IEEE, Beijing, China, July 2014.

[19] J. Yan, X. M. Li, L. J. Liu, and F. X. Zhang, "Weapon-target assignment based on Memetic optimization algorithm in beyond-visual-range cooperative air combat," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 40, no. 10, pp. 1424–1429, 2014.

[20] G. Peng, Y. W. Fang, D. Chai, Y. Xu, and W. S. Peng, "Multiobjective particle swarm optimization algorithm based on sharing-learning and Cauchy mutation," in *Proceedings of the 35th Chinese Control Conference*, pp. 9155–9160, Chengdu, China, 2016.

[21] Y.-C. Wang, G.-L. Shan, and J. Tong, "Solving sensor-target assignment problem based on cooperative memetic PSO algorithm," *Systems Engineering and Electronics*, vol. 35, no. 5, pp. 1000–1007, 2013.

[22] C. R. Raquel and P. C. Naval, "An effective use of crowding distance in multiob-jective particle swarm optimization," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pp. 257–264, Washington, DC, USA, 2005.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[24] J. Gu, J. Zhao, J. Yan, and X. Chen, "Cooperative weapon-target assignment based on multi-objective discrete particle swarm optimization-gravitational search algorithm in air combat," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 41, no. 2, pp. 252–258, 2015.

Hindawi
Journal of Optimization
Volume 2017, Article ID 8936164, 7 pages
https://doi.org/10.1155/2017/8936164

Research Article

An Improved Heuristic Algorithm for UCAV Path Planning

Kun Zhang,[1] Peipei Liu,[1] Weiren Kong,[1] Jie Zou,[2] and Min Liu[2]

[1] *School of Electronics and Information, Northwestern Polytechnical University, Xian, Shaanxi 710072, China*
[2] *Science and Technology on Electro-Optic Control Laboratory, Luoyang, Henan 471009, China*

Correspondence should be addressed to Kun Zhang; kunnpu@gmail.com

Received 29 December 2016; Accepted 23 February 2017; Published 10 April 2017

Academic Editor: Maoguo Gong

The study of unmanned combat aerial vehicle (UCAV) path planning is increasingly important in military and civil field. This paper presents a new mathematical model and an improved heuristic algorithm based on Sparse A^* Search (SAS) for UCAV path planning problem. In this paper, flight constrained conditions will be considered to meet the flight restrictions and task demands. With three simulations, the impacts of the model on the algorithms will be investigated, and the effectiveness and the advantages of the model and algorithm will be validated.

1. Introduction

Nowadays, unmanned combat aerial vehicle (UCAV) has long been a challenging area for researchers in military and civil field. Path planning is defined as looking for the optimal path of moving objects from the start point to the target point under specific constraints (including environmental constraint and movement constraint) [1]. The path planning aims to find the path with the highest survival rate, lower loss, and shorter period of time. Currently, the path planning problem has been widely used in different areas such as cruise missile, helicopter, and UCAV. In modern warfare, with the development of various air defense technologies, there is no doubt that the path planning problem is more and more being paid attention to in military field. Numerous scholars study the path planning problem constantly. The represented techniques of UCAV path planning are like PSO [1, 2], dynamic planning [3], A^* algorithm [4, 5], ant colony algorithm [6], genetic algorithm [7, 8], and so on [9–13]. Reference [14] discussed sparse algorithm, another effective way, which greatly improves the efficiency of the search, but it is easy to fall into a death cycle under the conditions of lacking maneuver ability. Reference [5] adopted sparse algorithm, but angle heuristic function is not taken into consideration and the number of constraint conditions is small; the simulation result is not scientific. Reference [7, 15] proposed the algorithms with large amount of calculation, making the algorithms unsuitable to seek optimization solution. To address these problems, an improved heuristic algorithm [16] is studied which takes angle information into account in a certain range, two methods of trajectory smooth straightening processing are adopted and compared, and the corresponding simulations are given in this paper.

2. Related Works

2.1. Basic Mathematical Model. The path planning problem of UCAV can be modeled as a constrained optimization problem. Before searching track, flight condition and elements (like terrain, threats, climate, etc.) of relevant path planning are represented as symbol information.

Let (x_L, y_L, z_L) be longitude, latitude, and height of a certain point in state space. The path planning space can be represented as a set: $\{(x_L, y_L, z_L) \mid 0 \leqslant x_L \leqslant \max X_L, 0 \leqslant y_L \leqslant \max Y_L, 0 \leqslant z_L \leqslant \max Z_L\}$, which represents a space district. In practical planning, the planning space is divided into two-dimensional grids or three-dimensional grids; a series of nodes are acquired and built into a network graph, as shown in Figure 1. The path planning problem can be simply attributed to a combinational optimization problem for getting the shortest path of the network graph. That is to say, when UCAV is flying along the path formed by some

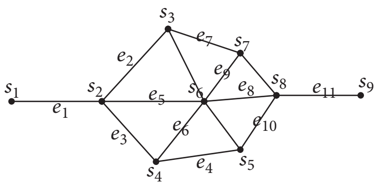


FIGURE 1: Network graph.

nodes of the network graph, a certain kind of path takes minimum cost.

Supposing the nodes of network graph form a set S

$$S = \{s_1, s_2, s_3, \ldots, s_m\}. \quad (1)$$

Define a set that includes all paths from the start point to the end point as E:

$$E = \{e_1, e_2, e_3, \ldots, e_n\}. \quad (2)$$

Let s_i and s_j be two adjacent nodes on the path e_k, the connecting line between two nodes can be expressed by $V(s_i, s_j)$, the cost value of the connecting line between two nodes can be expressed by u_{ij}, and the path planning problems of UCAV are defined as follows:

$$\begin{aligned} \min \quad & f(e_k) = \sum_{(s_i,s_j)\in e_k} u_{ij} \\ \text{s.t.} \quad & e_k \in E,\ s_i \in S,\ s_j \in S. \end{aligned} \quad (3)$$

As can be seen from the above content, the performance constraint of UCAV is not reflected in the planning. If the nodes in the network graph are feasible points which take performance constraint of UCAV into account, the path with the performance constraint of UCAV can be reflected from solving the above optimization problems. This is a new mathematical model. Compared with [5], our new mathematical model takes more constraints into account. And simulation result shows that it is highly useful for the approximate optimal solution. Besides, it is also good at processing path planning in complicated conditions.

2.2. Basic Constraint Conditions of Path Planning. There are many factors that influence the result of path planning. These factors, which include terrain features, threat locations, and mission requirements, are basic constraints in mathematical modeling. Path planning should meet basic constraints, and they mainly include the following constraints [4].

Minimum Route Length. Aircraft generally does not want to weave and turn constantly, because this adds to fuel cost and increases navigational errors.

Maximum Turning Angle. The turning angle of the aircraft does not exceed maximum turning angle. For instance, the aircraft cannot make severe turns without a greater risk of collision in formation fight.

Route Distance Constraint. The length of the route does not exceed maximum distance because of fuel restriction.

Specific Approaching Angle to Target Point. This constrains UCAV to approach the hostile aircraft from a predetermined angle to ensure UCAV defend the weak part.

2.3. Path Planning Cost. On the premise that some constraints are met, the UCAV path planning aims to generate trajectory with the highest survival rate. Therefore, threat locations in battle field should be fully taken into account. Threat factors and fuel restriction are mainly taken into account when calculating trajectory cost.

2.3.1. Models of Threats

Threat Model of Radar. The factors that influence the probability of radar detection mainly include earth curvature, atmospheric refraction and absorption, ground clutter interference, distance between aircraft and radar, radar cross-section, radar performance, and ground multipath effect. For the sake of simplification, here we mainly take the distance from aircraft to radar and radar performance into account. Supposing the flying height is h, the horizontal distance from aircraft to radar is R, radar maximal horizontal range is $R_{\max}$, radar performance coefficient is k, the probabilistic model of radar detection can be presented as follows:

$$P_R = e^{-kR^2/h - R^4/R_{\max}^4}. \quad (4)$$

If $a > 0$, the approximation expression e^{-a} can be equivalent as follows:

$$e^{-a} \approx \frac{1}{a+1}. \quad (5)$$

If $R < R_{\max}$, aircraft is in the detection range of radar, and threat model can be simplified as follows:

$$P_R = e^{-R^4/R_{\max}^4}. \quad (6)$$

Use formula (5) to get the approximation expression

$$P_R \approx \frac{R_{\max}^4}{R^4 + R_{\max}^4}. \quad (7)$$

If aircraft is out of the radar maximal horizontal range, threat for radar to aircraft is zero. Threat model of radar can be represented as follows:

$$P_R = \begin{cases} \dfrac{R_{\max}^4}{R^4 + R_{\max}^4} & R \leqslant R_{\max} \\ 0 & R > R_{\max}. \end{cases} \quad (8)$$

Threat Model of Surface-to-Air Missile. Supposing probability distribution of missile to target obeys Poisson distribution concerning horizontal distance R_m, killing probability of maximum attack radius $R_{M\max}$ is e^{-1}, and if $R_m < R_{M\max}$, threat model for missile to aircraft can be presented as follows:

$$P_M = e^{-R_m/R_{M\max}}. \quad (9)$$

According to formula (5), the approximation expression can be equivalent as follows:

$$P_M = \frac{R_{M\max}}{R_m + R_{M\max}}. \quad (10)$$

If aircraft is out of the maximum attack radius range, the threat for missile to aircraft is zero. Threat model of missile can be presented as follows:

$$P_R = \begin{cases} \dfrac{R_{M\max}}{R_m + R_{M\max}} & R_m \leqslant R_{M\max} \\ 0 & R_m > R_{M\max}. \end{cases} \quad (11)$$

Threat Model of Terrain. The main danger caused by terrain is peaks, which may be an obstacle of flight when aircraft is flying at the fixed height. And peaks can be expressed as cones. Supposing the horizontal cross-section of the peak is a circumference at the flight height and the radius of circumference is R_d, the central position of the peak is (x, y), and the current position of aircraft is (x_A, y_A), the horizontal distance R_{AT} from aircraft to the central of the peak can be expressed as follows:

$$R_{AT} = \sqrt{(x_A - x)^2 + (y_A - y)^2}. \quad (12)$$

Unlike several kinds of threats above, the collisions between the aircraft and peak are fatal risk for aircraft. Therefore, enough space and time must be maintained when the aircraft is bypassing the peak. Threat model of peaks can be presented as follows:

$$P_T = \begin{cases} 0 & R_{AT} > 10\,\text{Km} + R_d \\ 1 & R_{AT} \leqslant 2\,\text{Km} + R_d \\ \dfrac{1}{R_{AT}} & 2\,\text{Km} + R_d < R_{AT} < 10\,\text{Km} + R_d. \end{cases} \quad (13)$$

Threat Model of Atrocious Weather. Similar to the threat of peak to aircraft, supposing that the radius of atrocious weather on the cross-section is R_c and horizontal distance from aircraft to the center of atrocious weather is R_{AW}, mathematical model of atrocious weather can be presented as follows:

$$P_W = \begin{cases} 0 & R_{AW} > 5\,\text{Km} + R_c \\ \dfrac{1}{R_{AW}} & R_{AW} \leqslant 5\,\text{Km} + R_c. \end{cases} \quad (14)$$

2.3.2. Path Planning Cost Calculation Function. In this paper, the cost calculation function relates to threat cost and fuel cost. Because fuel cost is proportional to the voyage, the cost calculation function can be given as

$$J = \sum_{i=1}^{n} \left(w_s l_i + w_t f_{TAi}\right),$$
$$J_i = w_s l_i + w_t f_{TAi}, \quad (15)$$

where J is the total cost of the route, J_i is the ith route cost, l_i is the ith route length, it can decrease flight time of aircraft in enemy air defenses area by cutting down the total length of route, f_{TAi} is the threat index of the ith route, it ensures the aircraft can fly along the safe area, and w_s and w_t are the weight coefficients of distance factors and threat factors, and $w_s + w_t = 1$. Besides, they can be determined according to the specific requirements of route performance. For example, sometimes we need the minimum route, and sometimes the highest survival rate is required.

The calculation of the threat index on the ith route needs to be integrated along the ith route, for the sake of simplification, calculate average value of the threat index of certain points on the route, and then multiply the length of the influenced part. Divide route into m equal parts, select m break points on the route, and their positions can be expressed as

$$\frac{1}{2m} l_i, \frac{3}{2m} l_i, \ldots, \frac{2n+1}{2m} l_i \quad (n = 0, 1, 2, \ldots, \; n < m). \quad (16)$$

The threat index on the ith route is given by the expression

$$f_{TAi} = \frac{1}{m} l_i \sum_{j=1}^{N_t} \left\{ f_{TAj}\left(\frac{1}{2m} l_i\right) + f_{TAj}\left(\frac{3}{2m} l_i\right) + \cdots + f_{TAj}\left(\frac{2n+1}{2m} l_i\right) \right\}, \quad (17)$$

where N_t is the number of the known threat sources, l_i is the ith route length, f_{TAj} is the threat value of the break points, the value of m can be adjusted according to the computational accuracy and the route length. According to different threat classifications, it can be calculated by formulas (8); (11); (13); (14).

2.4. Sparse A^* Algorithm. Heuristic search [17] aims to use heuristic information to find the optimal path with the minimum cost. The main difference between heuristic search and other methods is that the cost information is related to heuristic information. The cost calculation function of heuristic search is given as

$$f(x) = J(x) + h(x), \quad (18)$$

where x is the extended node, $J(x)$ is the actual cost from the start node to the extended node, and $h(x)$ is the estimate cost from the extended node to the target node. According to different task demands, heuristic information may involve many factors such as ground threat sources, artificial obstacles, flight time, and fuel quantity. An appropriate heuristic function can greatly increase search speed and acquire the solution easily.

The primary problem to be solved is how to acquire the candidate node set in the search process. The expression of nodes in state space can be divided into two general types: nodes of graphic expression and nodes of grid expression. The former extends nodes out in the form of ray; the latter divides state space into grids with a certain size and then extends the adjacent grid points out. For example, basic A^* is based on

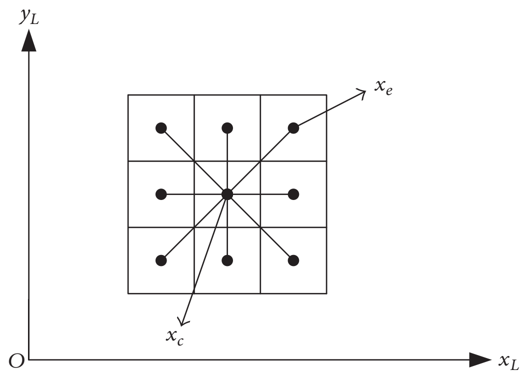


FIGURE 2: The basic extension of A^* nodes.

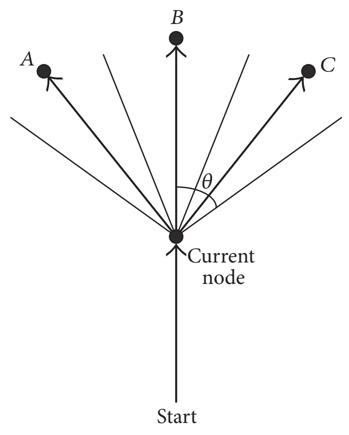


FIGURE 3: SAS extension of nodes.

grid search. Nodes can be divided into three states in heuristic search:

(1) Nodes have been extended.

(2) Nodes have been generated but has not been extended.

(3) Nodes have not been generated.

The first kind of nodes is called closed nodes, and we can construct a table named *CLOSED* to store this kind of nodes; the second kind of nodes is called open nodes because the nodes has been generated but has not been extended, such nodes can be stored in a table named *OPEN*. The start nodes are stored in *OPEN* table in the initial search. New nodes can be generated according to different extensible rules, and then these new nodes are inserted into *OPEN* table according to the increment of the cost value f, the node with the smallest value can be preferentially extended, and it can be stored in *CLOSED* table.

3. Path Planning Methods Based on Sparse A^* Searching Algorithm

3.1. Sparse A^* Algorithm Based on Constrains of Path Planning. Szczerba et al. provided an advanced version of the basic A^* searching algorithm in 2000 which is called SAS (Sparse A^* Search). Traditional A^* algorithm is designed on the basis of grid search. For example, in Figure 2, define x_c as the current node and x_e as the extended node. Eight neighborhood subnodes are taken into account when extending nodes. In addition, we can also take more neighborhood nodes into account when extending nodes. But, generally speaking, larger neighborhood corresponds to the more elaborate route, larger memory space, and longer period of convergence time. Since the nodes can only be extended to eight fixed directions, the generated path may not meet constraints of path planning introduced in Section 2.2.

SAS [5] could combine constraints such as minimum step, maximum turning angle, and maximum route into search algorithm, which condenses search space effectively. SAS not only improves the search efficiency but also meets the flight constrained conditions. In the following section, we will introduce how to combine flight constrained conditions into the process of the extension of nodes. We will also discuss how to set a reasonable cost function.

3.2. Extensible Rules of Nodes. Let L be the minimum step and θ be the maximum turning angle. Based on the known direction of current nodes, the search of the current nodes is limited to the fan-shaped region with a range of 2θ. The medial axis of the fan-shaped region is the direction of the current nodes. Then, the fan-shaped region is divided into m equal parts, and the cost of knot vector whose distance is between each subpart and the current nodes is calculated. To save space and speed up convergence rate, only the node with the minimum cost of each sector is reserved. As shown in Figure 3, the fan-shaped region is divided into three equal parts: A, B, and C representing the nodes with the minimum cost of each fan-shaped region, respectively. Set m according to different accuracy requirements and convergence rate, and the number of nodes should be smaller than m. This process is repeated when extending nodes. Remove the node with minimum cost from *OPEN* table, and then define this node as the current node and insert the current node into *CLOSED* table. Extend the current node, and then insert all subnodes into *OPEN* table according to the increment of cost value.

The constraint of route distance is the allowable maximum length of the route which represents the payload of fuel and arrival time constraint in a specific task. The route whose length is longer than maximum distance ($d_{\max}$) is termed as the invalid route. On the basis of the above discussion, before inserting the nodes with the minimum cost into *OPEN* table, define x_c as the current node, and we need to make a judgment of another condition as follows:

$$D(x_c) + SL(x_c) \leqslant d_{\max}, \tag{19}$$

where $D(x_c)$ is the actual distance between the start point and the current point, $SL(x_c)$ is the straight-line distance between current node and the target point, $d_{\max}$ is generally several times the value of the straight-line distance between the start point to the target point. If the current node meets the above requirements, it can be extended; otherwise it will be discarded. This method can not only meet flight restrictions but also generate direct route effectively.

3.3. Trajectory Cost Function. On the basis of the above discussion, the cost function of heuristic search can be represented as $f(x) = J(x) + h(x)$. The actual cost function $J(x)$ can be calculated by formula (15), and here we discuss the calculation of $h(x)$.

The basic A^* heuristic function adopts the Manhattan distance to show the distance from the current node to the target node:

$$h_1 = |x_b - x_m| + |y_b - y_m|, \tag{20}$$

where (x_b, y_b) is the coordinate of the current node and (x_m, y_m) is the coordinate of the target node. Considering the constraints of target approaching angle, aircraft is set to reach the target in a fixed direction. Angle information is taken into account when designing the cost function. The deviation between current course and target course is used to guide search process within the predetermined direction. The cost function [1] which adds the angle information can be expressed as follows:

$$h_2 = w_d h_1 + w_\theta h(\theta), \tag{21}$$

$$h(\theta) = \sqrt{\Delta\theta^2}, \tag{22}$$

where h_1 is distance heuristic function which can be calculated by formula (20), $h(\theta)$ is angle heuristic function which can be calculated by formula (22), w_d and w_θ are the weight coefficients of distance heuristic and angle heuristic, respectively, and $w_d + w_\theta = 1$. In formula (22), $\Delta\theta = \theta_x - \theta_m$, θ_x is the direction of the line which connects current nodes and target nodes and θ_m is the predetermined target approaching angle. Supposing $\theta_x \in (-\pi, +\pi]$, $\theta_m \in (-\pi, +\pi]$, and $\Delta\theta \in (-2\pi, +2\pi]$, $\Delta\theta$ is regulated as follows:

$$\Delta\theta = \begin{cases} -(2\pi - \Delta\theta) & \pi < \Delta\theta \leqslant 2\pi \\ \Delta\theta + 2\pi & -2\pi < \Delta\theta \leqslant -\pi. \end{cases} \tag{23}$$

In traditional search method, the path may prematurely tend to the predetermined target direction so that the resulting path may not be the shortest path. Only when the aircraft nears the target can the aircraft turn to the predetermined target direction in the long distance flight, it is not necessary to have been proceeding angle heuristic throughout the whole search process. In the proposed method, heuristic function h_2 takes angle information into consideration. Let radius be $a*d$, d be the simulation step size, and a be range coefficient. The size of the radius can be adjusted according to the maneuver ability and the simulation step size. The aircraft should be able to turn the predetermined direction timely within this range. If the nodes are within this range, heuristic function h_2 is used to calculate the cost; otherwise heuristic function h_1 is used. The proposed method can not only make the aircraft close to the target in the predetermined direction but also get an approximate optimal solution. In SAS algorithm, the calculation of route cost can be expressed as follows:

$$f(x) = \begin{cases} J(x) + h_1 & DL(x) > a * d \\ J(x) + h_2 & DL(x) \leqslant a * d, \end{cases} \tag{24}$$

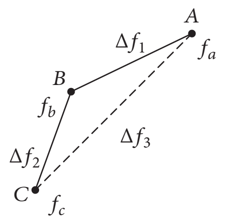


Figure 4: Track smooth straightening processing map.

where $DL(x)$ is the distance between x and target node and $J(x)$ is the actual cost value. The cost value generated by the distance and threat can be calculated by formula (15).

On the basis of the above discussion, it is very important to find a reasonable heuristic function for getting an approximate optimal solution without reducing search speed. It is also important to design corresponding appropriate heuristic function for different stages of track search.

3.4. Trajectory Smooth Straightening Processing

3.4.1. Method 1. From the above description, the ultimate goal of path planning system is to generate a set of track point data and then provide these data to flight task manager. Therefore, the initial route obtained by search algorithm needs to be processed so as to get the smaller number of track points. Track points located between start point and target point are stored in path table. Set start point as current point, and traverse other nodes in path table according to the order of the current point to next point. Check whether the connection line of current node and a certain visiting node will encounter threat. If the connection line of current node and a certain visiting node encounters threat, go back to the previous node, set the previous node as current node, delete all nodes between current node and the last current node, update information of current note, and retraverse from this current node until reaching the target node. Otherwise it continues to traverse and repeat above steps.

3.4.2. Method 2. The following method is optimized in the search process. As shown in Figure 4, let A be current node, B be father node of current node, C be ancestral node, and their cost values are f_a, f_b, and f_c, respectively. In the planning of extending current node from *OPEN* table to *CLOSED* table, calculate the route cost from current node A to ancestral node C, and the route cost is $\Delta f_3 = f_a - f_c$; the route cost from node A to its father node B is $\Delta f_1 = f_a - f_b$; the route cost from father node B to ancestral node C is $\Delta f_2 = f_b - f_c$. Judge their relationship, if $\Delta f_3 < \Delta f_1 + \Delta f_2$, which indicated that the connection cost from current node to its ancestral node is superior to that from current node to its father node. Therefore, set ancestral node of current node as its father node and adjust its cost value.

The route obtained from the above methods can not only reduce route cost but also constrain the number of turns. Besides, valid information of track points can be generated which is in favor of the future navigation.

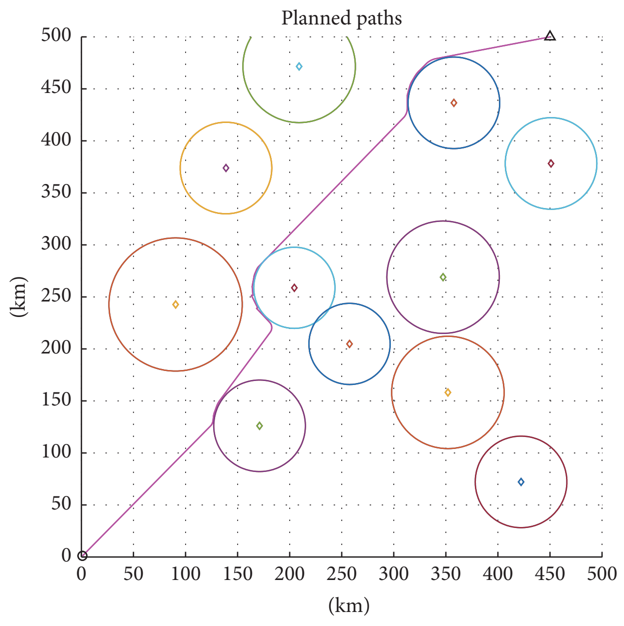


Figure 5: The first result.

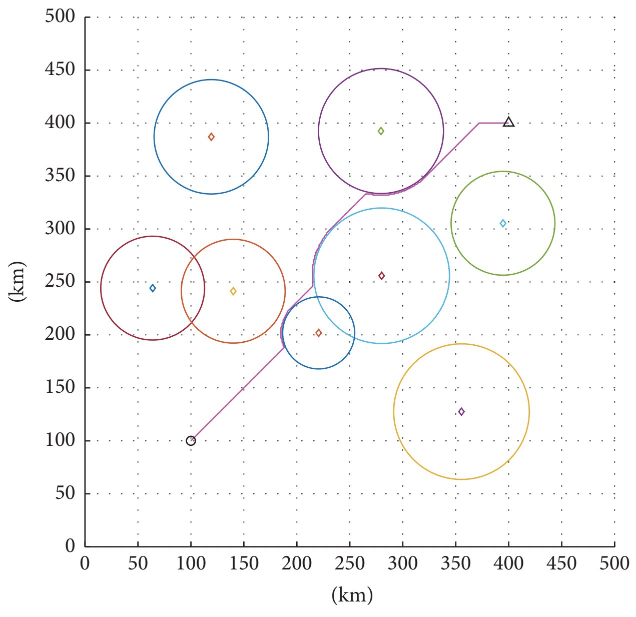


Figure 6: The second result.

4. Experimental Study

Define the range of path planning as 500 Km × 500 Km and the simulation step size and minimum route length d as 5 Km. Define distance coefficient and threat coefficient as $w_s = 0.5$ and $w_t = 0.5$, respectively, when calculating the actual cost; define distance coefficient and angle coefficient as $w_d = 0.5$ and $w_\theta = 0.5$, respectively, when calculating heuristic function which takes the angle information into account. The maximum route distance constraint is 1.5 times the straight distance between the start point and target point. Let angle heuristic range coefficient a be 5, maximum turning angle be 60, m be 3 when extending nodes.

(1) Supposing the coordinate of start point is $(0, 0)$, the coordinate of target point is $(450, 500)$. The threat distribution is given as Figure 5, and the result of path planning is presented in Figure 5.

(2) Supposing the coordinate of start point is $(100, 100)$, the coordinate of target point is $(400, 400)$. The threat distribution and the result of path planning are shown in Figure 6. The second method (see Section 3.4.2) is taken when considering route processing, and the result of path planning can be shown as Figure 7.

(3) Supposing the coordinate of start point is $(40, 30)$, the coordinate of target point is $(100, 100)$. The threat distribution is given as Figure 8, and the result of path planning can be shown as Figure 8. The first method (see Section 3.4.1) is taken when considering route processing, and the result of path planning can be shown as Figure 8.

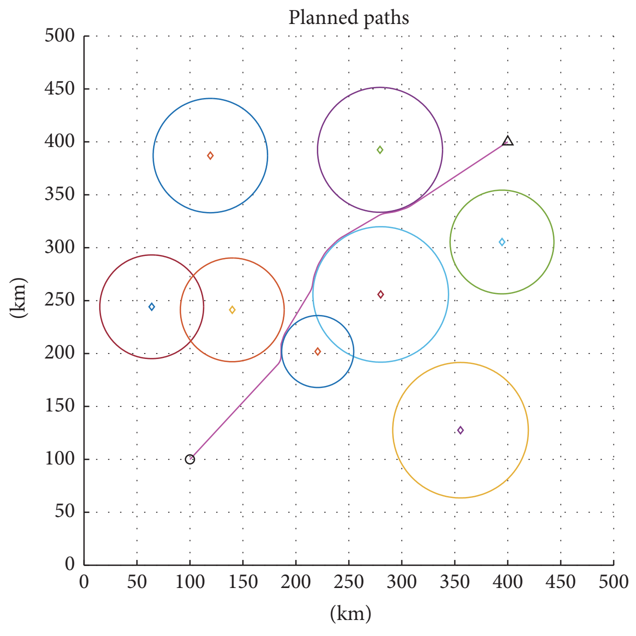


Figure 7: The third result.

5. Conclusion

This paper presents an improved heuristic algorithm which is an improved version of SAS algorithm for UCAV path planning. Our algorithm considers not only traditional constraints of path planning but also various flight constrained conditions, like angle information, track smooth straightening processing, and so on. Compared with [5], the simulation results show that angle information and trajectory smooth straightening processing are advisable, effective, and feasible. In addition, the algorithm can not only make the aircraft close to the target in the predetermined direction but also get an approximate optimal solution. And our improved algorithm has less extended nodes in complex conditions and the running efficiency is much better. Besides, some simulations have shown that our proposed new model and

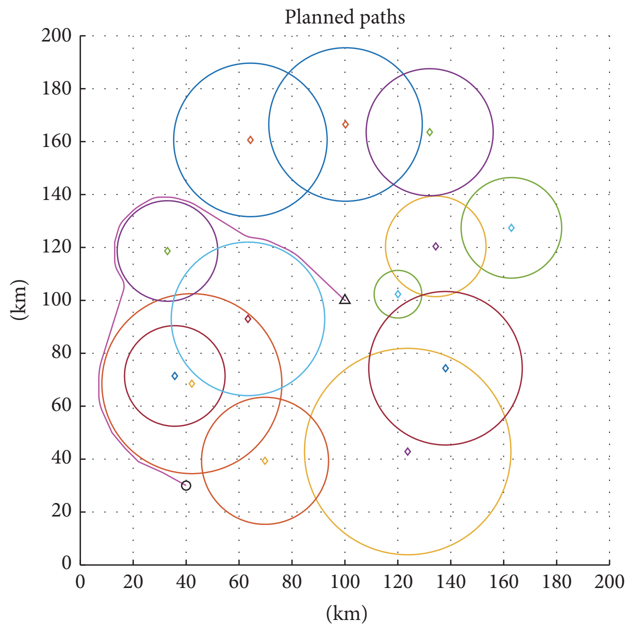


Figure 8: The fourth result.

algorithm can meet the flight restrictions and task demands of UCAV path planning.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was financially supported by National Natural Science Foundation of China (61401363), the Science and Technology on Avionics Integration Laboratory and Aeronautical Science Foundation (20155153034), and the Fundamental Research Funds for the Central Universities (3102016AXXX005, 3102015BJJGZ009).

References

[1] W. Ye, A. H. Zhu, and H. D. Fan, "Application of improved particle swarm optimization algorithm in UCAV path planning," *Computer Engineering*, vol. 34, pp. 178–180, 2008.

[2] Y. Zhang, L. Wu, and S. Wang, "UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimization," *Mathematical Problems in Engineering*, vol. 2013, Article ID 705238, 9 pages, 2013.

[3] X. C. Zhou, J. G. Yan, and R. Chen, "Cooperative path dynamic planning model of UCAV team based on global optimization method," *Journal of Electronic Science and Technology*, vol. 10, no. 4, pp. 363–367, 2012.

[4] C. W. Zheng, L. Li, F. J. Xu, F. C. Sun, and M. Y. Ding, "Evolutionary route planner for unmanned air vehicles," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 609–620, 2005.

[5] S. Chen, C. W. Liu, Z. P. Huang, and G. S. Cai, "Global path planning for AUV based on sparse A* search algorithm," *Torpedo Technology*, vol. 20, no. 4, pp. 271–274, 2012.

[6] Y. P. He, A. Zhang, and H. Y. Liu, "Path planning for UCAV based on Voronoi diagram and ant colony optimization," *Electronics Optics and Control*, vol. 16, no. 11, pp. 22–24, 2009.

[7] Z. C. Wang, J. Wang, and X. Jing, "Reach on multipath routing based on genetic algorithm," *Computer Engineering*, vol. 37, no. 20, pp. 197–199, 2011.

[8] Z.-J. Xu and S. Tang, "Flight path planning based on improved genetic algorithm," *Journal of Astronautics*, vol. 29, no. 5, pp. 1540–1545, 2008.

[9] J. F. Liu, G. Li, and H. T. Geng, "A new heuristic algorithm for the circular packing problem with equilibrium constraints," *Science China. Information Sciences*, vol. 54, no. 8, pp. 1572–1584, 2011.

[10] Y. Zhang, J. Chen, and L. Shen, "Hybrid hierarchical trajectory planning for a fixed-wing UCAV performing air-to-surface multi-target attack," *Journal of Systems Engineering and Electronics*, vol. 23, no. 4, pp. 536–552, 2012.

[11] H. Duan, Y. Yu, X. Zhang, and S. Shao, "Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm," *Simulation Modelling Practice and Theory*, vol. 18, no. 8, pp. 1104–1115, 2010.

[12] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, and J. Wang, "A hybrid meta-heuristic DE/CS Algorithm for UCAV path planning," *Journal of Information and Computational Science*, vol. 9, no. 16, pp. 4811–4818, 2012.

[13] J. B. Yuan, F. Yang, G. Y. Zhang, and Y. Liang, "A navigation method and its simulation for UAV formation flight," *Computer Simulation*, vol. 28, no. 11, pp. 64–67, 2011.

[14] R. J. Szczerba, P. Galkowski, I. S. Glickstein, and N. Ternullo, "Robust algorithm for real-time route planning," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 3, pp. 869–878, 2000.

[15] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, and M. Shao, "Path planning for uninhabited combat aerial vehicle using hybrid meta-heuristic DE/BBO algorithm," *Advanced Science, Engineering and Medicine*, vol. 4, no. 6, pp. 550–564, 2012.

[16] K. Zhang, P. P. Liu, W. R. Kong, J. Zou, and M. Liu, "An improved heuristic algorithm for UCAV path planning," in *Proceedings of the 11th International Conference on Bio-inspired Computing Theories and Applications*, pp. 54–59, Springer, Xi'an, China, October 2016.

[17] B. Huang and Y. M. Sun, "Improved methods for scheduling flexible manufacturing systems based on Petri nets and heuristic search," *Journal of Control Theory and Applications*, vol. 3, no. 2, pp. 139–144, 2005.