MATEC Web of Conferences **22**, 01040 (2015)
DOI: 10.1051/matecconf/20152201040
© Owned by the authors, published by EDP Sciences, 2015

A Novel Discrete Fruit Fly Optimization Algorithm for Intelligent Parallel Test sheets Generation

Fengrui Wang
College of Media and Communication, Liaocheng University, Liaocheng, Shandong, China

Wenhong Wang* & Jinxin Dong
College of Computer Science, Liaocheng University, Liaocheng, Shandong, China

Tianmin Feng
College of Media and Commutation, Liaocheng University, Liaocheng, Shandong, China

ABSTRACT: Parallel test sheet generation (PTSG) is a NP-hard combinational optimization problem, in which test sheet generation algorithm with high quality and efficiency is the core technology. Basic fruit fly optimization algorithm (FOA) has the defects of easily relapsing into local optimal and low convergence precision when solving PTSG problem. In this paper, a novel discrete fruit fly optimization algorithm is proposed to solve the PTSG problem, in which a discrete osphesis searching operator based on the problem-specific knowledge is designed to help the FOA escaping from being trapped in local minima. To evaluate the performance of the proposed algorithm, the simulation experiments were conducted using a series of item banks with different scales. The superiority of the proposed algorithm is demonstrated by comparing it with the particle swarm optimization algorithm and differential evolution algorithm.

Keywords: parallel test sheets generation; discrete fruit fly optimization algorithm; computer-aided testing system combinational optimization

1 INTRODUCTION

In order to handle emergency event or prevent cheating, parallel test sheets have to be constructed in a large variety of large scale examinations, such as achievement testing, college and graduate testing, professional certification and licensure testing, language testing and so on. Usually, parallel test sheets are generated according to similar test specifications, which should not only be paralleled in content but also should be paralleled in structure. With the efficiency and effectiveness of computer-based test, sheet generation has been confirmed by many early studies [1], and parallel test sheet generation by computer automatically has become an important and interesting research issue of computer-aided testing (CAT).

The PTSG problem is a NP-hard multiple constrained combinatorial optimization problem [1,2,3]. The goal of the problem is to select a subset of test items from test item bank to construct a series of parallel test sheets, which can meet multiple parallel test sheets assessment requirements (such as number of parallel test sheets, difficulty degree and etc.) of the user simultaneously. In order to solve the PTSG problem effectively, evolutionary algorithms are attractive option for their simplicity and computational efficiency. Previous investigations have shown that most of the existing test item selection approaches focus on the composition of a single test sheet, such as genetic algorithm [4], particle swarm optimization algorithm [5], and Tabu algorithm [6]. However, little research has been reported about parallel test sheet generation algorithm.

Very recently, a novel effective swarm intelligence optimization method, named fruit fly optimization algorithm (FOA), which is inspired by the knowledge from the fruit fly's foraging behavior, has been developed by Pan [7]. Compared with other existed swarm intelligence methods, FOA has some attractive characteristics including few parameters to adjusted, simple computation process, ease to understand and implement, and good convergence. Owing to these merits, the FOA has been successfully applied to tackling some academic and engineering optimization problems and becomes a competitive optimizer [8,9,10].

As the basic FOA is originally proposed to solve the continuous optimization problem, its main algorithm operations and strategies are developed for the continuous variables. When using it to solve PTSG problem, constrained by its optimization mechanism, FOA has the defects of easily relapsing into local optimal and low convergence precision in the later optimization. Therefore, in this study, we propose an improved FOA to solve the PTSG problem, which is designed based on the problem-specific knowledge. The effectiveness and efficiency of the proposed DFOA is demonstrated by simulation tests on a series of item

*Corresponding author: wwhem@126.com

This is an Open Access article distributed under the terms of the Creative Commons Attribution License 4.0, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Article available at http://www.matec-conferences.org or http://dx.doi.org/10.1051/matecconf/20152201040

banks.

2 MODLE FOR PTSG PROBLEM

Suppose test item bank $B=\{q_1,q_2,...,q_N\}$ contain N test items, K parallel test sheets with specific difficulty degree and similar discrimination need to be composed out of test item bank B. Moreover, all the K parallel test sheets will involve M course-related concepts $C=\{c_1,c_2,...,c_M\}$ to be tested and have no same test items. The variables used in the model are defined as follows.

2.1 *Test item attributes*

In addition to the test item content and answer, each question $q_i \in B$, $i \in \{1,2,...,N\}$, has $4+M$ measurement attributes:

Num_id: it is used to store the question identity.

dis_i , $1 \le i \le N$: discrimination degree of q_i.

dif_i , $1 \le i \le N$: difficulty degree of q_i.

t_i , $1 \le i \le N$: time needed for answering q_i.

$rela_{ij}$, $1 \le i \le N$, $1 \le j \le M$: degree of association between q_i and concept c_i.

2.2 *Parallel test sheets specification*

K: number of parallel test sheets that will be composed simultaneously.

Dif: target degree of difficulty for parallel test design.

h_j,$1 \le j \le M$: lower bound ratio on the expected relevance of C_j for each test sheet.

t_{upp}: upper bound on the expected time needed for answering each test sheet.

t_{low}: lower bound on the expected time needed for answering each test sheet.

S: number of test items that each parallel test sheets must contain.

2.3 *Problem Model*

The model aims to choose a fixed number of test items so that the maximum difference of discrimination degrees between any pair of the parallel test sheets is minimized and the difficulty degrees of all of the test sheets are the closest to the specified target difficulty. The goal of PTSG problem is to minimize the following objection function:

$$\text{Min } Z(x)=\alpha \times f(x)+\beta \times h(x) \tag{1}$$

Where

$$f(x)=\max_{1\le k,l\le K}\left\{\left|\frac{\sum_{i=1}^{N} dis_i x_{ik}}{\sum_{i=1}^{N} x_{ik}}-\frac{\sum_{i=1}^{N} dis_i x_{il}}{\sum_{i=1}^{N} x_{il}}\right|\right\} \tag{2}$$

$$h(x)=\left\{\frac{1}{k}\sum_{k=1}^{K}\left|\frac{\sum_{i=1}^{N} dif_i x_{ik}}{\sum_{i=1}^{N} x_{ik}}-Dif\right|\right\} \tag{3}$$

Subject to

$$\frac{\sum_{i=1}^{N} rela_{ij} x_{ik}}{\sum_{j=1}^{M}\sum_{i=1}^{N} rela_{ij} x_{ik}} \ge h_j, 1 \le j \le M, 1 \le k \le K \tag{4}$$

$$\sum_{i=1}^{N} t_i x_{ik} \ge t_{low}, 1 \le k \le K \tag{5}$$

$$t_{upp} \ge \sum_{i=1}^{N} t_i x_{ik}, 1 \le k \le K \tag{6}$$

$$\sum_{i=1}^{N} x_{ij} x_{ik} = 0, 1 \le j \ne k \le K \tag{7}$$

$$\sum_{i=1}^{N} x_{ik} = S, 1 \le k \le K \tag{8}$$

If test item q_i is chosen into the K parallel test sheet, then the decision variable x_{ik}=1, otherwise x_{ik}=0. Constraint (4) indicates that the selected test items in each parallel test sheet must have a relevance degree no less than the expected relevance degree to each specified concept. Equations (5) and (6) emphasis that total expected test time of each generated parallel test sheet must be in its specified range. Constraints (7) require that no pair of parallel test sheets can contain any identical test items. Finally, equation (8) claims that every parallel test sheet must contain S test items.

3 DISCRETE FRUIT FLY ALGORITHM FOR PTSG PROBLEM

In this section, the basic FOA is introduced firstly. Then a novel discrete fruit fly optimization algorithm (DFOA) is illustrated for solving PTSG problem, including the flowchart of FOA, population initialization, the encoding and decoding strategy, discrete osphesis food searching operator, vision food searching operator and repair operator.

3.1 *FOA introduction*

Fruit fly optimization algorithm [7] is a new swarm intelligence method for global optimization, which is

inspired by the food foraging behavior of fruit flies. Generally speaking, FOA mimics two foraging processes: (1) the fruit flies smell the food source by osphresis organs and fly towards the food location; (2) after getting close to the food, they fly towards a better direction using the sensitive vision ability and the location information of the swarm. The procedure of FOA can be presented as follows.

Step1. Initialize the algorithm parameter, that is, population size and a termination criterion.

Step2. Initialize the location of fruit fly swarm randomly in the search space.

Step3. Generate a fruit fly population randomly around the fruit fly swarm location using osphresis foraging process.

Step4. Evaluate the smell concentration or the fitness value of the individual location of each fruit fly.

Step5. Find the best food source, and then the fruit fly swarm flies towards that location using vision foraging ability.

Step6. Repeat step 3-5 to optimize iteratively until the termination condition is reached.

3.2 *DFOA procedure*

The evolution procedure of FOA reveals that the swarm will only learn from the best individual, which makes the algorithm be relapsed into local optimal in the later evolution easily. To balance the global and local search effectively, we designed an improved FOA for PTSG problem, which is presented in Figure 1.

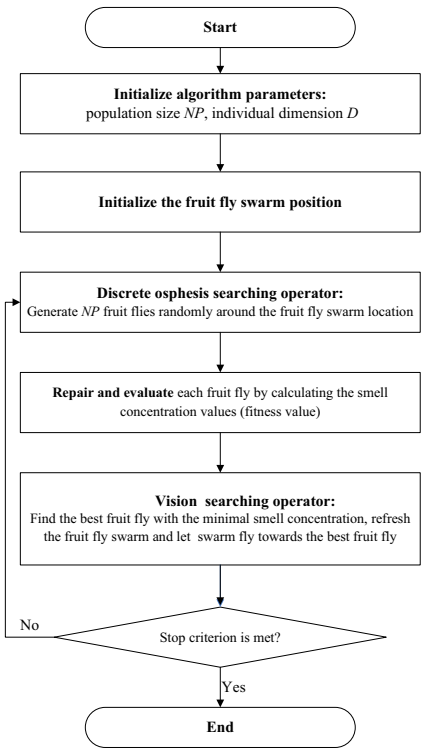

Figure 1. Flowchart of the DFOA

3.2.1 *Initialize the algorithm parameter and the location of the fruit fly population*

The population size *NP* and the dimension of each fruit fly individual *D* are initialized at first. The initial fruit fly swarm location is chosen randomly in the search space as follows.

$$xPosition_j = rand(), j = 1,2,...,D \quad (9)$$

$$yPosition_j = rand(), j = 1,2,...,D \quad (10)$$

where, rand() is a function to generate random number with outcome $\in [-\infty, \infty]$.

3.2.2 *Encoding and decoding strategy*

In the DFOA, a fruit fly is responding to a solution of the PTSG problem. To be specific, a fruit fly of *G* generation is denoted by two D-dimensional vector $x_i^G = \{x_{ij}^G \mid i = 1,2,...,NP, j = 1,2,...,D\}$ and $y_i^G = \{y_{ij}^G \mid i = 1,2,...,NP, j = 1,2,...,D\}$, where *NP* and *D* don't change during the population evolution process. Inspired by the research report [1], each fruit fly individual is sequentially divided into *K* segments with equal length, in which each segment represents the selected test item number of a dependent parallel test-sheet. Therefore, the dimension *D* of each fruit fly individual is equal to the total test item number selected in the *K* parallel test-sheets.

For the PTSG problem, each decision variable of the fruit fly individual is used to denote a number of the selected test item. The following decoding strategy is used to translate each component of the individual into a test item number.

(1) Distance vector dis_i^G is calculated by equation (11)

$$dis_{ij}^G = \sqrt{x_{ij}^G * x_{ij}^G + y_{ij}^G * y_{ij}^G} \quad (11)$$

where, each component dis_{ij}^G denotes the distance of point (x_{ij}^G, y_{ij}^G) to the origin point $(0,0)$.

(2) The smell concentration vector s_i^G corresponding to the distance vector is calculated by equation (12).

$$s_{ij}^G = 1 / dis_{ij}^G \quad (12)$$

(3) As PTSG is a discrete combinatorial optimization problem, the real parameter individual s_i^G must be decoded to a solution composed of integer variables. According to the designed decoding schema, each element s_{ij}^G of the smell concentration vector should belong to $[0,1)$. If $s_{ij}^G \notin [0,1)$, the later repair operator will be employed until $s_{ij}^G \in [0,1)$.

The smell concentration vector s_{ij}^G is mapped to the PTSG problem solution $tin = (tin_1, tin_2, ..., tin_D)$ as the following way:

$$tin_j = floor(s_{ij}^G * itemNum) \quad (13)$$

where, *floor*() is a function for truncating a floating-point number to an integer.

3.2.3 *Discrete osphresis searching operator*

The basic FOA is originally proposed to solve the continuous optimization problem, its main algorithm operations and strategies are developed for the continuous variables, which has the problems of being trapped in local optimal or premature in the later optimization procedure. In order to improve the performance of FOA, a discrete osphresis food searching strategy based on the characteristic of the PTSG problem is designed to balance the global and local searching ability of FOA. The procedure of the strategy is shown in Figure 2.

Procedure

1: Set i={1,2,…,NP}

2: FOR j=1 to D

3: $x_{ij}^G = xPosition_j^G$

4: $y_{ij}^G = yPosition_j^G$

5: END FOR

6: FOR m=1 to K

7: Select a integer z from {1,2,…,S}randomly

8: Set $c = z + (\text{m-1}) \times S$

9: $x_{ic}^G = xPosition_c^G + (2 \times rand[0,1]\text{-}1)$

// $rand[0,1]$ produce a random float number of [0,1]

10: $y_{ic}^G = yPosition_c^G + (2 \times rand[0,1]\text{-}1)$

11: END FOR

Figure 2. Discrete osphesis search operator

3.2.4 *Vision searching operator*

In vision food searching process, the smell concentration of each fruit fly is evaluated by equation (1). Then, the best fruit fly with the minimum smell concentration in the population is found out, and the fruit fly group is replaced with the best fruit fly. Simultaneously, the fruit fly population location is refreshed with the best individual location.

$$xPosition_j = x_{bestIndexj}, j = 1,2,...,D \quad (14)$$

$$yPosition_j = y_{bestIndexj}, j = 1,2,...,D \quad (15)$$

where, *bestIndexj* is the position index of the best individual.

3.2.5 *Repair operator*

If the element s_{ij}^G of each individual not belong to $[0,1)$, the repair operator will be employed by equation (20) and (21) until the newly refreshed $s_{ij}^G \in [0,1)$.

$$x_{ij}^G = x_{ij}^G \times (1 + rand[0,1]) \quad (16)$$

$$y_{ij}^G = y_{ij}^G \times (1 + rand[0,1]) \quad (17)$$

4 SIMULATION TESTING RESULTS AND COMPARISION

To test the performance of the proposed DFOA, five simulation item banks with different scale are used. There are 20 concept topics in each item bank and all the item banks have the same table structure. All the algorithms used in this paper are coded using C language in Microsoft Windows 2008 server (32bite) operating system on a PC with 2.7GHz CPU, 2G RAM.

4.1 *Performance comparison*

First, we compare the DFOA with the particle swarm optimization algorithm (PSO) and differential evolution (DE) algorithm on five simulation item banks.

Suppose three parallel test sheets with sixty test items will be generated. The parameters of the three algorithms were set empirically in the following. We set w=0.5, $c1$=2.05, $c2$=2.05, $MAXV$=1.5, NP=20, and D=180 for PSO. Set NP=20, D=180, F=0.5, CR=0.1for DE. For DFOA, we set NP=30 and D=180. All algorithms are stopped when the fitness value is smaller than 1 or the running time reaches 10 minutes. Each algorithm is executed 20 times on each simulation item bank independently.

Table 1 gives the comparative results of success ratio for three approaches on five different item banks, where the column “SR” is the success ratio, which denotes the ratio of number of successful runs to that of total runs. Table 2 presents the time efficiency of the three compared algorithms running successful on three item banks, which is reflected by the execution time of the algorithm in searching for the optimal solution. In table 2, “*Avg*” and “*Std*” stand for the average value and standard deviation of the execution time in 20 runs. According to the PTSG problem model, a run is considered to be successful if it stops after the fitness value of the algorithm is smaller than one, and simultaneously the running time is not more than 10 minutes. The shorter the execution time, the faster the algorithm find the optimal solution.

For the small-scale test item banks, it can be seen from Table 1 that the success ratios of DFOA are all 100%. Although the success ratios of PSO and DE on large-scale can reach 100%, but the success ratios of them on small-scale test item banks will descend. Furthermore, as shown in Table 2, it can be clearly seen that the DFOA is superior to the PSO and DE in terms of the average value and standard deviation of the execution time on three large-scale test item banks. The comparative results of Tables 1 and 2 indicate that the DFOA is able to obtain better solutions than the PSO and DE in a more effective way on five item banks with different scale.

Then, the three compared algorithms were used to compose two and three parallel test sheets simultaneously on item bank 10000. The stopping criterion is the same with previous experiment and each algorithm is also executed 20 times on every test sheet specification. The comparative results are listed in Table 3, in which "NOPTS" denotes the number of parallel test sheets, "*Avg*" and "*Std*" respectively denotes the average value, and standard deviation of the execution time.

It can be observed from Table 3 that when generating two or three parallel test sheets simultaneously, DFOA exhibit superior optimization performance than PSO and DE in terms of the average value and the standard deviation of the execution time. Besides, although the average execution time of the three compared algorithms will both increase when the number of parallel test sheets grows, but the execution time DFOA has no significant increase.

Table 1. Success ratio of three compared algorithms

Item bank scale	Success ratio		
	PSO	DE	DFOA
1000	20%	95%	**100%**
2000	90%	100%	**100%**
5000	100%	100%	**100%**
10000	100%	100%	**100%**
20000	100%	100%	**100%**

Table 2. Execution time of three algorithms on various item banks (second)

Item bank scale	PSO		DE		DFOA	
	Avg	*Std*	*Avg*	*Std*	*Avg*	*Std*
5000	85.19	99.32	28.25	11.72	**0.41**	**0.04**
10000	120.22	167.32	15.34	7.12	**0.69**	**0.05**
20000	135.99	151.10	15.91	9.49	**1.29**	**0.13**

Table 3. Experiment results on the specified number of parallel test sheets (second).

NOPTS	PSO		DE		DFOA	
	Avg	*Std*	*Avg*	*Std*	*Avg*	*Std*
2	11.75	6.47	1.01	0.42	**0.62**	**0.05**
3	120.22	167..32	15.34	7.12	**0.69**	**0.05**

4.2 *Convergence analysis*

The convergence curves of the three compared algorithms on item bank 1000 are shown in Figure 3. The convergence curves of the Figure 3 reveals that DFOA can balance the exploitation and exploration effectively, therefore it has better convergence speed and accuracy than PSO and DE algorithm.

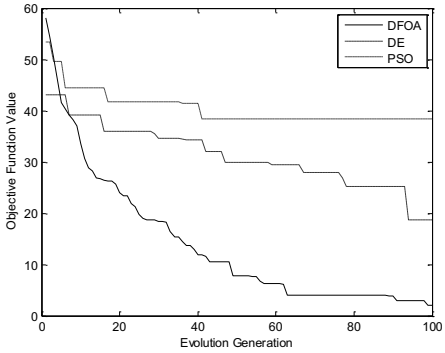

Figure 3. Convergence curves of three algorithms

Figure 4 illustrates the flying route of the fruit fly swarm in searching space. The starting point and the end point are denoted by square and hollow circle shapes respectively. It can be found out from Figure 4 that the flying route of the swarm is very stable, and the swarm can find the food source directly and accurately after less volatility, which indicates the proposed discrete osphesis search operator has powerful local search ability, thus causing the DFOA has better convergence speed and accuracy.

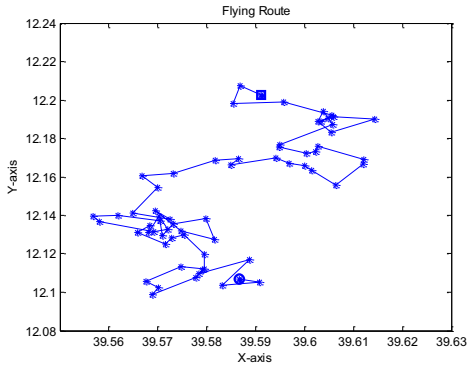

Figure 4. Flying route of the fruit fly swarm

5 CONCLUSION

This article dealt with the parallel test sheet generation problem, which is very important in computer-aided

testing system development. To solve the problem, a novel discrete fruit fly algorithm was put forward, in which a discrete osphesis food search operator is designed based on the characteristic of the PTSG problem. Several comparative experiments results conducted on a series of item banks with different scale demonstrate that the DFOA can solve the PTSG problem effectively. In addition, it could be interesting to make the PTSG problem model in line with the actual exam and how to further exploring problem-specific characteristics to enhance the performance of the DFOA in the future.

ACKNOWLEDGEMENTS

This paper is supported by Science and Technology Program Project of Shandong Province Higher Educational, China (GN: J12LN38&J13LN33) and the Educational and Scientific Project of Shandong Province "the Twelfths-Five-Year-Plan" (GN: 2013GG051).

REFERENCES

[1] Gwojen Huang, Huichun Chu, Pengyeng Yin. 2008. An innovative parallel test sheet composition approach to meet multiple assessment criteria for national tests. *Computer & Education*, 51(3): 1058-1072.

[2] Tingyi Chang, Youfu Shiu. 2012 Simultaneously construct IRT-based parallel tests based on an adapted CLONALG algorithm. *Application intelligence*, 36(4): 979-994.

[3] Tsufeng Ho, Pengyeng Yin, Gwojen Hwang, Shyong Jian Shyu, Yanan Yean. 2009. Multiple-objective parallel test sheet composition using enhanced particle swarm optimization. *Educational Technology & Society*, 12(4): 193-206.

[4] Gwojen Hwang, Bertrand M. T. Lin, Hsienhao Tseng, Tsungliang Lin. 2005. On the development of a computer-assisted testing system with genetic test sheet-generating approach. *IEEE Transaction on Systems Man and Cybernetic*, 35(4): 590-594.

[5] Kun Hua Tsai, Tzone I. Wang, Tung Cheng Hsieh, Ti Kai Chiu, Ming Che Lee. 2010. Dynamic computerized testlet-based test generation system by discrete PSO with partial course ontology. *Expert Systems with Applications*, 37: 774-786.

[6] Minh Luan Nguyen, Siu Cheung Hui, Alvis C.M. Fong. 2011. A divide and conquer tabu search approach for online test paper generation. *Proceedings of AI 2011*, perth, 5-8 December 2011:717-726.

[7] W.T. Pan. 2012. A new fruit fly optimization algorithm: taking the financial distress model, *Knowl.-Based Syst*., 26: 69-74.

[8] L. Wang, X.L. Zheng, S.Y. Wang. 2013. A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem, *Knowl.-Based Syst*., 48: 17-23.

[9] S.M. Lin. 2013. Analysis of service satisfaction in web auction logistics service using a combination of fruit fly optimization algorithm and general regression neural-network, *Neur. Comput. Appl*., 7:459-465.

[10] Quanke Pan, Hongyan Sang, Junhua Duan, Liang Gao. 2014. An improved fruit fly optimization algorithm for continuous function optimization problems, *Knowl.-Based Syst.*, 62: 69-83.