Final Accepted Version. This paper has been accepted by The 2017 4th International Conference on Systems and Informatics (ICSAI 2017), IEEE press, pp. 1328-1333, Hangzhou, Zhejiang, China, 11-13 November, 2017

Gravitation Field Algorithm with Optimal Detection for Unconstrained Optimization

Lan Huang, Xuemei Hu, Yan Wang*, Fang Zhang, Zhendong Liu
College of Computer Science and Technology, Jilin University
Changchun, China
*E-mail: wy6868@jlu.edu.cn

Wei Pang*
Department of Computing Science
University of Aberdeen
Aberdeen, UK

*E-mail: pang.wei@abdn.ac.uk

Abstract—**Gravitation field algorithm (GFA) is a novel optimization algorithm derived from the Solar Nebular Disk Model (SNDM) in astronomy, based on the formation of planets, in recent years. In this research, an improved GFA with Optimal Detection (GFA-OD) is proposed for unconstrained optimization problems. Optimal Detection can efficiently locate the space that more likely contains the optimal solution(s) by initializing part of dust population randomly in the search space of a given problem, and then improves the accuracy of solutions. The comparison of results on four classical unconstrained optimization problems with varying dimensions demonstrates that the proposed GFA-OD outperforms many other classical heuristic optimization algorithms in accuracy, efficiency and running time in lower dimensions, such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO).**

Keywords-gravitation field algorithm; optimal detection; unconstraint optimization;

I. Introduction

It is well known that the optimization problems have existed in all aspects of life, many of which are unconstrained optimization problems. Although research in optimization has been a long history, it is still challenging to acquire desirable solutions for complex unconstraint optimization problems, regardless of accuracy, efficiency, and running time. The traditional mathematical methods for solving those unconstrained optimization problems, such as the steepest descent method [1], the Newton method [2], the conjugate gradient method [3], and the quasi-Newton method [4], have such requirements that the optimization problems have to be continuous and differentiable and even more prior knowledge needs to be available in advance. This makes it difficult to solve these complex optimization problems in higher dimensions.

To address the above issues, many innovative computational intelligence (CI) have been proposed over the past several decades, some of which have achieved varying degrees of success. The significant advantages make computational intelligence (CI) algorithms widely adopted to solve unconstrained optimization problems in real-world application fields. During the second half of last century, many effective computational intelligence algorithms have been developed based on imitations of natural phenomena. For instance, Simulated Annealing (SA) algorithm, proposed by Metropolis *et al.* in 1953, was inspired by the annealing process in metallurgy. Genetic algorithm (GA) [5, 6] originally proposed by J. Holland in 1975, simulates the natural evolution selection process, based on Darwin's biological evolution theory. Ant colony optimization (ACO) [7, 8] proposed by Marco Dorigo in 1992 is based on the heuristic process of ants' food discovery and incorporated the communication mechanisms among the colony members. Particle swarm optimization (PSO) [9, 10] proposed by Eberhart and Kennedy in 1995 simulates a series of bird foraging behavior. Gravitation field algorithm (GFA) [11,12,13] proposed by Zheng *et al.* in 2012 simulates the formation of planets based on the SNDM [14] in astronomy. These innovative CI algorithms have good performance and some of them have been successful applied in many real-world optimization problems, especially, in unconstrained optimization problems.

Although the above innovative CI algorithms have achieved good results when solving unconstrained optimization problems with respect to accuracy, efficiency, or running time, they still have much room for improvement, such as, improving the accuracy of solutions in the given time.

In this paper, we address the above key issues and propose an improved version of GFA, which is called Gravitation Field Algorithm with optimal detection (GFA-OD). In GFA-OD, we devise an improved strategy called Optimal Detection locate a small enough search space which more likely contains the optimal solution(s) by initializing part of the dust population randomly in given search space for a problem, which decreases the running time and suits better for any other swarm intelligence optimization algorithms. From the experimental results on four complex unconstraint optimization problems we find that GFA-OD is superior to two other classical optimization algorithms: GA and PSO, in terms of accuracy, efficiency, and running time. The experimental results demonstrate the promising application potential of GFA-OD to more complex and real-world unconstrained optimization problems.

This work is supported by the National Natural Science Foundation of China (Grant Nos. 61472159, 61572227), Development Project of Jilin Province of China (Nos. 20160204022GX, 20160414009GH, 2017C033).

II. Gravitation Field Algorithm with optimal detection

A. *The Original Gravitation Field Algorithm*

GFA originally proposed by Zheng *et al.* in 2012 is a novel heuristic search algorithm. The basic idea of GFA is to simulate the formation process of planets, derived from the Solar Nebular Disk Model (SNDM). In GFA, all individuals can be mimicked as dusts with mass and each of them belongs to a certain group, and the best one is regarded as center dusts and others are the surrounding dusts in each group. Based on SNDM, each center dusts attracts their surrounding dusts by the gravitation field, and the field makes all surrounding dusts move toward their center dusts with heaviest mass. In addition, each dust has four characteristics: position, mass, group number, a Boolean value that records whether it is a center. The first one corresponds to a solution of the problem, the second one is initialized randomly, and the other two are determined by mass function. The detailed steps of GFA is as follows:

1) First, initialization. Generating n dusts $D_i\left(i=1,2,\ldots n\right)$ randomly distributed in the mass function domain boundary to establish the initial solution space.

2) Second, dividing the search space into several subspaces. In any subspace, the dust with the biggest value of mass is called the center dust and we assign the flag $F_i=1$, the others are surrounding dusts and we assign the flag $F_i=0$.

3) Third, moving dusts. The pace of movement is determined by Equation (1).

$$Pace_i = M \times dis_i\ , \qquad (1)$$

where dis_i is the distance between the moving surrounding dust and its center dust in [11, 12] and M is a weight value for the distance.

4) Fourth, absorbing dusts. Some surrounding dusts which are close enough to their center dusts are absorbed by their center dusts and are eliminated from the initial search space for increasing the convergence speed of GFA.

5) Finally, checking the termination criterion. If the algorithm does not meet the stopping condition, GFA will go to 3), otherwise, the algorithm stops.

The study on initial application cases demonstrates that original GFA can handle unimodal and multimodal functions optimization effectively. But there are still some limitations of the original GFA.

1) It is particularly sensitive to the initial population. It is reported that the accuracy of solution is closely related to the initial population.

2) The accuracy of optimal solution is not desirable. As with other swarm intelligence optimization algorithms, it is easy to fall into local optima.

B. *Gravitation Field Algorithm with optimal detection*

To overcome the shortcoming of GFA presented above, an improved Gravitation Field Algorithm with optimal detection (GFA-OD for short) was proposed. Optimal Detection was performed when the parameter settings are completed. It can efficiently locate a small enough search space which more likely contains the optimal solution(s) by initializing part of dusts population randomly in certain areas of the problem search space, and this can decrease the number of iterations for the whole GFA-OD and thus shorten the algorithm running time. In addition, the improvement can help obtain a more desirable optimal solution.

1) Optimal Detection

The original GFA, which is started with dusts population being initialized randomly in given search space, makes it a long iterative process to find a solution that meets the accuracy requirement. And the solution that the original GFA finds may be even a suboptimal solution whose accuracy could be comparable to the solutions that classical heuristic search algorithms find, such as GA and PSO. Therefore, to improve the accuracy and shorten the overall running time, an optimal solution improvement strategy named Optimal Detection is proposed for GFA-OD in this research.

Optimal Detection is not only an important improvement in this paper but also the core part for GFA-OD. The task of Optimal Detection is to acquire a small enough space quickly involving the optimal solution with higher probability. It begins with part of dusts being initialized randomly in the problem search space. Then the dusts in the top *w*% of the population are selected to calculate the boundary of the target space. If the space is small enough, this process will stop and the next process will be performed; otherwise repeat the above process of detection several times until the space is small enough, based on the requirement of accuracy for a given problem. The small enough target space called Optimal Space involving the optimal solution will be identified eventually when the process of detection is completed. Once the algorithm found the Optimal Space, GFA-OD does not need to initialize the population randomly any more across the entire search space, and it just needs to initialize the population randomly within the optimal space that GFA-OD has identified. The strategy of initializing the population in optimal space shortens the running time because GFA-OD does not need a long iterative process any more to find a solution that meets the accuracy requirement. The Optimal Detection proposed for GFA-OD is illustrated in Fig. 1.

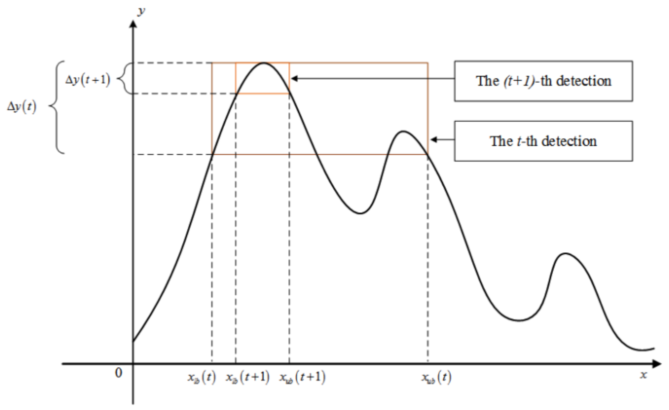


Figure 1. The process of optimal detection

Fig. 1 shows that the space that GFA-OD acquires to be smaller and smaller with the process of ongoing detection. In each detection, the dusts in the top $w\%$ of current population, namely *maxs*, need to be picked out. To get the dusts in the top $w\%$ of the entire population, GFA-OD has to spend time in sorting part of dusts. Obviously, the smaller the value of the $w\%$ is, the less time on sorting the top $w\%$ dusts is required. Therefore, the value of $w\%$ determines the speed at which the optimal space is found. In addition, the value of $w\%$ is closely related to the quality of the finally solution. When the value of $w\%$ is close to 0, the time spent on detection is less, but in this case, the algorithm falls into local optimal solutions with higher probability. When the value of $w\%$ is close to 1, the time spending on detection is higher; in this case, the algorithm falls into local optimal solutions with lower probability. In this paper, we set $w\%$ as $5\%-20\%$ based on the size of population and the dimensions of given problems.

```
***OptimalDetect***
1. for $k = 1:d$
2.    $maxi \leftarrow inf$
3.    $mini \leftarrow inf$
4.    for $i = 1:lm$
5.         if $maxs_i.X_k > maxi$ then
6.              $maxi \leftarrow maxs_i.X_k$
7.         elseif $maxs_i.X_k < mini$ then
8.              $mini \leftarrow maxs_i.X_k$
9.         end if
10.   end for
11.   $bestBound(i:1) \leftarrow mini$
12.   $bestBound(i:2) maxi$
13. end
```

The pseudo code of acquiring the value of boundary of the Optimal Space is presented in ***OptimalDetect***, which is the core part of the optimal detection. Where d is the number of dimensions of the search space for a given problem, N is the size of dusts population, lm is the size of dusts that are the dusts in the top $w\%$ of the entire population, $0 < w\% < 1$, variable *maxi* is used to get a $d \times 2$ matrix named *bestBound* in this research, which stores the values of boundary for the optimal space.

For the sake of getting the *maxs* to calculate the values of boundary for the target space, the top $w\%$ dusts need to be picked out in the entire current population of size N firstly, which requires $O(N)$ comparisons for each dust. When this process continues to find all members of *maxs* in the dust population, the total complexity of this process is $O(N \times lm)$. To get the lower bound and the upper bound for each dimension with the *maxs* of size lm that is described in ***OptimalDetect*** requires $O(d \times lm)$ comparisons. In summary, the process of optimal detection requires an overall of $O(N \times lm + d \times lm) = O((N+d) \times N)$ complexity. Note that $O(N)$ storage is just required for this process.

2) Main loop

According to the above details of GFA-OD, the pseudo-code of GFA-OD is given in **Algorithm** GFA-OD.

```
**Algorithm** GFA-OD
1. bestBound ← OptimalDetect (N, bound, targetFun);
2. initialdusts ← Initial (bestBound, N, targeFun)
3. [groupdusts,center] ← Group (initialdusts, G)
4. dust ← groupdusts
5. while the stop conditions are not met do
6.     movedust ← Move (dust, center, targetFun);
7.     absdust Absorb (center, movedust, bestBound);
8.     rotatedusts ← Rotate(absorbdust, targetFun)
9.     dust ← absdust
10.    [center,dust] ← GetCenters (G, dust);
11.    best ← GetBest (best, center);
12. end
13. return optimal solution
```

In Algorithm GFA-OD, the variable bound gives the value range of X_i in all dimensions for each solution D_i, N is the size of population, G is the number of groups, and d is the number of dimensions of the search space.

The method '***OptimalDetect***' corresponds to the process of optimal detection presented above and saves the information of optimal space with the variable *BestBound*. The method '***Initialize***', '***Group***', '***Move***', '***Absorb***', '***Rotate***' still corresponds to the five processes as in the original GFA. The method '***GetCenters***' is used to get the center dusts for each group and the method '***GetBest***' is devoted to update the historical best solution. If M is the number of iterations and set as one of the stop conditions, it is obvious that the methods '***Initial***', '***Group***', '***MoveAndRotate***', '***Absorb***', '***GetCenters***' and '***GetBest***' just require $O(N)$ computations. Since '***OptimalDetect***' require $O((N+d) \times N)$ computations, then the main loop of GFA-OD require overall $O(M(N+d) \times N)$ computations. Note that $O(N)$ storage is only required for GFA-OD.

The strategy proposed in GFA-OD overcomes the limitations of the original GFA, such as the accuracy of solution not meeting the requirements of XX and the long running time. However, the introduction of this strategy is at the cost of running time. It is noted that the strategy of optimal detection avoids the long iterative process and shortens the running time although the time complexity has increased to $O(M(N+d) \times N)$.

III. Experiments and Discussions

To assess the performance of GFA-OD proposed in this research, the following four classical unconstrained optimization problems in Table I are chosen, which are used to test the accuracy, efficiency, and running time in different dimensions. At the same time, this series of classical test problems are also applied to the GA and PSO to compare with GFA-OD. 100 independent trials of each algorithm are performed for solving four benchmark problems, and we choose the median, the Mean squared error, and the Standard Deviation of the obtained solutions to measure the performance of these three algorithms. Lastly, we present the experimental results and make discussions in this section.

A. Benchmark problems and performance measure

1) Test problems

The functions listed in Table I are part of the most commonly used functions and datasets to test the performance of unconstraint optimization algorithms. These benchmark problems are chosen from a number of significant past studies in unconstraint optimization problems. The functions are known widely as the Sphere [15], Griewangk [15], Ackley [15], Zakharov [15], Rotated Hyper-Ellipsoid [16], and Levy [16] functions, and they can be scaled to any number of variables (dimensions). Table I also shows the domain, the objective function, and the global minimum for every benchmark problem. Additionally, the four benchmark problems have the same global minimum, and the global minimum of the four problems are equal to zero, regardless of the number of variables. The Zakharov, Sphere and Rotated Hyper-Ellipsoid functions are continuous, convex, unimodal, and multidimensional; the first one is plate-shaped and the latter two are bowl-shaped in their two-dimensional forms. The other functions are multimodal, multidimensional, with a large number of local optima.

TABLE I. THE TEST PROBLEMS OF GFA-OD, GA AND PSO

Name	Variable ranges	Objective function and optimum
Sphere	$x_i \in [-50,50]$	$f(x)=\sum_{i=1}^{d} x_i^2,\ f(0,\mathrm{L}\ ,0)=0$
Griewangk	$x_i \in [-5,5]$	$f(x)=\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$, $f(0,\mathrm{L}\ ,0)=0$
Ackley	$x_i \in [-15,30]$	$f(x)=20+e-20e^{\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right)}-e^{\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right)}$, $f(0,\mathrm{L}\ ,0)=0$
Zakharov	$x_i \in [-5,10]$	$f(x)=\sum_{i=1}^{d}x_i^2+\left(\sum_{i=1}^{d}0.5ix_i\right)^2+\left(\sum_{i=1}^{d}0.5ix_i\right)^4$, $f(0,\mathrm{L}\ ,0)=0$

2) Performance measure

To investigate the accuracy, efficiency, and running time of GFA-OD, we choose three common performance metrics to evaluate the performance of the three algorithms: GFA-OD, GA and PSO.

The median value, we choose the median value of solutions that are found by the three algorithms from 100 individual trials.

The Mean squared error (MSE) [17]*:* it has the following general definition as in (2), where *n* is the number of tests, $f(x_i)$ is the solution of the *ith* trial and $f_{opt}(x)$ is the actual global minimum. Obviously, a lower value of MSE means that the algorithm achieves better performance.

$$MSE=\frac{1}{n}\sum_{i=1}^{n}\left(f(x_i)-f_{opt}(x)\right)^2. \tag{2}$$

The Standard Deviation (STD) [18]*:* it has the following general definition as in (3), where *n* is the number of tests, $f(x_i)$ is the solution of the ith trial and $\overline{f(x)}$ is the mean of all the $f(x_i)$. As with MSE, a lower value of MSE stands for better performance of the algorithm.

$$STD=\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(f(x_i)-\left(\overline{f(x)}\right)\right)^2}. \tag{3}$$

The median value and the MSE are adopted to measure the accuracy of solutions that are obtained by three different algorithms. The STD is employed to measure the stability of the performance for the three test algorithms.

3) Parameter settings

The basic parameters for three algorithms are set as in Table II. The size of the population is set as 1000* d , where d is the number of dimensions of given search space, moreover, d =2, 3, 5, 10, 20 in this research. The maximum number of iterations are set differently for the three different algorithms to control their running time in the same dimension, which for GFA-OD is 20, for GA and PSO is 200* d . It is noted that the maximum number of iterations for GFA-OD is much less than that for GA and PSO, because the introduction of Optimal Detection can find the approximate location of the optimal solution in search space, which is employed to avoid the long iterative process. The number of groups is just set as 3 for GFA-OD to ensure multi-agents. The parameter TolFun is the average change in the value of the fitness function or mass function and it is changed from the default value of 1.0e-6 to 1.0e-30 to ensure that the optimal solution they acquire is accurate enough.

TABLE II. THE PARAMETERS SETTING FOR EGFA, GA AND PSO

Parameters setting	EGFA	GA	PSO
Population size	100* d	100* d	100* d
Max number iterations	20	200* d	200* d
The number of groups	3	-	-
TolFun	1.0e-30	1.0e-30	1.0e-30

B. Results and Discussions

In this research, every benchmark problem shown in Table I, is scaled to different number of variables, whose value range of each dimension is also set in Table I. The basic parameters of each algorithm are set as in Table II. Three performance metrics, the median, MSE and STD, are employed to measure the accuracy and efficiency of each algorithm in given running time. In this research, we do not make any systematic investigations on finding the best parameters for EGFA and we just focus on the accuracy of each algorithm in a given running time. In addition, a higher accuracy of the optimal solution indicates better performance of the algorithm.

To compare the performance of EGFA, GA and PSO on unconstrained optimization problems with different number of dimensions in given running time, four test problems with 2, 3, 5, 10, 20 dimensions are considered in this study. The comparisons of three algorithms: EGFA, GA and PSO on three of the four test problems in median, MSE and STD are shown in Figs. 2-4. (The results of Figs. 2-4 are the median value, the MSE and the STD of 100 tests, which the size of the population is set as 1000* *d* , where *d* is the number of dimensions and *d*=2, 3, 5, 10, 20, the iterations for EGFA is set as 20, for GA and PSO is set as 200*.) Fig. 2 shows that EGFA has better performance in accuracy than GA and PSO, especially in low-dimension search spaces. Fig. 3 shows that the solution EGFA finds has smaller error than the other two classical optimization algorithms. Fig. 4 demonstrates that EGFA has more stable performance compared with GA and PSO. In addition, the accuracy of solutions obtained by EGFA will decrease with the increase of the dimensions of search space like PSO, but EGFA has better accuracy than PSO as shown in Figs. 5-7.

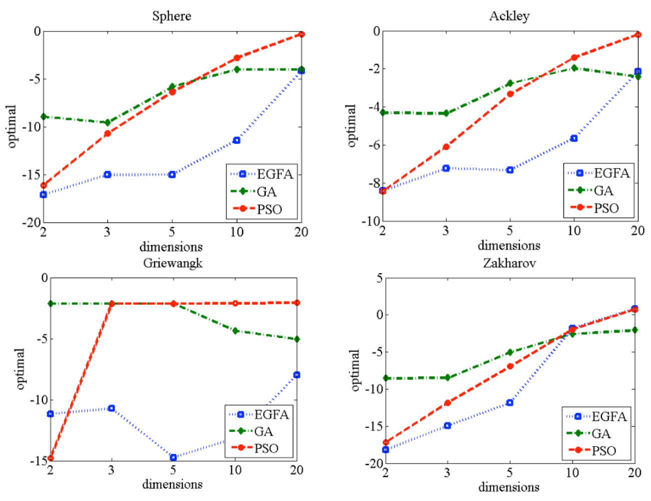
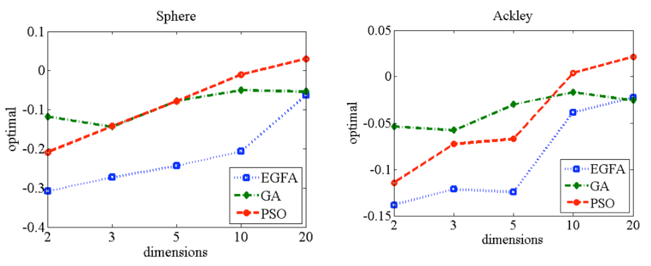


Figure 2. A Comparison of EGFA, GA and PSO in log10(median) with 2,3,5,10,20 dimensions



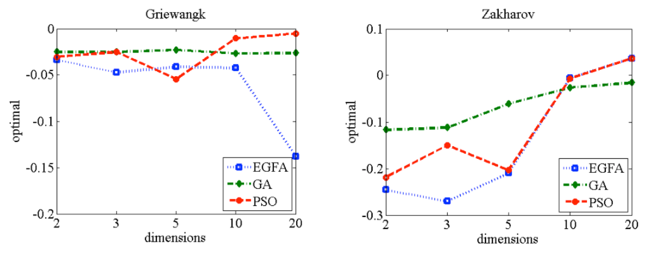


Figure 3. A Comparison of EGFA, GA and PSO in log10(MSE) with 2,3,5,10,20 dimensions

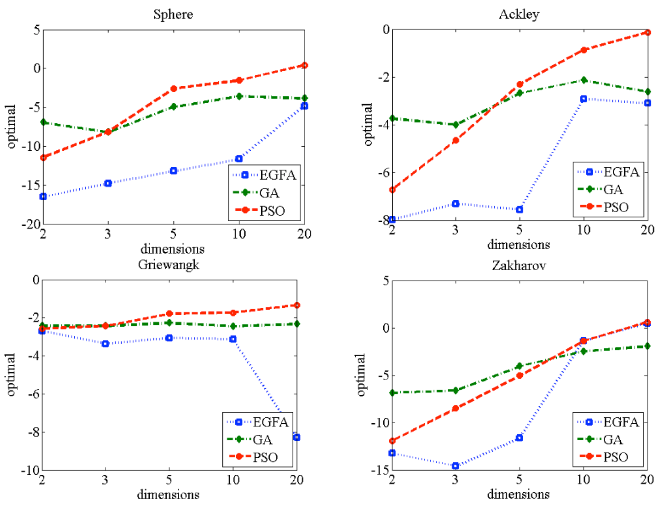
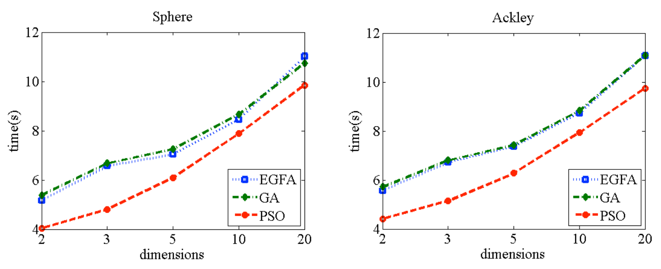
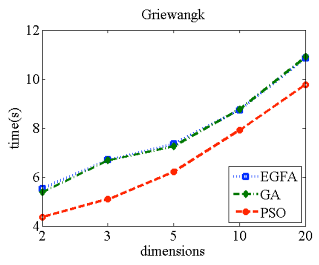


Figure 4. A Comparison of EGFA, GA and PSO in log10(STD) value with 2,3,5,10,20 dimensions

Besides the accuracy, the running time is another very important factor to measure the efficiency of an algorithm. It is obvious that the introduction of Optimal Detection is at cost of time, but the optimal space acquired by the Optimal Detection can help to decrease the number of iterations and shorten the overall algorithm running time. The total running time of three algorithms for 100 trials on three of the four test problems with the number of dimensions d=2, 3, 5, 10, 20 is reported in Fig. 5. As one can see that PSO is the fastest algorithm, but it has the lowest accuracy compared to EGFA and GA. Moreover, EGFA performs better than GA on these test problems with the same running time. Table III also shows that the running time of the three algorithms on the benchmark problems increases dramatically with the increase of the dimensions. All experiments were implemented on a PC (i5-4200M, 8GB, Windows 7, Matlab R2014a).





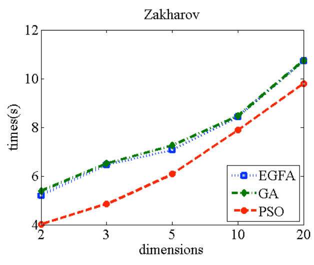


Figure 5. A Comparison of EGFA, GA and PSO in log10(STD) value with 2,3,5,10,20 dimensions

IV. Conclusion

In this research, a novel GFA-OD is presented based on the original GFA. A novel accuracy improvement strategy called Optimal Detection is employed to quickly identify the so-called optimal space that contains the optimal solution in search space with higher probability. All the four complex benchmark problems taken from literatures in unconstraint optimization are chosen to evaluate the performance of GFA-OD. The experimental results show that the proposed GFA-OD has better performance in accuracy, efficiency, and running time compared with GA and PSO. It is noted that the solutions which GFA-OD finds are frequently closer to the actual optimal solutions than the other two algorithms in lower dimension search spaces on all the four benchmark problems, but at the same time, we also notice the fact that all the three optimization algorithms face challenges on complex unconstraint optimization problems in higher dimensions both in terms of accuracy and running time, especially when the dimension is larger than 20, which needs us to make more efforts in future research. Our study of GFA-OD applied to complex unconstraint optimization problems in higher dimensions is in progress. Future research will include how to find more effective methods to adjust the parameters according to the information of specific problems.

References

[1] B. Chatterjee, Steepest Descent Method: Springer US, 2013.

[2] F. Ahmad, E. Tohidi, and J. A. Carrasco, "A parameterized multi-step Newton method for solving systems of nonlinear equations," Numerical Algorithms, vol. 71, pp. 1-23, 2016.

[3] J. Zhao, E. A. H. Vollebregt, and C. W. Oosterlee, "A fast nonlinear conjugate gradient based method for 3D concentrated frictional contact problems," Journal of Computational Physics, vol. 288, pp. 86-100, 2015.

[4] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, "A Stochastic Quasi-Newton Method for Large-Scale Optimization," Siam Journal on Optimization, vol. 26, 2015.

[5] M. A. Salido, J. Escamilla, A. Giret, and F. Barber, "A genetic algorithm for energy-efficiency in job-shop scheduling," The International Journal of Advanced Manufacturing Technology, vol. 85, pp. 1303-1314, 2016.

[6] E. Elyan and M. M. Gaber, "A Genetic Algorithm Approach to Optimising Random Forests Applied to Class Engineered Data," Information Sciences, 2016.

[7] T. Liao, K. Socha, M. A. M. D. Oca, and T. Stützle, "Ant Colony Optimization for Mixed-Variable Optimization Problems," IEEE Transactions on Evolutionary Computation, vol. 18, pp. 503-518, 2014.

[8] Z. Wang, H. Xing, T. Li, and Y. Yang, "A Modified Ant Colony Optimization Algorithm for Network Coding Resource Minimization," IEEE Transactions on Evolutionary Computation, vol. 20, pp. 1-1, 2015.

[9] J. Kennedy and R. Eberhart, Particle swarm optimization: Springer US, 2011.

[10] X. L. Wen, J. C. Huang, D. H. Sheng, and F. L. Wang, "Conicity and cylindricity error evaluation using particle swarm optimization," Precision Engineering, vol. 34, pp. 338-344, 2010.

[11] M. Zheng, G. Liu, C. Zhou, Y. Liang, and Y. Wang, "Gravitation field algorithm and its application in gene cluster," Algorithms for Molecular Biology, vol. 5, pp. 1-11, 2010.

[12] M. Zheng, Y. Sun, G. Liu, Y. Zhou, and C. Zhou, "Improved Gravitation Field Algorithm and Its Application in Hierarchical Clustering," Plos One, vol. 7, p. e49039, 2012.

[13] M. Zheng, G. X. Liu, Y. Zhou, and C. G. Zhou, "Reconstruction of gene regulatory network based on gravitation field algorithm," Jilin Daxue Xuebao, vol. 44, pp. 427-432, 2014.

[14] K. J. Walsh and A. Morbidelli, "The Effect of an Early Planetesimal-Driven Migration of the Giant Planets on Terrestrial Planet Formation," vol. 526, pp. 202-207, 2011.

[15] Yang and Xin ‐ She, "Appendix A: Test Problems in Optimization," Engineering Optimization, pp. 261-266, 2010.

[16] M. Laguna and R. Martí, "Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions," Journal of Global Optimization, vol. 33, pp. 235-255, 2005.

[17] G. Steenackers and P. Guillaume, "Bias-specified robust design optimization: A generalized mean squared error approach," Computers & Industrial Engineering, vol. 54, pp. 259-268, 2008.

[18] A. Majumder, "Application of Standard Deviation Method Integrated PSO Approach in Optimization of Manufacturing Process Parameters," Handbook of Research on Artificial Intelligence Techniques & Algorithms, 2015.