Study on Evolutionary Algorithm Online Performance Evaluation Visualization Based on Python Programming Language

Ruifeng SHI

School of Control & Computer Engineering, North China Electric Power University, Beijing 102206
E-mail: shi.ruifeng@126.com

Ning ZHANG

School of Control & Computer Engineering, North China Electric Power University, Beijing 102206
E-mail: zhangning629@126.com

Runhai JIAO

School of Control & Computer Engineering, North China Electric Power University, Beijing 102206
E-mail: runhaijiao@ncepu.edu.cn

Zhenyu ZHOU

School of Electric & Electronics Engineering, North China Electric Power University, Beijing 102206
E-mail: zhouzhenyu@ncepu.edu.cn

Li ZHANG

High-Tech Research and Development Center, Ministry of Science and Technology, Beijing 100044
E-mail: janezL612@126.com

Abstract Evolutionary computations are kinds of random searching algorithms derived from natural selection and biological genetic evolution behavior. Evaluating the performance of an algorithm is a fundamental task to track and find the way to improve the algorithm, while visualization technique may play an important act during the process. Based on current existing algorithm performance evaluation criteria and methods, a Python-based programming tracking strategy, which employs 2-D graphical library of python matplotlib for online algorithm performance evaluation, is proposed in this paper. Tracking and displaying the performance of genetic algorithm (GA) and particle swarm optimization (PSO) optimizing two typical numerical benchmark problems are employed for verification and validation. Results show that the tracking strategy based on Python language for online performance evaluation of evolutionary algorithms is valid, and can be used to help researchers on algorithms' performance evaluation and finding ways to improve it.

Keywords evolutionary algorithm; online performance evaluation; Python language; genetic algorithm; particle swarm optimization

---

Received November 27, 2013, accepted December 26, 2013

Supported by the Natural Science Foundation of China (Grant No. 61203100) and the Fundamental Research Funds for the Central Universities (Grant No. 13MS19)

1 Introduction

Evolutionary computations are kinds of modern intelligent optimization methods. The concepts of genetic evolution, mutation, natural selection and hybridization are derived from evolutionary biology. By simulating the law and characteristics of the process, the problems to be solved are translated into individual fitness improvement. With highly parallel, random and adaptive optimization, the population can converge to the “best adapted to their environment” individual eventually. Evolutionary algorithm includes not only the genetic algorithm in the traditional sense, but also the other algorithms with intelligent behaviours simulating organic groups’ foraging, such as particle swarm optimization (PSO), ant colony algorithm (AC), etc. How to properly evaluate the performance of these algorithms is still one of the hot topics in the study^[1].

The performance of a genetic algorithm mainly refers to its search efficiency and convergence, it is usually measured by evaluating the fitness of an algorithm to another for comparison study of different algorithms^[2]. De Jong using the on-line and off-line performance to evaluate the performance of GA in his early study: off-line performance is used to measure the algorithm convergence, on-line performance is used to measure the algorithm dynamic performance^[3]. At the same time, De Jong further pointed out, the on-line and off-line performance evaluate the algorithm performance through only one result of run, then comparing two strategies. But because of random oriented probability search essence of GA, the rationality of using single optimization result to evaluate performance of algorithm was questioned. Most scholars think that the influence of random factors on search performance of the algorithm should be considered in the evaluation criterion^[4]. Sun proposed one kind of performance evaluation criteria for GA based on “the average cut-off generation” and “the cut-off generation distribution entropy”^[5]. Cut-off generation is the population evolution generation that the search results of GA reach the required precision. The average cut-off generation is the mean of cut-off generations that the algorithm runs N times. The cut-off generation distribution entropy is the information entropy of these cut-off generations: though this criterion can make quantitative evaluation on the optimization efficiency of GA, well considering influence of random factors on the algorithm performance, but using this method for evaluation must know in advance the theory optimal solution of the problem in order to calculate the required evolution generation (cut-off generation) when the search results can meet the required precision. And these limit the application and population of this method. Gao et al. proposed a performance evaluation criteria for GA one kind based on the concept of average deviation distance and standard deviation of the deviating distance^[6], according to statistical parameters of optimization results for evaluating the algorithm performance to eliminate the effect of random factors on the results of the algorithm optimization.

The theoretical analysis of PSO algorithm performance has been one of the difficulties in this algorithm research. At present, most researchers are through large-scale computing on the algorithm performance to carry out statistical analysis. The difficulty of PSO algorithm theoretical analysis is the introduction of random variables in PSO, thus limiting the application of many mathematical analysis methods. Maurice conducted analysis preliminarily on the iterative formula of PSO^[7]. Van den Bergh^[8] improved the algorithm convergence analysis

based on Maurice et al. in 2001.

Compared to single objective evolutionary algorithm, performance evaluation of multi-objective evolutionary algorithm includes two aspects: that the algorithm solutions approximate the real non-inferior frontier and the solutions number. Thus it is difficult to evaluate the algorithm performance properly through a direct index. Scholars specially designed generation distance (GD), convergence index, diversity index δ and other indirect indices to evaluate the algorithm's performance^[9–11].

Previous studies mostly established the algorithm research platform, packaging the algorithm general framework and operator code to simplify operation in order to facilitate the use of researchers^[12–13], but the performance of optimization algorithms is not discussed. Based on the tracking evaluation research on the existing evolutionary algorithm, this paper presents graphical display method for tracking evolutionary algorithm online performance based on Python programming language. Through example analysis for tracking online performance of two typical evolution algorithms, the feasibility of this method is validated preliminarily.

2 Online graphical performance exploitation with Python programming language

The benefit of Python programming language lies in its merits of simple, easy, free, open source, portability, interpreted, object-oriented, scalability, embedded rich library, standard code and a series of advantages. Tracking and displaying online performance of an evolutionary algorithm with Python programming language is feasible and potentially high efficiency. The *Matplotlib* drawing database of Python programming language is introduced to draw the EA algorithms performance with different parameter indices.

Matplotlib is a pure independent Python library, which is similar to Matlab. *Matplotlib* is the most excellent graphics library, and its function is perfect. Its style is very similar to Matlab, but also inherited simple and clear style of Python. It is convenient to design and output of 2D and 3D data, which provides the conventional Descartes coordinates, polar coordinate, Spherical coordinate, 3D coordinate. The drawing aim is to appear the function with image, so we have to deal with two things using Python, one is the function and another is the image. The part of function uses *Numpy* which is three party libraries in *Matplotlib* use. *Numpy* is powerful and easy to use.

Matplotlib can conveniently draw scatter fine figure. When drawing scatter diagram, *Matplotlib* library is referenced with sentence of “import pylab” first, and then the scatter diagram is drawn with the function of scatter.

3 Framework of performance tracking programming for two typical evolutionary algorithms

Based on the understanding of the *Matplotlib*, in order to realize the tracking and display for online performance of GA and PSO algorithms, we review the basic principle of the two typical algorithms; and the frame of tracking and display for online performance based on Python is inserted into the procedure of the two algorithms; then the pseudo-code of this process is shown for further study.

3.1 Program design of tracking and displaying for GA online performance

3.1.1 A brief introduction to genetic algorithm

Genetic algorithm introduces the survival of the fittest evolutionary idea into the string structure with organized and random information exchange. With the algorithm from generation to generation, the individual continues combination, resulting in a better individual. Thus good characteristics are preserved, and differential characteristics are constantly out. The individuals of new generation in general are better than the old individuals, and at the same time the overall performance of the group is improved continuously, ultimately achieving the optimal value^[11].

3.1.2 Flowchart for genetic algorithm online performance tracking program

The flowchart of a genetic algorithm performance tracking program with Python can be illustrated as Figure 1.

Figure 1 GA program flow chart of tracking for online performance based on Python

In the above algorithm flow, tracking for online performance of this paper is mainly reflected in the block diagram on the right, which is embedded in the iterative process. For online tracking process, related variables of current objective function values are calculated and stored in real-time to show the tracking for algorithm dynamic process.

3.1.3 Pseudo-code for GA online performance tracking with Python

Natural evolution model is used in GA, the pseudo-code of the program are as shown as Figure 2.

On the basic of the complete genetic algorithm program, the program of online tracking is embedded in it. After calculating the individual objective function value, the average value, best value and worst value of all individuals objective function for per generation are calculated,

and then these three values are stored respectively in array *obj_average*[], *obj_best*[], *obj_worst*[]. Finally they are stored together in array *obj_record*[]. Then drawing online tracking diagram, the points of the array are drawn with different colors and symbols. They are stored in file “online tracking diagram.pdf”. Tracking and diagram display program can be shown as Figure 3.

```

Procedure  GA
Begin
    Initialize  P(0)
    t=0
    while (t<MaxDT) :
        for i in range(Popsize):
            Evaluate fitness of P(t)
        for i in range(Popsize):
            Select operation to P(t)
        for i in range(0,Popsize,2):
            Crossover operation to P(t)
        for i in range(Popsize):
            Mutation operation to P(t)
        for i in range(Popsize):
            P(t+1) = P(t)
        t=t+1
    end

```

Figure 2 The pseudo-code of GA based on Python

```

def tracking(self):
    #online tracking
    self.obj_best=self.obj[0]
    self.obj_worst=self.obj[0]
    self.obj_average=0
    for i in range(self.Popsize):
        if self.obj[i]<self.obj_best:
            self.obj_best=self.obj[i]
        if self.obj[i]>self.obj_worst:
            self.obj_worst=self.obj[i]
        self.obj_average+=self.obj[i]
    self.obj_average=self.obj_average/float(self.Popsize)
    self.obj_record.append([self.obj_best,self.obj_worst,self.obj_average])

```

Figure 3 The program of online tracking and diagram display

3.2 Program design of tracking and displaying for PSO online performance

3.2.1 The basic optimization principle of a PSO algorithm

Unlike GA, PSO algorithm does not have selection, crossover and mutation operations. Instead, each individual in the group is regarded as a particle with no mass and volume in the multi-dimensional search space. These particles fly with a certain speed in the search space, and the flight speed is dynamically adjusted based on itself and its companions flying experience. In other words, each particle constantly modifies its direction and speed by the statistics on itself and the group’s optimal value in the iterative process, thus forming positive feedback

mechanism of group optimization. The individuals gradually move to the better area based on the fitness of each particle on the environment for PSO, ultimately searching and finding the optimal solution.

3.2.2 Flowchart for PSO algorithm online performance tracking program

The flowchart of a genetic algorithm performance tracking program with Python can be illustrated as Figure 4.

Figure 4 Flow chart of python-based PSO algorithm for online performance evaluation

In the above algorithm flow, tracking for online performance of this paper is mainly reflected in the block diagram on the right, which is embedded in the iterative process. For online tracking process, related variables of current objective function values are calculated and stored in real-time to show the tracking for algorithm dynamic process.

3.2.3 Pseudo-code for PSO algorithm online performance tracking with Python

The pseudo-code of the basic PSO algorithm is shown as Figure 5.

```

begin
    initiation
    generation=0
    while(generation< MaxDT):
        for i in range(Swarmsize):
            for j in range(Dim):
                calculate Xid
                calculate Vid
            calculate fitness i of particle Xi
        update P, G, Pbest,Gbest
        generation+1
    end

```

Figure 5 The pseudo-code of PSO based on Python

Based on complete PSO program, the online tracking program is embedded in it. The embedding process is similar to GA in the last chapter.

4 Experimental case study

4.1 Problem description

In order to verify the effectiveness of the method that tracking and display for online performance of evolutionary algorithm proposed in this paper, we select two typical test examples^[13]. Examples are described as follows.

4.2 Experimental design

Control parameters setting issue is crucial in the process of evolutionary algorithm, having a great impact on the quality of the final solution. The operation performance of the algorithm is related with many factors, tracking for online performance proposed in this paper can well show the influence of parameters selection on the effectiveness of the optimization and convergence speed. The experimental design of this article is divided into two parts: the first part of the experiment is used to implement the crosswise comparison of algorithm performance between GA and PSO. Under the same set of parameters, the best, worst and average values of the objective function for every generation in the algorithm (GA and PSO) evolution are tracked and displayed, in order to achieve crosswise online tracking. The intermediate optimization process of GA and PSO is initially showed, visually observing the evolutionary trajectory of the two algorithms. And then the performance of convergence speed and others are compared. Parameter settings are show in Table 1.

Table 1 The algorithm parameters setting for case study						
		$PopSize$	$MaxGens$	$LenGens$	P_{Cross}	P_{Mutate}
GA	Case 1	400	100	10	0.8	0.08
GA	Case 2	800	100	30	0.8	0.08
		$PopSize$	$MaxGens$	$LearnFactor_1$	$LearnFactor_2$	$WeightInertia$
PSO	Case 1	80	60	2.1	1.9	0.6
PSO	Case 2	80	60	2.1	1.9	0.6

The second part is used to realize the effect of the same algorithm with different parameter combinations, verifying that carefully selected combination of parameters can greatly improve the optimization result and approving ‘no free lunch’ conclusion in the field of optimization. For example 1, in different parameter combinations, the best objective function value for every generation of the same algorithm (take PSO for example) evolutionary process are tracked and displayed, in order to achieve online tracking for longitudinal comparison. The influence of parameter variation on the algorithm iterative process is directly showed through different evolution parameter combinations, in order to find the best parameter combination. Parameter settings are as shown in Table 2.

Table 2 The algorithm parameters setting study for PSO

	$PopSize$	$MaxGens$	$LearnFactor_1$	$LearnFactor_2$	$Weight_{Inertia}$
Parameters Setting Scheme 1	50	70	2.1	1.9	0.6
Parameters Setting Scheme 2	90	50	2.1	1.9	0.6
Parameters Setting Scheme 3	130	30	2.1	1.9	0.6

4.3 Result analysis

4.3.1 The algorithm performance difference with regarding to the same set of parameters

In the case of the parameter settings in Table 1, the optimization results are shown in Figure 6~9, where the horizontal axis represents the algorithm evolution generation, and the vertical axis represents the objective function value.

Figure 6 Results display for example 1 of GA **Figure 7** Results display for example 2 of GA

Figure 8 Results display for example 1 of PSO

Figure 9 Results display for example 2 of PSO

Comparing these two algorithms performance indicators in optimization process, it is not difficult to find the similarities and differences of optimization process for GA and PSO:

1. Similarity

With the evolution generations increasing, the general trend of the best objective function value for per generation tends to decrease, and eventually converging to the respective optimal values. It shows that with the increase of the algorithm evolution generation, the performance of each individual constantly improved, eventually converging to the optimal value.

2. Differences

(1) With the increase of the evolution generation for PSO, the best, worst and average values of objective function eventually converge to the true optimal value; however three curves of GA are not convergence. This is because GA introduces fitness balancing strategy to make the

relative inferior solution obtain a certain amount of survival chance in order to avoid falling into the algorithm precocious: on one hand, it can avoid that individual good genetic information spread too quickly in the evolutionary population, and losing population diversity to lead to premature convergence; on the other hand, it will cause the decline of the algorithm convergence.

(2) The best objective function value for per generation fluctuates in PSO optimization, while GA's is relatively stable. This is because GA puts to use elitist strategy in the iteration process, and will not cause the degenerate of the optimization solution; while PSO is only learning part of the local optimum and global optimum for the current generation, existing blindness search, which leads to results bias.

(3) Compared with PSO, GA has a better convergent rate at its early evolutionary process, while its later process is respectively slow. This indicates that: the performance of GA in the global search is excellent, but the local search ability is poor. So its convergence speed in later evolutionary is low.

4.3.2 Analysis of the algorithm performance with different parameters combination

In the case of the parameter setting in Table 2, the results are shown in Figure 10.

Figure 10 Results display with different parameters combination of PSO

Different optimization efficiency and optimization results are obtained with three different groups of parameter combinations for PSO: the population of the tenth generation converges to the optimal value for the third group of parameters; the population of more than twentieth generation converges to the optimal value for the second group of parameter; while the population converges to the optimal value until more than thirtieth generation. It shows: for test function 1, convergence speed and population scale are approximately linearly proportional relationship, while the role of evolution generation on the convergence speed is relatively small. Thus the sensitivity of breadth search priority is higher than the sensitivity of depth search priority for this problem. And consequently in the same computational overhead, the combination of larger population size and smaller evolution generation can obtain optimum results.

5 Conclusion

In this paper, through online tracking and display strategy of evolutionary algorithm based on Python language, a comparative study is conducted on the performance of two typical evolutionary algorithms (GA and PSO algorithm). The module of tracking for online performance designed in this paper is verified through two examples. How to promote the above method to online performance tracking of multi-objective evolutionary algorithm, and enhancing the interaction of parameter adjustment and user-friendliness is to be further study.

References

- [1] Wang X, Wang S. Modern intelligent information processing method in practice. Beijing: Tsinghua University Press, 2009.
- [2] Hou G, Wu C. Performance analysis of genetic algorithm. *Control & Decision*, 1999, 5(12): 66–69.
- [3] Li M, Kou J. Coordinate multi-population genetic algorithm for multi-modal function optimization. *Acta Automatica Sinica*, 2002, 28(4): 497–504.
- [4] Goldberg D E. Genetic algorithm in search, optimization and machine learning. Addison-Wesley, Reading, MA, 1989.
- [5] Kong X, Qu L. Quantitative evaluation of genetic algorithm optimization efficiency. *Journal of Automation*, 2009, 26(4): 552–556.
- [6] Gao Q, Lü W, Du X, et al. Research on evaluation criteria for optimization performance of genetic algorithm. *Journal of Xi'an Jiaotong University*, 2006, 7(14): 803–806.
- [7] Maurice C, Kennedy J. The particle swarm — Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 2002, 6(1): 58–73.
- [8] Van Den Bergh F. An analysis of particle swarm optimizers. PhD Thesis: Department of Computer Science, University of Pretoria, South Africa, 2002.
- [9] Lei D, Yan X. Multi-objective intelligent optimization algorithm and its application. Beijing: Science Press, 2009.
- [10] Zheng J. Multi-objective evolutionary algorithm and its application. Beijing: Science Press, 2007.
- [11] Li M, Zheng J. A evaluation method for solution set distribution scope of multi-objective evolutionary algorithm. *Chinese Journal of Computers*, 2011, 34(4): 647–664.
- [12] Kong L. Object-oriented genetic algorithm platform design and application. Shanghai: Shanghai Jiaotong University, 2008.
- [13] Xu X, Li Y, Wu Y, et al. Particle swarm optimization algorithm platform design based on strategy pattern. *Journal of Wuhan University*, 2010, 43(3): 361–365.