A Cooperative Particle Swarm Optimization Algorithm Based on Greedy Disturbance

Xing Huo[1], Fei Zhang[1], Chao Luo[1], Jieqing Tan[1], and Kun Shao[2(✉)]

[1] School of Mathematics, Hefei University of Technology, Hefei, China
[2] School of Software, Hefei University of Technology, Hefei, China
zzzf0214@163.com

Abstract. In this paper, an improved Particle Swarm Optimization Algorithm (GCPSO) is proposed to solve the shortcomings of the existing Particle Swarm Optimization Algorithm (PSO) which has low convergence precision, slow convergence rate and is easy to fall into local optimum when performing high-dimensional optimization in the late iteration. First, the whole particle swarm of the algorithm was divided into three sub-groups, and different ranges of inertia weight ω are set for balances global search and local search in each sub-group, which improves the algorithm's ability to explore. Then we add Gaussian perturbation with the greedy strategy to PSO to avoid the algorithm falling into local optimum and improve the convergence speed. And finally, the proposed algorithm is compared with Genetic Algorithm (GA), PSO and Grasshopper Optimization Algorithm (GOA) to analyse its performance and speed. Through experimental analysis, GCPSO has a significant improvement at convergence speed, convergence accuracy and stability.

Keywords: Optimization algorithm · Greedy strategy · PSO · GA · GOA

1 Introduction

Optimization problem often appears in scientific research, and many engineering problems ultimately boil down to optimization problems. It can be expressed as a mathematical problem. It generally refers to the question of how to find a specific factor (variable) under a given constraint to make the target reach the optimal value. The optimization algorithm is used to solve the optimization problem, and the objective function of the optimization problem is established as an optimization model to obtain the optimal value. For the more complex optimization problems of non-linear, multi-dimensional and global optimization, traditional optimization algorithms have been difficult to meet the needs. And various intelligent optimization algorithms that are inspired by bionics have a better solution in the complex optimization problem such as Genetic Algorithm (GA), Particle Swarm Optimization Algorithm (PSO), and Grasshopper Optimization Algorithm (GOA). And they have attracted the attention of many scholars. Because of the high efficiency and strong convergence of this kind of algorithm, more and more scholars have applied it to their respective fields and achieved good results [1–3]. Genetic Algorithm (GA) is a meta-heuristic algorithm proposed by Professor Holland [4] in 1975. Its principle on Darwin's evolutionary

Z. Lin et al. (Eds.): PRCV 2019, LNCS 11859, pp. 52–61, 2019.
https://doi.org/10.1007/978-3-030-31726-3_5

theory of survival of the fittest. Genetic Algorithm takes all individuals in a group as variable objects and represents the gene sequence in binary code form. The algorithm searches for the optimal value within the range of the coded variables through the genetic operations of selection, crossover and mutation, which retain the excellent individuals and eliminate the poor individuals, and then form a new population. And the optimal solution is obtained after repeated iterations. However, the convergence efficiency of GA is low, and it is easy to converge prematurely. Particle Swarm Optimization (PSO) is another metaheuristic algorithm proposed by Kennedy et al. [5]. This algorithm simulates the predation behaviour of a flock of birds. The solution of the optimization problem is compared to a bird in the search space which called "particles". And all particles are searched in the space of the variable range, and the fitness value is calculated by the optimized function to determine the distance of the current location to the food. The algorithm finds the global optimal by updating the individual historical optimal value and the overall population optimal value. The original PSO algorithm has the disadvantages of slow convergence speed and low convergence precision. In order to balance the local search ability and global search ability in the original PSO algorithm, Shi et al. [6] proposed an improved algorithm with inertia weight ω to adjust convergence and convergence speed dynamically which is called the standard PSO algorithm (For the convenience of description, PSO refers to the standard PSO algorithm in this paper). However, there are two problems with the algorithms: the algorithm is easy to fall into local optimization and has poor convergence precision when performing high-dimensional optimization; the convergence efficiency is low when entering the late iteration. For this reason, Li et al. [7] proposed an efficient and improved particle swarm optimization strategy, which divides the whole population into several sub-groups for the division of labor and information exchange to improve the local search ability and global search ability of the algorithm. Grasshopper Optimization Algorithm (GOA) is a new meta-heuristic algorithm proposed by Salemii et al. [8] in 2017. The basic idea is based on the regularity of grasshopper cluster activities and the model of group intelligence activities. The influencing factors are wind direction, gravity, effects of other grasshoppers in the population and the optimal position reached in the current population. However, GOA is not only easy to fall into local optimum but also has high design complexity and time-consuming. Therefore, it is necessary to improve the algorithm to get a better algorithm. And there are also some other metaheuristics such as Grey Wolf Optimization Algorithms [9], Whale Optimization Algorithms [10], etc.

Based on the work of Li, this paper improves PSO. The improved algorithm (GCPSO) uses the variation of inertia weight and adds a Gaussian perturbation strategy based on greedy thought to make the particles maintain strong vitality during the evolution process. The algorithm was carried out on the benchmark functions and compared with other intelligent optimization algorithms. Experiments show that GCPSO has a significant improvement at convergence speed, convergence accuracy and stability.

2 GCPSO Algorithm

Co-evolutionary algorithm establishes two or more populations to establish competition or cooperation between them [11]. Each population enhances its performance through its iterative strategy and interaction to achieve the purpose of population optimization. Traditional particle swarm optimization algorithm uses a single group iterative strategy. The algorithm has slow convergence speed and is easy to fall into local optimum when dealing with high-dimensional complex functions so that the satisfactory results cannot be obtained. This paper draws on the division strategy idea of the co-evolutionary algorithm, combines Gaussian perturbation strategy based on greedy thought [12], and proposes an algorithm (GCPSO) with the cooperative division of labor based on greedy disturbance. The algorithm effectively compensates for the defect. GCPSO is described as follows:

The whole particle swarm is divided into three subgroups: S1, S2 and S3. Each subgroup has different iterative strategies. The subgroup S1 adopts the traditional standard PSO iterative strategy, and the subgroup S2 adopts the global search to enhance the strategy gradually. The subgroup S3 only uses the "social experience" part, which considers the information sharing and cooperation between particles. Let x_i refers to the coordinate position of the particle i in the particle group, and v_i is the corresponding velocity, c_1 and c_2 are constant named learning factors, r_1 and r_2 are uniform random numbers between [0, 1]; $pbest_i$ is the individual historical optimum of the particle i and gbest is the global best value of the particle swarm. Let t be the current number of iterations and T be the maximum number of iterations. Then the iteration formula for each group is formulated as follows:

$$\text{Population S1}: \quad v_i = \omega_1 v_i + c_1 r_1 (pbest_i - x_i) + c_2 r_2 (gbest - x_i) \tag{1}$$

$$\text{Where,} \quad \omega_1 = \omega_{1max} - t * \frac{\omega_{1max} - \omega_{1min}}{T} \tag{2}$$

$$\text{Population S2} \quad v_i = \omega_2 v_i + c_1 r_1 (pbest_i - x_i) + c_2 r_2 (gbest - x_i) \tag{3}$$

$$\text{Where,} \quad \omega_2 = \omega_{2min} + t * \frac{\omega_{2max} - \omega_{2min}}{T} \tag{4}$$

$$\text{Population S3} \quad v_i = \omega_3 v_i + c_2 r_2 (gbest - x_i) \tag{5}$$

$$\text{Where,} \quad \omega_3 = \frac{\omega_1 + \omega_2}{2} \tag{6}$$

Among them, ω_1, ω_2, and ω_3 are iterative weights, ω_{1max}, ω_{1min} and ω_{2max}, ω_{2min} are the maximum and minimum values of the iterative weights, respectively. The larger iterative weight has better exploration ability and global convergence ability, while the smaller iterative weight makes stronger local convergence ability in the later stage, which can get more accurate results. At the same time, to prevent the algorithm from crossing the boundary, the boundary values v_{min} and v_{max} are set for the above velocity term.

The coordinate position x_i of the particle i in each of the above subgroups is updated by the formula:

$$x_i = x_i + v_i \tag{7}$$

To further avoid the algorithm falling into local optimum, a Gaussian perturbation is added to the global best position of the particles:

$$\text{gbest} = \text{gbest} * (c + \text{gau}) \tag{8}$$

Where gau represents White Gaussian Noise, and c represents the interference factor, which is a constant.

To increase the convergence rate, we add the greedy strategy idea and iterate multiple times in the Gaussian perturbation process:

$$\begin{aligned}
&\text{for } i = 1 : M \\
&\quad \text{gbest_temp} = \text{gbest} \times (a + \text{gau}) \\
&\quad \text{gbest_temp_fit} = f(\text{gbest_temp}); \\
&\quad \text{if } \text{gbest_temp_fit} < \text{gbest_fit} \\
&\qquad \text{gbest} = \text{gbest_temp}; \\
&\quad \text{end} \\
&\text{end}
\end{aligned} \tag{9}$$

Where M represents the maximum number of iterations and f represents the function to be optimized, $\text{gbest\_fit} = f(\text{gbest})$.

The working principle of GCPSO is to divide the whole group into several subgroups and assign different evolution strategies to different sub-groups. Different subgroups exchange information by sharing global best information gbest to complete group collaboration and accelerate the convergence speed. And Gaussian perturbation with greedy thought is added to prevent local optimum and achieve fast and accurate convergence. In GCPSO, the subgroup S1 is iterated according to the standard PSO. And the inertia weight value ω_1 of S1 is linearly decreased, representing particle optimization gradually evolves from the strong global convergence at the early stage to the strong local convergence at the later stage, and the accurate convergence results are obtained. The inertia weight value ω_2 of the subgroup S2 is linearly increased to improve the global search capabilities of whole particle swarm and avoid the local convergence of S1 in the later stage of the algorithm. The subgroup S3 only contains the "social experience" part, that is, it only searches near the current optimal position so that it can quickly converge to the current optimal position.

At the same time, to improve the convergence rate and further avoid the algorithm falling into local optimum, the disturbance with the thought of greedy strategy is added. GCPSO improves the efficiency and accuracy of optimization through the divide-and-conquer strategy of cooperative thinking and greedy disturbances. The pseudo code for GCPSO is given in Table 1.

Table 1. The procedure of GCPSO.

Procedure:

Input: Iterator times: T; Dimension: D; Three population sizes; Disturbance times: M

Output: The global best particle's position $gbest$ and corresponding function value $fbest$

Initialize each particle i's position $x1_i$ and speed $v1_i$ in swarm1;

Initialize each particle j's position $x2_j$ and speed $v2_j$ in swarm2;

Initialize each particle l's position $x3_l$ and speed $v3_l$ in swarm3;

Set speed boundary v_{max} and v_{min}, inertia weight boundary $\omega_{1\min}$, $\omega_{1\max}$ and $\omega_{2\min}$, $\omega_{2\max}$;

Calculate the fitness of each particle;

Set $pbest1_i = x1_i$, $pbest2_j = x2_j$, $pbest_l = x3_l$ for each particle in three populations;

Update the $gbest$ position of all particles in three populations;

While $(t \le T)$

Set inertia weight ω_1 using Eq.(2);

For each particle in swarm1

update the speed formula by the Eq.(1);

update the position of the current particle by the Eq.(7);

End For

Calculate the fitness of each particle in swarm1;

Update the position $pbest1$ for each particle in swarm1;

Set inertia weight ω_2 using Eq.(4);

For each particle in swarm2

update the speed formula by the Eq.(3);

update the position of the current particle by the Eq.(7);

End For

Calculate the fitness of each particle in swarm2;

Update the position $pbest2$ for each particle in swarm2;

Set inertia weight ω_3 using Eq.(6);

For each particle in swarm3

update the speed formula by the Eq.(5);

update the position of the current particle by the Eq.(7);

End For

Calculate the fitness of each particle in swarm3;

Update the position $pbest3$ for each particle in swarm3;

Update the position $gbest$ of all particles in three populations;

For k=1 to M

Update the $gbest$ position by the operation (9);

End For

$t \leftarrow t+1$;

End While

Return $gbest$ and $fbest$.

3 Experiments and Analysis

In this section, we focused on the effect of the improved particle swarm optimization algorithm in global optimization. Ten classical benchmark functions [13, 14] in Table 2 are used to test the algorithm. The functions $f_1 - f_6$ are unimodal functions, and the functions $f_7 - f_{11}$ are multimodal functions. The expressions and parameters of the functions are shown in Table 2. Dim represents the dimension of the function, and Range represents the range of values of each variable of the function, f_{min} represents the minimum value of the function, and D represents the dimension of the function f_7.

Table 2. Description of benchmark functions.

Fun	Dim	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100, 100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10, 10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	30	[100, 100]	0
$f_4(x) = \max_i \{ \lvert x_i \rvert ,\ \ 1 \le i \le n \}$	30	[−100, 100]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[−30, 30]	0
$f_6(x) = \sum_{i=1}^{n} i x_i^4 + random[0\ ,\ 1)$	30	[−1.28, 1.28]	0
$f_7(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[−500, 500]	−418.9829*D
$f_8(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[−5.12, 5.12]	0
$f_9(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$	30	[−32, 32]	0
$f_{10}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[−600, 600]	0
$f_{11}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\}$ $+ \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ where $y_i = 1 + \frac{x_i + 1}{4}$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	30	[−50, 50]	0

The improved algorithm-GCPSO is compared with PSO, GA and GOA in function optimization experiments. For comparing the experimental performance of each algorithm quantitatively, the maximum number of iterations is set as 1000 in the experiment. The experimental parameters of each algorithm are given in Table 3.

Table 3. Parameter settings.

Algorithm	Parameter	Parameter value
GCPSO	Perturbation times: M	10
	Interference factor: a	0.5
	ω_{1min}, ω_{1max}	0.001, 0.9
	ω_{2min}, ω_{2max}	0.001, 0.9
	v_{min}, v_{max}	−4, 4
GA	Swarm size	300
	Crossover probability Mutation probability	0.6 0.001
PSO	Swarm size	300
	c_1, c_2	2, 2
	ω_{min}, ω_{max}	0.4, 0.9
	v_{min}, v_{max}	−4, 4
GOA	Swarm size	300
	c_{min}, c_{max}	0.00004, 1

3.1 Quality Analysis of Solutions

Table 4 shows the performance of GPSO, PSO, GA and GOA on different benchmark functions. F denotes the benchmark function, ave denotes the average optimal value of the function, std denotes the average standard deviation of the function value, and tim denotes the average running time of the algorithm on the function. And each benchmark function was run many times to generate these statistical results. The dimension of the experimental search space is 30-dimensional, and the population size is set to 300. The improved particle swarm optimization algorithm-GCPSO contains 100 particles per subpopulation, and each test function was run 30 times independently.

From the Table 4, we can see that the proposed algorithm-GCPSO takes a little shorter time than other algorithms except for PSO, but the mean value of the function is closer to the theoretical value than PSO. It indicates that GCPSO has advantages in solving high-dimensional function problems. It effectively solves the problem of poor convergence and local optimum of PSO in the later iteration period and improves the accuracy of the solution. At the same time, GCPSO has lower average standard deviation than PSO, which indicates that GCPSO improves the stability of the original algorithm. Comparing GCPSO with GA, we can see that GCPSO has better results in mean, standard deviation and running time for all functions except for a slight difference in the optimization of function f_{11}. This shows that the GCPSO proposed in this paper is much better than GA in convergence, convergence accuracy, optimization speed and robustness. Compared with GOA, except for the function f_7, GCPSO is also in a leading position in three statistical parameters for benchmark functions: the average time-consuming is short, indicating that GCPSO optimization speed is faster; the mean value of the function is closer to the theoretical value, indicating that GCPSO has better global convergence and convergence accuracy; the average standard deviation is lower, indicating that GCPSO has higher robustness than GOA.

Table 4. Comparison of optimization results.

F	GCPSO			PSO			GA			GOA		
	ave	std	tim	ave	std	tim	ave	std	tim	ave	std	tim
f_1	**5.8805e −230**	0	0.6725	0.1054	0.1749	0.5453	0.4606	0.1857	1.1866	0.0095	0.0045	357.4484
f_2	**4.9256e −118**	2.6887e −117	0.7178	2.4504	1.3186	0.5789	0.2848	0.0713	1.4102	0.0371	0.0563	348.2115
f_3	**4.1432e −118**	0	2.8946	5.5406	2.9889	2.7797	9.9922e +03	3.2617e +03	3.0920	10.9614	4.8229	349.2198
f_4	**7.5103e −118**	3.6256e −117	0.6809	1.2578	0.6080	0.5478	2.3943	0.5729	0.9109	0.2287	0.1603	339.9627
f_5	**28.4772**	0.2550	0.9670	165.8761	138.1158	0.9180	162.0951	70.7509	1.1970	44.9997	34.5939	345.7606
f_6	**8.6464e −05**	9.4700e −05	1.8967	0.0676	0.0368	1.7553	0.0027	9.4723e −04	2.0853	0.0214	0.0042	341.6339
f_7	−5.3806e +03	639.7023	1.0790	−6.4752e +03	697.6042	0.9488	−3.0650e +03	421.7024	1.1954	**−6.6520e +03**	852.6289	339.2142
f_8	**0**	0	0.7485	55.5829	16.1783	0.7485	26.0967	11.6551	1.0943	36.1833	34.0773	345.8481
f_9	**8.8818e −16**	0	0.7693	2.1103	0.6317	0.7515	10.1765	6.8661	1.5543	0.6351	0.7783	350.7806
f_{10}	**0**	0	1.0285	0.2342	0.2348	0.9198	1.0558	0.0163	1.6603	0.0223	0.0116	341.1963
f_{11}	0.0206	0.0306	4.1695	1.8660	1.2255	3.9015	**6.7226e −04**	3.2511e −04	4.8457	0.5705	1.3724	361.6257

3.2 Convergence Analysis of the Algorithm

Figure 1 shows the convergence curves of the function values varying with the number of iterations on the benchmark functions under different optimization algorithms. And the convergence curves for the functions $f_1 - f_{11}$ are arranged in the order from left to right and top to bottom. In the lower right corner of the figure, the enlargement effect in the yellow frame is given to show more clearly.

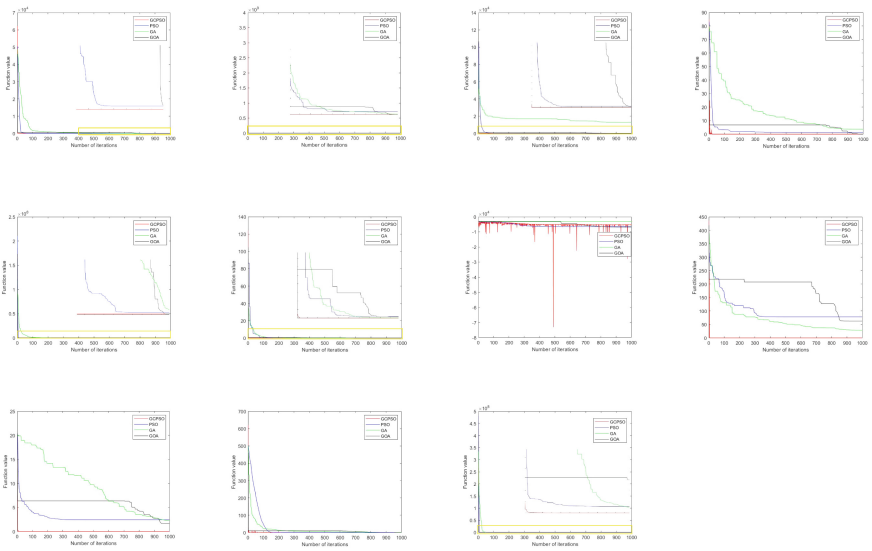

Fig. 1. Convergence curves.

It can be seen from the Fig. 1 that the final convergence values of GCPSO on the benchmark functions are the smallest except for the function f_7, which indicates that GCPSO has the best global convergence and high convergence precision, while the other algorithms fall into local convergence on the benchmark functions, resulting in low convergence and low convergence accuracy. Observing the convergence curves, GCPSO can reach the minimum in the number of iterations less than 50 on most functions, that is, the convergence rate is fast, and the PSO algorithm follows. This shows that the proposed algorithm-GCPSO not only improves the global search ability and convergence efficiency of PSO but also can find the optimal value more quickly. Compared with other algorithms, the proposed algorithm-GCPSO can converge to the optimal value stably and quickly, the global search ability of the algorithm is stronger, and the convergence results are better.

In conclusion, the proposed algorithm-GCPSO has the characteristics of high convergence efficiency, strong global convergence, high convergence accuracy and good robustness.

4 Conclusion

The algorithm-GCPSO proposed in this paper borrows the divide-and-conquer strategy of cooperative thinking to makes full use of the advantages of group division and cooperation. And the algorithm combines the perturbation based on the greedy strategy, which not only improves the convergence efficiency of the algorithm but also improves the convergence accuracy of the algorithm. Experiments of 11 benchmark functions show that GCPSO has great advantages in accuracy, speed and stability compared with other algorithms. Future research will focus on simplifying the initial parameters and more complex high-dimensional optimization problems to enhance the universality of the algorithm.

References

1. Hammouche, K., Diaf, M., Siarry, P.: A multilevel automatic thresholding method based on a genetic algorithm for a fast image segmentation. Comput. Vis. Image Underst. **109**(2), 163–175 (2008)
2. Sathya, P.D., Kayalvizhi, R.: PSO-based Tsallis thresholding selection procedure for image segmentation. Int. J. Comput. Appl. **5**(4), 39–46 (2010)
3. Sharma, A., Sharma, M., Rajneesh: SAR image segmentation using Grasshopper Optimization Algorithm. Int. J. Electron. **6**(12), 19–25 (2017)
4. Holland, J.H.: Genetic algorithms. Sci. Am. **267**(1), 66–73 (1992)
5. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
6. Shi, Y.H., Eberhart, R.C.: A modified particle swarm optimizer. In: Proceedings of the IEEE Conference on Evolutionary Computation, Anchorage, pp. 69–73 (1998)
7. Li, H.L., Hou, C.Z., Zhou, S.S.: High efficient algorithm of modified particle swarm optimization. Comput. Eng. Appl. **44**(1), 14–16 (2008)

8. Saremi, S., Mirjalili, S., Lewis, A.: Grasshopper optimization algorithm: theory and application. Adv. Eng. Softw. **105**, 30–47 (2017)
9. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
10. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
11. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_269
12. Pan, G., Li, K., Ouyang, A., et al.: Hybrid immune algorithm based on greedy algorithm and delete-cross operator for solving TSP. Soft. Comput. **20**(2), 555–566 (2016)
13. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)
14. Digalakis, J.G., Margaritis, K.G.: On benchmarking functions for genetic algorithms. Int. J. Comput. Math **77**(4), 481–506 (2001)