TELKOMNIKA, Vol. 11, No. 6, June 2013, pp. 2926 ~ 2932
e-ISSN: 2087-278X

Orthogonal Particle Swarm Optimization Algorithm and Its Application in Circuit Design

Xuesong Yan*[1], Qinghua Wu[2,3], Hammin Liu[4]
[1]School of Computer Science, China University of Geosciences, Wuhan, China
[2]Hubei Provincial Key Laboratory of Intelligent Robot, Wuhan Institute of Technology, Wuhan, China
[3]School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan, China
[4]Wuhan Institute of Shipbuilding Technology, Wuhan, China
*Corresponding author, e-mail: yanxs1999@126.com

Abstract

In this paper, aim at the disadvantages of standard Particle Swarm Optimization (PSO) algorithm like being trapped easily into a local optimum, we improves the standard PSO and proposes a new algorithm to solve the overcomes of the standard PSO. The new algorithm keeps not only the fast convergence speed characteristic of PSO, but effectively improves the capability of global searching as well. Experiment results reveal that the proposed algorithm can find better solutions when compared to the standard particle swarm optimization algorithm. We use the proposed algorithm for digital circuit optimization design, and the final circuit is optimized in terms of complexity (with the minimum number of gates).

Keywords*: particle swarm optimization, orthogonal design, benchmark function, circuit design*

1. Introduction

Swarm intelligence is an important research topic based on the collective behavior of decentralized and self-organized systems in computational intelligence. It consists of a population which simulates the animals behavior in the real world. Now there are many swarm intelligence optimization algorithms, such as genetic algorithms, particle swarm optimization, ant colony optimization, bee colony algorithm, differential evolution, fish-warm algorithm, etc. Due to the simple concept, easy implementation and quick convergence, PSO has gained much attention and been successfully applied in a variety of fields mainly for optimization problems.

Particle Swarm Optimization (PSO) algorithm was an intelligent technology first presented in 1995 by Eberhart and Kennedy, and it was developed under the inspiration of behavior laws of bird flocks, fish schools and human communities [1]. If we compare PSO with Genetic Algorithms (GAs), we may find that they are all maneuvered on the basis of population operated. But PSO doesn't rely on genetic operators like selection operators, crossover operators and mutation operators to operate individual, it optimizes the population through information exchange among individuals. PSO achieves its optimum solution by starting from a group of random solution and then searching repeatedly. Once PSO was presented, it invited widespread concerns among scholars in the optimization fields and shortly afterwards it had become a studying focus within only several years. A number of scientific achievements had emerged in these fields [2-4]. PSO was proved to be a sort of high efficient optimization algorithm by numerous research and experiments [5-8]. PSO is a meta-heuristic as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, meta-heuristics such as PSO do not guarantee an optimal solution is ever found. More specifically, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-Newton methods. PSO can therefore also be used on optimization problems that are partially irregular, noisy, change over time, etc. This paper improves the disadvantages of standard PSO being easily trapped into a local optimum and proposed a new algorithm which proves to be more simply conducted and with more efficient global searching capability.

Received January 7, 2013; Revised March 29, 2013; Accepted April 7, 2013

2. Particle Swarm Optimization Algorithm

The standard PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles). These particles are moved around in the search space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search space as well as the entire swarm's best known position. When improved positions are being discovered, they are used to guide the movements of the swarm. The process is repeated until that a satisfactory solution is discovered.

In PSO, each solution is regarded as a bird in the search space and called “particle”. Each particle has a fitness value which is determined by a target function. The status of each particle includes its position and velocity. Its velocity determines its flying direction. All particles (the swarm) work together in searching for the optimal solution in the solution space. PSO achieves the searches via updating the status of each particle. At the beginning, it randomly initiates a group of particles (random solutions) with a specific position and velocity for each. It updates the status (its velocity and position, referring Formula (1) and (2) of each particle using the recorded best position experienced for this particle (called P_{idb} in Formula 1) and the best position of the whole swarm (called P_{gdb} in Formula 1) until now. This updating procedure continues until it reaches the maximum number of iterations, which is set up at the beginning. Through this iteratively updating, PSO finds (or approaches to) optimal solutions in the solution space. Thus, the status of each particle is updated not only based on itself best position (P_{idb}) experienced, but also based on the best position of the whole swarm (its companions). That is, the particles inside the swarm share the information (P_{gdb}). Specifically, for a particle id, its velocity and its position is updated according to the formula as follows respectively:

$$V_{id}^{'} = \omega V_{id} + \eta_1 rand()(P_{idb} - X_{id}) + \eta_2 rand()(P_{gdb} - X_{id}) \quad (1)$$

$$X_{id}^{'} = X_{id} + V_{id}^{'} \quad (2)$$

where ω is called the inertia weight. It is a proportion factor concerned with former velocity ($0 < \omega < 1$). η_1 and η_2 are constant accelerating factors, normally $\eta_1 = \eta_2 = 2$. The random function $rand()$ is to generate random numbers. X_{id} represents the position of particle id . V_{id} represents the velocity of particle id . P_{idb} and P_{gdb} represent the best position of the particle id found and the best position of the whole swarm found respectively until this moment.

In the formula (1) above, the first part represents the impact of the former velocity of the particle. It enables the particle to possess expanding tendency in the searching space, and thus makes the algorithm be more capable in global searching. The second part is called a cognition part. It represents the process of absorbing individual experience knowledge of the particle. The third part is called a social part. It represents the process of learning from the experience of other particles. It also shows the information sharing and social cooperation among particles.

The most obvious advantage of PSO is that the convergence speed of the swarm is very high, scholars like Clerc [9] has presented proof on its convergence. Here a fatal weakness may result from this characteristic. With constant increase of iterations, the velocity of particles will gradually diminish and reach zero in the end. At this time, the whole swarm will be converged at one point in the solution space, if g_{best} particles haven't found g_{best}, the whole swarm will be trapped into a local optimum; and the capacity of swarm jump out of a local optimum is rather weak. The probability of the occurrence is especially high so far for multi-peaks functions, we have test the algorithm for the multi-peaks benchmark functions to verify these. We select three multi-peaks functions for our experiment, the functions described as following and Table 1 is the experiment results.

Table 1. Experiment Results

Function	Best Value	Mean Value	Worst Value	$f_{\min}$
F1	1495.71	4224.775	7032.89	0
F2	72.5069	101.410452	123.954	0
F3	-5038.62	-4005.02	-3233.13	-12569.5

$$F1: f(x) = \sum_{i=1}^{n} x_i^2, x_i \in (-100,100), f_{min} = 0$$

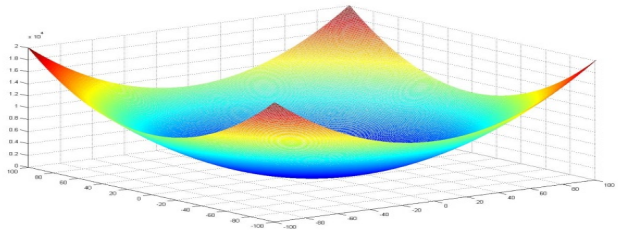

Figure 1. Benchmark Function F1

$$F2: f(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos(\frac{x_i - 100}{\sqrt{i}}) + 1, x_i \in (-300,300), f_{min} = 0$$

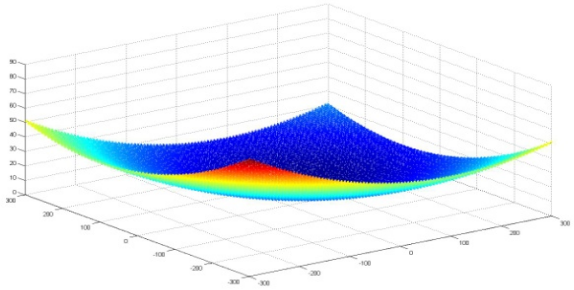

Figure 2. Benchmark Function F2

$$F3: f(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}), x_i \in (-500,500), f_{min} = -12569.5$$

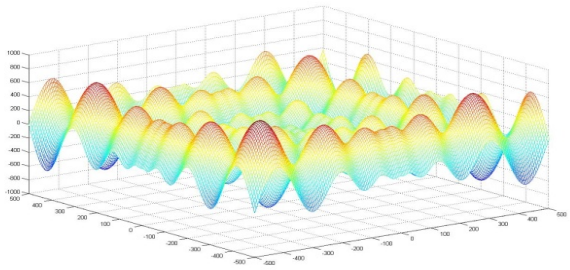

Figure 3. Benchmark Function F3

3. Orthogonal Particle Swarm Optimization Algorithm

3.1. Orthogonal Initialization

The traditional method of particle swarm optimization algorithm is randomly initialized population, that is, generate a series of random numbers in the solution space of the question. Design the new algorithm, we using the orthogonal initialization [10-12] in the initialization phase. For the general condition, before seeking out the optimal solution the location of the global optimal solution is impossible to know, for some high-dimensional and multi-mode functions to optimize, the function itself has a lot of poles, and the global optimum location of the function is unknown. If the initial population of chromosomes can be evenly distributed in the feasible solution space, the algorithm can evenly search in the solution space for the global optimum. Orthogonal initialization is to use the orthogonal table has the dispersion and uniformity comparable; the individual will be initialized uniformly dispersed into the search space, so the orthogonal design method can be used to generate uniformly distributed initial population.

3.2. Intergenerational Elite Mechanism

In standard PSO algorithm, the next flying direction of each particle is nearly definite, it can fly to the best individual and the best individuals for the whole swarm. From the above conclusion we may easily to know it will be the danger for being trapped into a local optimum. In order to decrease the possibility of being trapped into the local optimum, the improved PSO introduces elite selection strategy. Traditional genetic algorithm is usually complete the selection operation based on the individual's fitness value, in the mechanism of elite selection, the population of the front generation mixed with the new population which generate through genetic operations, in the mixed population select the optimum individuals according to a certain probability. The specific procedure is as follows:

Step1: Using crossover and mutation operations for population P1 which size is N then generating the next generation of sub-populations P2;

Step2: The current population P1 and the next generation of sub-populations P2 mixed together form a temporary population;

Step3: Temporary population according to fitness values in descending order, to retain the best N individuals to form new populations P1.

The characteristic of this strategy is mainly in the following aspects. First is robust, because of using this selection strategy, even when the genetic operations to produce more inferior individuals, as the results of the majority of individual residues of the original population, does not cause lower the fitness value of the individual. The second is in genetic diversity maintaining, the operation of large populations, you can better maintain the genetic diversity of the population evolution process. Third is in the sorting method, it is good to overcome proportional to adapt to the calculation of scale. This process of this strategy in improve PSO like this: To set particle number in the swarm as m, father population and son population add up to 2m. To select randomly q pairs from m; as to each individual particle i, if the fitness value of i is smaller than its opponents, we will win out and then add one to its mark, and finally select those particles which have the maximum mark value into the next generation. The experiment result shows that this strategy greatly reduces the possibility of being trapped into a local optimum when solving certain functions.

We also use the three multi-peaks functions to test our new algorithm and the experiment result showed in Table 2. From the experiment results, we can say the new algorithm has got the better solution.

Table 2. Experiment results

Function	Algorithm	Best Value	Mean Value	Worst Value	$f_{\min}$
F1	PSO	1495.71	4224.775	7032.89	0
	New	8.13E-29	10.46E-26	5.08E-24	0
F2	PSO	72.5069	101.410452	123.954	0
	New	12.18E-12	0.070377	18.63E-25	0
F3	PSO	-5038.62	-4005.02	-3233.13	-12569.5
	New	-8535.19	-7741.27	-5203.56	-12569.5

4. Cicuit Design Experiment

Evolutionary Electronics applies the concepts of genetic algorithms to the evolution of electronic circuits. The main idea behind this research field is that each possible electronic circuit can be represented as an individual or a chromosome of an evolutionary process, which performs standard genetic operations over the circuits. Due to the broad scope of the area, researchers have been focusing on different problems, such as placement, Field Programmable Gate Array (FPGA) mapping, optimization of combinational and sequential digital circuits, synthesis of digital circuits, synthesis of passive and active analog circuits, synthesis of operational amplifiers, and transistor size optimization. Of great relevance are the works focusing on “intrinsic” hardware evolution in which fitness evaluation is performed in silicon, allowing a higher degree of exploration of the physical properties of the medium. This particular area is frequently called Evolvable Hardware [13-15].

In the sequence of this work, Coello, Christiansen and Aguirre presented a computer program that automatically generates high-quality circuit designs [16]. Miller, Thompson and Fogarty applied evolutionary algorithms for the design of arithmetic circuits [17]. Kalganova,

Miller and Lipnitskaya proposed another technique for designing multiple-valued circuits [18]. In order to solve complex systems, Torresen proposed the method of increased complexity evolution. The idea is to evolve a system gradually as a kind of divide-and-conquer method [19]. Based on the Miller's method, Yan applied Gene Expression Programming (GEP) [20, 21], Particle Swarm Optimization Algorithms (PSO) [7], Cultural Algorithms (CA) [22-24], Orthogonal Evolutionary Algorithm [25] and evolutionary algorithm [26, 27] for the design of electronic circuits.

4.1. One-bit Adder

Evolving the one-bit adder was easier to do on a larger geometry but resulted in a less efficient circuit. That is many genetic algorithm was able to discover 100% functional solutions was intimately related to the size of the geometry, but our algorithm use small geometry to find the fully functional solutions.

The original circuit is showed in Figure 4 (with five gates), Figure 5 is the resulting circuit designed by Miller's algorithm (with three gates) [28] and Figure 6 is our algorithm's result (with three gates). From the figures we know it is a gratifying result to obtain as it is clear that this design is an optimum solution.

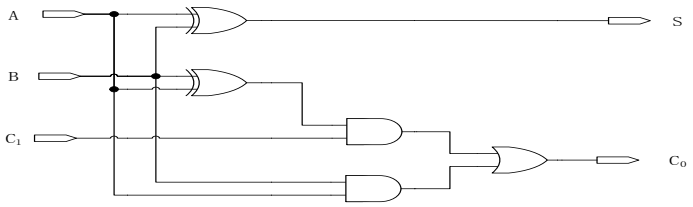

Figure 4. One-bit Full Adder Circuit Designed without Optimum

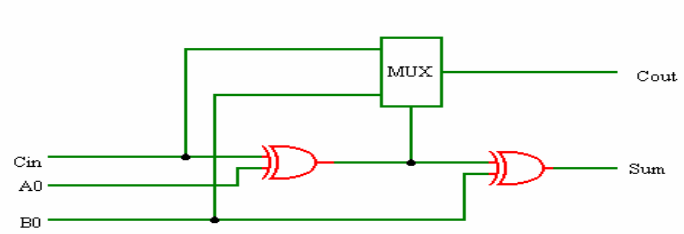

Figure 5. One-bit Full Adder Circuit Designed by Miller

Figure 6. One-bit Full Adder Circuit Designed by Our

4.2. Two-bit Adder

A two-bit full adder circuit, which with a truth table with 5 inputs and 3 outputs. In this case, our algorithm use small geometry to find the fully functional solutions, the matrix has a size of 3×3. The original circuit is showed in Figure 7 (with ten gates) Figure 8 is the resulting circuit designed by Miller's algorithm (with six gates) [28] and Figure 9 is our algorithm's result (with six gates). From the figures we know it is a gratifying result to obtain as it is clear that this design is an optimum solution.

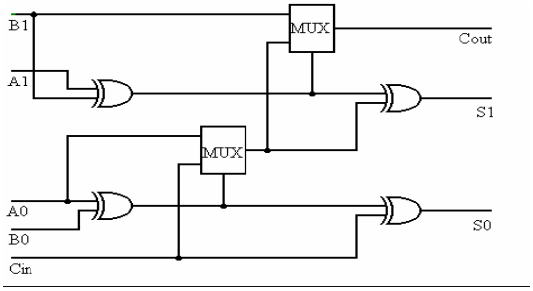

Figure 7. Two-bit full adder circuit designed without optimum

Figure 8. Two-bit full adder circuit designed by Miller

Figure 9. Two-bit full adder circuit designed by Our

5. Conclusion

This paper introduce a new algorithm based on the standard PSO algorithm, for the standard PSO algorithm the new algorithm has done two improvements: 1. In the initialization stage, we use orthogonal design method, thus enlarge global searching space and reduce the possibility of particles to be trapped into a local optimum; 2. By introducing a new selection strategy, decreased the possibility of being trapped into a local optimum. Compared with the standard PSO algorithm, the new algorithm enlarges the searching space and the complexity is not high. Experiment results based on benchmark functions and comparisons with previously reported results demonstrate the effectiveness, efficiency and robustness of the new algorithm. We use the proposed algorithm for digital circuit optimization design, and the final circuit is optimized in terms of complexity (with the minimum number of gates).

Acknowledgments

This paper is supported by Natural Science Foundation of China. (No.61272470 and No.61203307), National Civil Aerospace Pre-research Project of China, the Provincial Natural Science Foundation of Hubei (No. 2012FFB04101) and the Fundamental Research Founds for National University, China University of Geosciences (Wuhan).

References

[1] J Kennedy, RC Eberhart. Particle Swarm Optimization. *IEEE International Conference on Neural Networks.* 1995: 1942-1948.
[2] Clare M, Kennedy J. The Particle Swarm Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Trans. on Evolutionary Computation*. 2002; 6(1): 58-73.
[3] Coello CAC, MS Lechuga, Mopso. *A proposal for multiple objective particle swarm optimization*. In IEEE Proceedings World Congress on Computational Intelligence. 2002: 1051-1056
[4] J Kennedy. *The particle swarm: social adaptation of knowledge*. Proc.IEEE int. Conf. on evolutionary computation. 1997: 3003-3008.
[5] E Oscan, CK Mohan. Analysis of A Simple Particle Swarm Optimization System. *Intelligence Engineering Systems Through Artificial Neural Networks.* 1998: 253-258.
[6] XS Yan, Qing Hua Wu. *A New Optimizaiton Algorithm for Function Optimization*. Proceedings of the 3rd International Symposium on Intelligence Computation & Applications. 2009: 144-150.

[7] Xue Song Yan, Qing Hua Wu, Cheng Yu Hu, Qing Zhong Liang. Circuit Design Based on Particle Swarm Optimization Algorithms. *Key Engineering Materials*. 2011; 474-476: 1093-1098.
[8] Xuesong Yan, Can Zhang, Wenjing Luo, Wei Li, Wei Chen, Hanmin Liu. Solve Traveling Salesman Problem Using Particle Swarm Optimization Algorithm. *International Journal of Computer Science Issues*. 2012; 9 (6): 264-271.
[9] M Clerc, J Kennedy. The Particle Swarm: Explosion, Stability and Convergence in a Multi-Dimensional Complex Space. *IEEE Trans. on Evolutionary Computation*. 2002; 6: 58-73.
[10] Leung Yiu-Wing, Wang Yuping. An Orthogonal Genetic Algorithm with Quantization for Global Numerical Optimization. *IEEE Transactions on Evolutionary Computation*. 2001; 5(1): 41-53.
[11] Qinghua Wu, Hanmin Liu, Yuxin Sun, Fang Xie, Jin Zhang, Xuesong Yan. Research of Function Optimization Algorithm. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(4): 858-863.
[12] Xuesong Yan, Qinghua Wu, Can Zhang, Wei Li, Wei Chen, Wenjing Luo. An Improved Genetic Algorithm and Its Application. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2012; 10(5): 1081-1086.
[13] Zebulum RS, Pacheco MA, Belasco MM. *Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms*. CRC Press. 2001.
[14] Thompson A, Layzell P. Analysis of unconventional evolved electronics. *Communications of the ACM*. 1999; 42: 71-79.
[15] Louis SJ, Rawlins GJ. *Designer Genetic Algorithms: Genetic Algorithms in Structure Design*. Proceedings of the Fourth International Conference on Genetic Algorithms. 1991.
[16] Cello CA, Christiansen AD, Aguirre AH. Using Genetic Algorithms to Design Combinational Logic Circuits. *Intelligent Engineering through Artificial Neural Networks*, 1996; 6: 391-396.
[17] Miller JF, Thompson P, Fogarty T. Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advancements and Industrial Applications. 1997; 6.
[18] Kalgan ova T, Miller JF, Lipnitskaya N. *Multiple-Valued Combinational Circuits Synthesised using Evolvable Hardware*. Proceedings of the 7th Workshop on Post-Binary Ultra Large Scale Integration Systems. 1998.
[19] Torresen J. *A Divide-and-Conquer Approach to Evolvable Hardware*. Proceedings of the Second International Conference on Evolvable Hardware. 1998; 1478: 57-65.
[20] XS Yan, Wei Wei. *Design Electronic Circuits by Means of Gene Expression Programming*. Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems. 2006: 194-199.
[21] XS Yan, Wei Wei. *Designing Electronic Circuits by Means of Gene Expression Programming* . Proceedings of the 7th International Conference on Evolvable Systems: From Biology to Hardware. 2007: 319-330.
[22] XS Yan, Wei Wei, Kang Wang, Chengyu Hu. *Designing Electronic Circuits Using Cultural Algorithms*. Proceedings of Third International Workshop on Advanced Computational Intelligence. 2010: 299-303.
[23] XS Yan, Qinghua Wu. Electronic Circuits Optimization Design Based On Cultural Algorithms. *International Journal of Information Processing and Management*, 2011; 2(1): 49-56.
[24] Xuesong Yan, Wei Chen, Qinghua Wu, Hanmin Liu. Research of Embedded Hardware Optimization Design Algorithm. *International Journal of Computer Science Issues*. 2012; 9 (6): 70-78.
[25] XS Yan, QH Wu, HM Liu. Orthogonal Evolutionary Algorithm and its Application in Circuit Design. *Przeglad Elektrotechniczny*. 2012; 88(5b): 7-10.
[26] Xue Song Yan, Qing Hua Wu, Cheng Yu Hu, Qing Zhong Liang. Circuit Optimization Design Using Evolutionary Algorithms. *Advanced Materials Research*. 2011; 187: 303-308.
[27] Xuesong Yan, Hong Yao, Qingzhong Liang, Chengyu Hu. Research of Digital Circuit Optimization Design Algorithm. *Advances in Information Sciences and Service Sciences*. 2012; 4 (21): 556-563.
[28] Miller JF, Job D, Vassilev VK. Principles in the Evolutionary Design of Digital Circuits - Part I. *Genetic Programming and Evolvable Machines*. 2000; 1: 8-35.