Genetic Algorithm Particle Swarm Optimization Based Hardware Evolution Strategy

Zhang Junbin, Cai Jinyan, Meng Yafeng，Meng Tianzhen

Department of Electronic and Optical Engineering,
Mechanical Engineering College
Shijiazhuang , Hebei, China, 050003
mes369@163.com

Abstract: - There are many problems exist in the Evolutionary Algorithm (EA) using Genetic Algorithm (GA), such as slow convergence speed, being easy to fall into the partial optimum ,etc. Particle Swarm Optimization (PSO) can accelerate the space searching and reduce the number of convergences and iterations. The proposed characteristics of Genetic Algorithm Particle Swarm Optimization (GAPSO) are proved by many examples, when the GA, PSO and GAPSO are adopted under the same conditions, GAPSO can get the least iteration numbers and the highest evolvable success rate. It also can reduce the number of convergence iteration and raise the accuracy of searching. And the performance of PSO is inferior to the performance of GAPSO, while the GA has the worst searching performance. It also can be found that the number of initializing particles will affect the number of convergences and iterations. The larger the number of the initializing particles is, the less the number of iterations will be.

Key-Words: - Hardware Evolution, Evolutionary Algorithm, GAPSO, Fault Self-repair

1 Introduction

The traditional electronic circuit structures are hard to change after designed. With the high speed development of information technology, electronic systems are widely used in the electronic equipments [1]. The electronic system is mainly constituted by the large-scale and ultra large scale digital integrated circuits, while the Field Programmable Gate Array (FPGA) is the core of integrated circuit. When the integrated circuits work under the complex and changeable environments (such as dust, high and low temperature, strong electromagnetic scenes and so on), the performance of digital integrated circuits will be changed. The efficacy of electronic systems be reduced by faults of these equipments, and even lead to the significant casualties and property losses. If the viability of electronic systems want to be improved, and the capability of the electronic system want to be ensured in harsh environments, a series of problems need to be solved [2-5].

If we want to ensure the normal operation of the equipment, implementing the digital system faults diagnosis and fault repaired timely are necessary.The emergence of Hardware Evolution (EHW) makes the faults self-repair technology highly improved [5-9].

Evolutionary Algorithm (EA) can be treated as a combinatorial optimization search problem, it is also known as an evolution problem [9-13]. It aims at searching for the circuit hardware codes which satisfied with the objective function and using less time as possible. The EA searches in a large solution space. The EA, such as Genetic Algorithm (GA) algorithm and Particle Swarm Optimization (PSO) Algorithm are typical search algorithms; they are the most suitable as evolvable algorithms [14-17]. However, they do not have same performances.

In the paper, the current mainstream EAs, such as GA [18], Genetic Programming (EP), and Evolutionary Strategy (ES) are analyzed. There are some problems in EA, for example, slow convergence speed, being easy to fall into the partial optimum and so on. Since PSO can accelerate the space searching and reduce the number of convergences and iterations. The Genetic Algorithm Particle Swarm Optimization (GAPSO) is the most proposed here. The searching accuracy can be improved by the new algorithm, and the evolution iteration times can be reduced by the new algorithm.

The PSO was researched in reference paper [6], but it was only the software design of the on existing PSO. The comparison of PSO and GPASO were not presented. These capabilities will be analyzed in the paper.

The structure of this paper is as follows: in Section 1, the importance of EHW is discribed which is used to self-repair, and the problems of mainly EAS are also analyzed. In Section 2, the theory of EHW is introduced. In Section 3, the GAPSO strategy theory is analyzed in detail. Section 4 shows the example of GAPSO strategy. The conclusions are obtained in Section 5.

2 Basic Theory of EHW

EHW is the new and developing technique. It has been the great concern content of related researchers. The process of hardware evolution is without any other intervention. It is self-generated controlled by the computer or hardware systems.

During the evolvable period, when the environment conditions are changed, the system structure and its function can bechanged through EHW. EHW is based on a programmable device. There are some advantages exiting in EHW, e.g., self-organizing, self-adaptive and so on. It has played an important role in the area of self-repair in digital circuits.

EHW is a kind of special hardware. Its characteristics are self-adapting, self-organization and self-repairing, which can change its own structures according to the environmental changes.

The formula of EHW is expressed as

$EAs + PLDs = EHW$, EAs is Evolutionary Algorithm, PLD is Programmable Logic Device [2,3][6][12].

PLD is the hardware foundation of EHW, while EAs is the core part of EHW, which decides whether we can quickly find the effective circuit structure codes or not. The implementation of EHW strategy consists of the following steps.

(1) Get the detailed analysis of the PLD hardware structure.

(2) Encode the hardware structure.

(3) Use EA, finding the hardware structure code which conforms to the target circuit.

(4) Configure FPGA with optimized code, thus EHW completed.

The fixed-length chromosome codes is widely used in EHW.

All the switches of PLD fuse array are coded. it means that when the digital circuits have n inputs, $2n+1$ bits codes will be needed in every AND line, and then the AND lines will be selected by the product item choice matrix.

For example, when the circuit function expression is $Y=AB+\overline{C}E+\overline{B}D+ACE$. Fig.1 shows the hardware schematic diagram of function Y.

Then we can get 40-bit fixed-length chromosome codes, as 1010000000000001001 000010010001000100010. Here the codes do not include the product item choice matrix, so it is 4-bit less.

EA can automatically delete duplicate items.

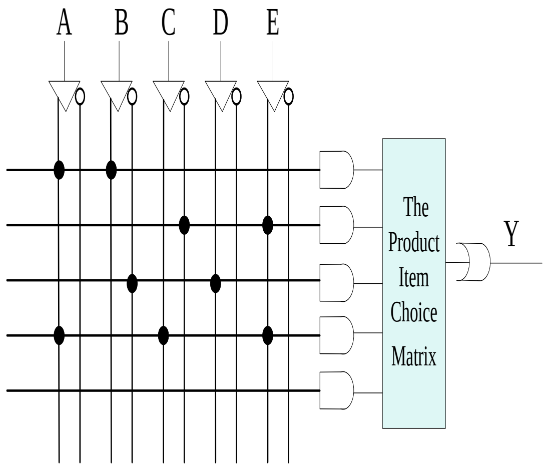

Fig.1 Hardware schematic diagram of function Y

3. GAPSO Hardware Evolution Strategy

3.1. GA and PSO Theory

3.1.1 GA Theory

In the field of EHW, GA algorithm is the most basic EA [5, 18-20]. GA algorithm was first proposed by Professor J. Holland, it drew on the experience of the natural selection and the genetic mechanism of nature. It is a random search algorithm. The characteristics of GA are not relied on gradient information. It reflects its superiority in dealing with complex nonlinear problems. GA can be widely used in artificial intelligence, pattern recognition, function optimization and self-adaptive [6].

Nowadays GA is the most basic EA. Most researchers concern the improvement of GA [7]. GA is a random search method, which uses the natural evolution thoughts. It can solve the complex problems which cannot be solved by the traditional search method. Its advantages are shown on combinatorial optimization and adaptive control [10].

The flow diagram containing the implementation of GA is shown in Fig.2.

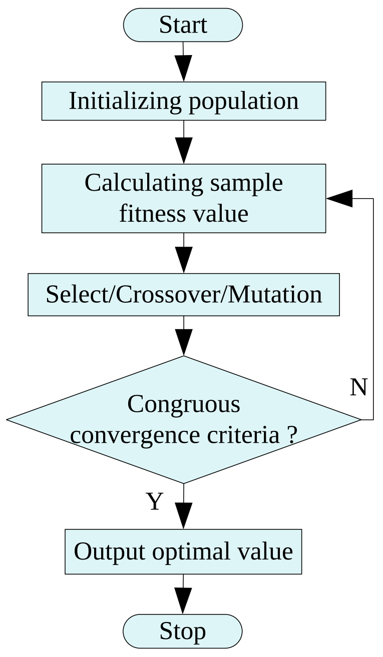
Start
Initializing population
Calculating sample fitness value
Select/Crossover/Mutation
Congruous convergence criteria ?
N
Y
Output optimal value
Stop

Fig.2 The flow diagram containing the implementation of GA

Its main implementation processes are given as follows:

Step 1 Encode the problem by using the chromosome codes.

Step 2 Initialize the population.

Step 3 Calculate the fitness function of the populations of the various solutions.

Step 4 Select two samples randomly from the population as a parental generation.

Step 5 Deal with the parental generation by using the genetic operations (including crossover, mutation and so on), so as to produce an offspring sample.

Step 6 According to certain rules, replace the original population with the descendants samples according to certain riles, so as to update the population.

Step 7 Judge whether the current populations fits for the conditions of stopping the evolution. If it does, terminates the algorithm, otherwise then turns to Step 4.

The selection operation, crossover operation and mutation operation are the core of GA. Two samples of the initial population are selected as parents randomly, and cross-points are randomly selected from parents. Sample variance can increase the diversity of the population. The paper is based on actual demand, and we select two samples in the initial population randomly, the algorithm adopts multi-point crossover and multi-point mutation, and the search speed can be improved and the search time can be reduced.

Then the fitness value of the individual will be calculated after the crossover and mutation. If the fitness is better than former samples' fitness, the former parent individuals can be replaced by the crossover and mutation individual.

GA also has many deficiencies. Its search speed is slow, and it can easy cause the search to stop into a local optimum [21].

3.1.2 PSO Theory

PSO was proposed by Dr. Eberhart and Dr. Kennedy firstly in 1995, and it derives from the research of flock foraging behavior. Compared with the GA, PSO algorithm not only has the global optimization ability of genetic algorithms, but also can adjust parameters easily, and it is suitable for processing of computer programming. By adjusting the parameters, PSO algorithm also has strong local searching capability. PSO algorithm can converge to the optimal solution more quickly than the GA under the same conditions, and the degenerate phenomenon of completely random optimization can be avoided [22,23].

The core idea of the PSO algorithm is initialized a group of random particles firstly, also it is known as random solutions. The location and speed of each particle are updated according to the current global optimal position ($pbest$) and current position ($gbest$). The process tends to the global optimal value of the gathered acceleration. Through continuous iteration, the optimal solution is found eventually [24-25].

The model can be constructed as follows. Suppose the $i-th$ particle's position in the N dimensional search space is $X^i = (x_{i,1} \quad x_{i,2} \quad \cdots \quad x_{i,N})$, its speed is $V^i = (v_{i,1} \quad v_{i,2} \quad \cdots \quad v_{i,N})$. In each iteration will produce, the $pbest$ will be produced through continuous tracking of $gbest$. The speed and location of all particles can be updated by expression (1) and expression (2) [22-28].

$$v_{i,j}{}^{t+1} = wv_{i,j}{}^{t} + c_1 r_1^1 (pbest_{i,j}{}^{t} - x_{i,j}{}^{t}) + c_2 r_2^1 (gbest_{i,j}{}^{t} - x_{i,j}{}^{t}) \quad (1)$$

$$x_{i,j}{}^{t+1} = x_{i,j}{}^{t} + v_{i,j}{}^{t+1} \quad (2)$$

Expression (1) is used to update particle’s speed, and expression (2) is used to update particle’s location. Where w is weight coefficient, c_1 and c_2 are positive learning factors, r_1 and r_2 are the symmetrical distribution random numbers between 0 and 1.

According to the situation that the chromosome codes in EA only can be 0 or 1, expression (3) is used as the commonly used fuzzy function for particle discretization.

$$sig(x_{i,j}) = \frac{1}{1+e^{-x_{i,j}}} \quad (3)$$

While the updating expression of every particle is given as expression (4).

$$X_{i,j}(t+1) = \begin{cases} 0 & rand > sig(X(i,:)) \\ 1 & rand \le sig(X(i,:)) \end{cases} \quad (4)$$

When updating the particle’s location, illegal codes may appear. In other words, the ith (i is odd) code and the $(i+1)th$ ($i+1$ is even) code can not be “1” at the same time. Illegal codes must be deleted when it appear.

Although the PSO algorithm has many advantages when compared with GA, it is not perfect. Its shortcoming is prone to premature phenomenon into a local optimum.

3.2. GAPSO Evolvable Strategy

3.2.1 GAPSO Theory

Crossover and mutation are the core of GA, the limits of the GA are very small in optimization problems. It has less requirements for the objective function and constraints, and has better robustness and adaptability merits. It does not require neither that the problem is continuous, nor that it is differentiable. It only requires that the problem can be calculated. Thus GA is better at overcoming the problem, and it is easy to fall into the plight of local minima in the optimization process. However, the convergence rate is relatively slow, while the search speed of PSO is fast. So we can combine GA with PSO algorithm in order to overcome the above problems. When GAPSO is introduced into the circuit evolution, the required time of evolve the target circuit is greatly reduced.

GAPSO is different from PSO. It has combined the crossover and mutation with the PSO algorithm, which can improve the diversity of the particle swarm significantly, and it can prevent the PSO partial optimum. Fig.3 shows the process of GAPSO.

Step1 Initialize populations of the number of particles (circuit codes initialization).

Step2 Initialize the position and speed of each particle randomly.

Step3 Calculate the fitness of each particle. And the current position of particles and the fitness value are stored in each particle’s $pbest$. Then compare all $pbest$, and the position and fitness of best individual are stored in $gbest$.

Step4 Select the individual particle swarm randomly, and execute crossover and mutation operation.

Step5 Update the position and speed of all the particles with expression (1) and expression (2).

Step6 Judge whether achieving the convergence criteria. If the convergence criterion is achieved, output the best individual, and stop the search algorithm, otherwise repeat Step3.

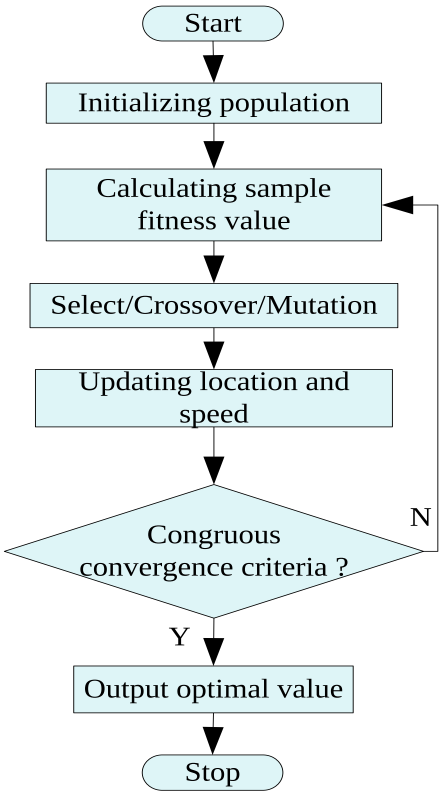
Start
Initializing population
Calculating sample fitness value
Select/Crossover/Mutation
Updating location and speed
Congruous convergence criteria ?
N
Y
Output optimal value
Stop

Fig.3 The process of GAPSO

When compared with the GA and PSO, GAPSO mainly has the following advantages.

(1) The search capabilities on the evolution strategy should be enhanced.

(2) It can improve the convergence rate of the partial area.

(3) It can avoid the partial area convergence stagnation phenomenon which exists in the search process of GA.

(4) The number of successful evolution iterations can be reduced, and the searching accuracy can be improved.

3.2.2 Objective Function Design of GAPSO

In the optimization problem, the fitness functions are mostly transformed from objective function. The output of the fitness function is also called fitness value, which can reflect the ability of individuals and adapt to the evolution. The value is greater, the adaptability is stronger, and the possibility of being preserved in evolution is greater. In order to prevent fall into local optimum and avoid the slow convergence, the difference among the individual fitness value can not be neither too big nor too small. So the reasonable fitness function needs to be designed necessary.

It is directly related to the algorithm convergence speed and accuracy whether we can get the right target function. The target of evolution circuit is to find the circuit topological structure which is according to the truth table.

We can get the target function y based on the truth table. $fitnumber_i$ can be defined which respects the ith input combination of truth table suffice for the evolution circuit, where $i \in [1, 2^n]$,and n is the number of the inputs.

Then we need to find out the maximum of optimal value ($fitvalue$), and $fitvalue$ can be considered the target function.

$$fitvalue = \sum_{1}^{2^n} fitnumber_i \qquad (5)$$

4. Analysis Example

4.1 Example 1

In this section, the GAPSO is used in EHW, and the GAPSO programs are run in MATLAB. The fixed-length chromosome codes is also used in EA.

Table.1 is the truth table of required evolvable circuit [29]. We can find that the circuit has four inputs and one output. Four root AND lines are used at the initialization of evolution. Here we adopt the ellipsis product options way, and the code length is 32 bits.

Here we apply the GA, PSO, and GAPSO in simulation experiments. The number of initialization population is $N = 60$, the crossover rate is 0.95 and the mutation rate is 0.05. Four-point crossover is used in crossover operator, and the mutation operator adopts two-point mutation.

The number of Initialize particles is 32 in the PSO and GAPSO algorithm (it equals circuit chromosome codes), the learning factor $c_1 = c_1 = 2$, w is inertia weight, $w_{\max} = 0.9$, $w_{\min} = 0.4$, the inertia weight can be expressed as (6).

Table.1 The truth table of required evolvable circuit

Input Port				Output Port
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

$$w = 0.9 - \frac{0.4 \times t}{t_{\max}} \qquad (6)$$

Where the number of current iteration is t, $t_{\max}$ is the max number of iterations.

The performance of three algorithms are compared by under the same conditions. The data of evolution iterations are shown in Table 2.

600 times are chosen as the number of iterations to terminate, then conclusions can be summaried as follows:

(1) When GA is used, the success rate is 0. It requires more iterations, and the magnitude changes are very large.

(2) When PSO is used, the success rate is 100%. It only needs 326 times iterations.

(3) When GAPSO is used, the success rate also is 100%. It only needs 172 times iterations.

Table.2 Convergence data of three algorithms

The number of trials	Algorithm GA	PSO	GAPSO
1	6965	236	109
2	------	297	168
3	4493	302	125
4	------	454	137
5	4203	301	231
6	5387	257	190
7	------	324	172
8	6301	432	169
9	4564	388	203
10	------	269	211
The number of the average iterations	------	326	172

However, for a fixed circuit functions truth table, the successful evolved circuit code is not unique, the successful code for the evolution of the three algorithms are also not unique.

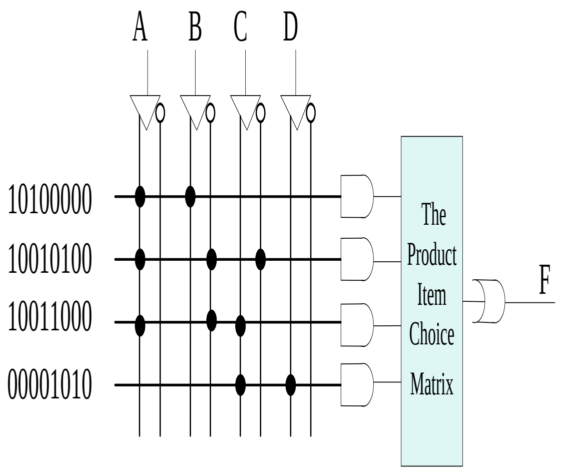

Fig.4 Circuit structure of chromosome encodes

Taking the GAPSO as an example, after the success of the evolution, we get the 32 bits chromosome code,and it is 10100000101001001010100000001010. For the above evolved chromosome code, the circuit structure is shown in Fig 4.

Corresponding circuit function expression is $F = AB + AB\overline{C} + CD + ABC$. However, it is not an optimal structure expression.

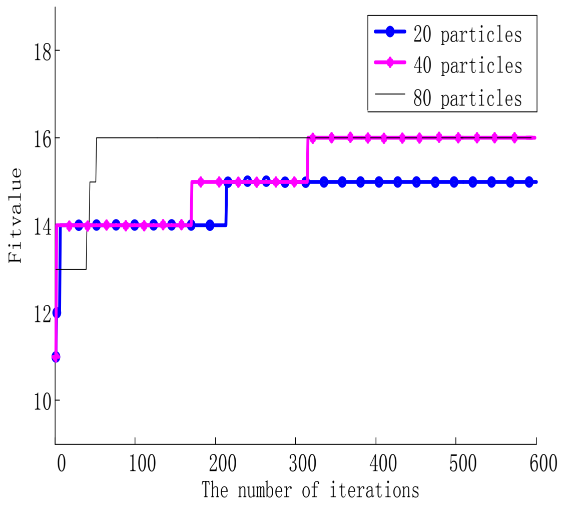

Fig.5 The diagram of the initialize the number of particles and the number of iterations

The number of initialization particle swarm (IPS) has also been researched in this paper. When GAPSO is used as the EA，the number of IPS is a variable, it is equal to 20， 40 and 80, and the results are shown as Fig.5. Through related examples, we can find that when the number of IPS is 80, evolution iteration requires the least numbers, and it only needs 83 times. When the number of IPS is 40, the number of evolution iteration is 309. When the number of IPS is 20, the number of evolution iteration is 968，and it exceeds 600 times. So the larger number of the initializing particles will lead to the less number of iterations.

4.2 Example 2

In the first example, the target evolution circuit is a multiple-input and one-output circuit. Here multiple-inputs and multiple-outputs circuit will be simulated. Here 2-4 binary decoder circuit is the target evolution circuit, and its truth table is shown in Table 3.

Table.3 The truth table of 2-4 binary decoder

Input Port			Output Port			
EN	X1	X0	Y3	Y2	Y1	Y0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Because the evolved circuit has multiple inputs and outputs, 5 AND lines in evolution are used, the length of chromosome codes are 30 bits. Here we take the GA, PSO, and GAPSO as the example. The number of initialization population is $N = 60$, the crossover rate is 0.95 and the mutation rate is 0.05. Four-point crossover is used in crossover operator, and the mutation operator adopts two-point mutation. In the PSO and GAPSO algorithm, the learning factor $c_1 = c_1 = 2$, Equation (6) is still used as inertia weight (w).

Three algorithms are compared through 6 experiments under the same conditions. The maximum number of iterations is 400 times. The data of evolution iterations are shown in Table 4.

600 times is also chosen as the number of iterations to terminate the process of EA, then we can obtain these conclusions. When we apply the GA, the success rate is 0%, and it requires more iteration. When the PSO is adopted, the success rate is 100%, and it only needs 242.5 times iterations. When GAPSO is used in experiment, the success rate also is 100%,and it only needs 168.2 times iterations.

Above results are decided by the nature of GA, GA is actually a random search algorithm, and the purpose of the search process is not strong. While PSO and GAPSO algorithms have their certain search goals, their performance can be balanced by learning factor and inertia weight, and the effects are certainly better than GA. These conclusions are proved by above experiments. However, the PSO algorithm may fall into local optimum. Combined GA’s core idea with the PSO algorithm, GAPSO algorithm can achieve preferable results. It can search faster, and prevent the algorithm from fall into local optimum at the same time.

Table.4 Convergence data of three algorithms

The number of trials	Algorithm		
	GA	PSO	GAPSO
1	------	243	194
2	------	221	158
3	------	216	157
4	------	306	161
5	------	264	173
6	------	205	166
The number of the average iterations	------	242.5	168.2

Although here the 2-4 binary decoder is a multiple-inputs multiple-outputs circuit, a multi-objective evolutionary problems can also be converted to a simple-objective evolutionary problems. Fig.6 is the sketch map of converting truth table. In fact, the second example can be treated as a circuit which has 15 input combinations, and one input combination is reduced than the first example.

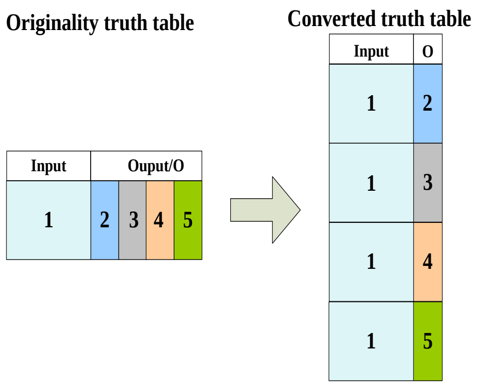

Fig6. Sketch map of converted truth table multiple-target to simple target

5. Conclusion

When the GA, PSO and GAPSO are used under the same conditions, GAPSO has gotten the least iteration numbers and the highest evolvable success rate. And the performance of GAPSO is followed by the PSO, while the GA has the worst search performance. The number of initializing particles can affect the number of convergences and iterations. The larger number of the initializing particles is, the less number of iteration will be. The GAPSO can reduce the number of convergence iterations and raise the search accuracy. It can overcome the defects of GA and PSO.

EA is an important part of the EHW, and the existing hardware evolutions mostly adopt GA. It is meaningful to improve the algorithm and make some innovations, and the GAPSO has important engineering application value.

References:

[1] Yang Shiyuan. Digital System Faults Diagnose and Reliability Design [M]. Beijing: Tsinghua University Press, 1999.

[2] Pan Zhengjun, Kang Lishan, Chen Yuping. Evolvable Arithmetic [M]. Beijing: Tsinghua University Press, 1997.

[3] Kang Li Shan, He Wei, Chen Yuping. Evolvable Hardware realized with function type programmable device [J]. CHINESE J. COMPUTERS, 1999, 22(7): pp.781-783.

[4] Tufte, G. Haddow, P. C. Evolvable Hardware [M].Proceedings of the First NASA/DoD Wor kshop on. 1999

[5] Taher NIKNAM, Ehsan AZAD FARSANI. A hybrid evolutionary algorithm for distribution feeder. SCIENCE CHINA(Technological Sciences). 2010, vol.53 no. 4 ,pp.950–959.

[6] XU Xing, LI Yuanxiang, WU Yu, et al. Design of algorithm platform for particle swarm optimization based on strategy pattern [J]. Engineering Journal of Wuhan University, 2010, 43 (3): pp.361-366.

[7] Al. G, Madhu M, and Deepak S. Java Genes: Evolving Molecular Force Field Parameters with Genetic Algorithm, Computer Modeling

in Engineering and Science, 2002, 5 (3) pp.557-574

[8] Miller J. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In Proc. of the Genetic and Evolutionary Computation Conference (GECCO'99), 1999, 4 (1) pp.1135–1142

[9] Higuchi, T. Iwata, M. Keymeulen, D. Sakanashi, H, et. al. Real-world applications of analog and digital evolvable hardware. IEEE Transactions on Evolutionary Computation, 1999, vol.3, no. 3, pp.220-235.

[10] Gu Yingsong, Zhang Xinping，Yang Zhicun. Robust flutter analysis based on genetic algorithm [J]. SCIENCE CHINA (Technological Sciences). 2012Vol 55, no. 9. pp. 2474–2481.

[11] Sekanina L ， Friedl S . An Evolvable Combinational Unit for FPGAs ［J］. Computing and Informatics，2005, 24(3): pp1-26.

[12] Li Yong. Research Fault Tolerance Methods Based on Evolvable Hardware [D]. Hefei: University of Science and Technique of China, 2007.

[13] Liu Changqing, Zhang Xihuang. Research and application of evolutionary algorithm in evolvable hardware [J]. Computer Engineering and Design, 2008, 29(24) pp.6390－639.

[14] Xue Yuncan, Zheng Dongliang, YANG Qiwen. Optimum charge plan of steelmaking continuous casting based on the modified discrete particle swarm optimization algorithm [J]. Computer Integrated Manufacturing Systems, 2011, 17(7) pp.1509-1517.

[15] Deng Tieqing, Ren Genquan, LiuYingbo.. Genetic Algorithm Based Approach to Personal Worklist Resource Scheduling [J]. Journal of Software, 2012, 23(7),pp.1702-1714

[16] [Wang Zhu-Rrong, Zhang Jiulong, Cui Duwu. Optimization Algorithm for Solving Degree-Constrained Minimum Spanning Tree Problem [J]. Journal of Software, 2010, 21(12),pp.3068-3081

[17] Mu Caihong, Jiao Licheng, Liu Yi. M-Elite Coevolutionary Algorithm for Numerical Optimization [J]. Journal of Software, 2009, 20(11): pp.2925-2938

[18] Zhang L, Zhang B. Good point set based genetic algorithm. Chinese Journal of Computers,2001, 24(9), pp.917-922

[19] Zhang Xuguang, Hu Shuo, Chen Dan, et. al. Fast Covariance Matching With Fuzzy Genetic Algorithm. IEEE Transactions on Industrial Informatics. 2012, vol. 8, no. 1, pp. 148-157.

[20] Hu S, Zhu M, Song H, et. al. "Fast image template matching based on evolutionary algorithms," in Proc. Int. Conf. Comput. Graph. Imag, 2004, pp. 435–437.

[21] Liu Lizhi, Wu Fangxiang, Zhang W J. Inference of Biological S-System Using the Separable Estimation Method and the Genetic Algorithm. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2012, vol.9, no. 4, pp. 955-965.

[22] Peram T, Veera machaneni K, Mohan CK. Fitness-Distance-Ratio based particle swarm optimization. In: Proc.of the Swarm Intelligence Symp. 2003. pp.174-181.

[23] Van den Bergh F, Engelbrecht AP. A cooperative approach to particle swarm optimization. IEEE Trans. on Evolutionary Computation, 2004, 8(3), pp.225-239.

[24] Li Pei, Duan Haibin. Path planning of unmanned aerial vehicle based on improved gravitational search algorithm. SCIENCE CHINA (Technological Sciences). 2012, Vol.55, No.10, pp. 2712–2719.

[25] Trelea IC. The particle swarm optimization algorithm: Convergence analysis and parameter selection. Information Processing Letters, 2003, 85(6),pp.317-325.

[26] Wang Chao, Gao Jinghuai. High-Dimensional Waveform Inversion With Cooperative Coevolutionary Differential Evolution

Algorithm. IEEE Geoscience and Remote Sensing Letters, 2012,vol.9, no. 2, pp. 297-301.

[27] Liu Lizhi, Wu Fangxiang. An Organizational Evolutionary Algorithm for Numerical Optimization. IEEE Trans. on Systems, Man, and Cybernetics—Part B: Cybernetics, 2007, 37(4):,pp.1052-1064.

[28] Tao Xinmin, Xu Jing, Yang Libiao. Improved Cluster Algorithm Based on K-Means and Particle Swarm Optimization [J]. Journal of Electronics & Information Technology, 2010, 32(1),pp.92-97.

[29] Zhang Wei. Foundation Research of Hardware Evolution based on Evolvable Arithmetic [D]. Nanjing: Nanjing University of Science and Technology, 2008.