Optimization of the Particle Swarm Algorithm

J. Chytil
Department of Theoretical and Experimental Electrical Engineering
Brno University of Technology, Technicka 12, Brno 616 00, Czech Republic

Abstract— Particle Swarm Optimization is a swarm intelligence based and stochastic algorithm to solve the optimization problem. This paper presents the Multidimensional Particle Swarm algorithm with non-equidistant discrete input data such as E-Series or Renard numbers for circuit design. The authors describe the optimization of this method for different circuit designs, agent recycling, and omission of already computed points. The problem of omission of already computed points is to determinate when it is faster to omit the points than compute them. The personal omission history can be used for agent recycling or trajectory corrections. There is also described effect of recycled agents with corrected parameters on the convergence of optimalization. Parameters corrections are based on the principles of genetic algorithms in the other words inheritance from the best rated agents. System of agents rating is described briefly.

1. INTRODUCTION

This paper presents problems related to the need of implementing the method of Particle Swarm Optimization in purpose-built software for circuit design. A substantial portion of circuit components can be computed using analytic equations. The results, however, are only rarely accurate within E-Series values [1]. The algorithm based on Particle Swarm Optimization (also referred to as PSO) should choose the best design out of the set of E-series components, considering their tolerance. Another aspect to be assessed by the algorithm consists in the results of multicriterial circuit evaluation. With respect to the implementation of Particle Swarm Optimization [2, 3], substantial effort has been made to accelerate this method to the highest posssible degree.

2. THE PROBLEM OF OMISSION OF ALREADY COMPUTED POINTS

The very first way to make PSO faster is right decision when is faster to compute already computed value of the function or when is better to omit the computing. That problem can be resolved by programmer or can be set by selfevaluating methode. Self evaluating methode used measurement of function computing timeconsumtion and compare it with timeconsumption of omitting algorythm. The selfevaluating algorythm is hard to set and it is depended on a lot of parameters like agent count, maximum runs and last but not least on the shape of function. And it also afected by size of space.

Limiting size of space is on of most importat thing to do for speed up PSO alogorythm. The first borders have to be based on physical properties. For example and for simlicity: If we are going to design resistor consist of two serial connected resistors we definitely know that the each of resistors must have lower or equal resistance like searched resistor.

Empiricaly was shown that there is better to compute the function rather then trying to omit the computing if there are an analitical equations. And counter it, there is prefered trying to omit the computing if there is simulation necessary to obtain value of the criterial function. And there is still the selfevaluating algorythm that can be used.

This algorythm measure the time of computing criterial function for each agent and compare this time with measured time of looking for possibility of duplical computing in pregenerated artifical set of used position. And considering the results the method will be choosen.

3. OCCASION OF AGENT TERMINATING OR RECYCLING

The only reason to terminate an agent or his recyclation is that this agent haven't got the benefits for algorythm. The benefits of agent is assessed by evaluating function. An example of evaluated properties and it's score or penalization are in Table 3.

If the agent evaluating function is determined. It is necessary to determine decision function which decide which agent will be terminated or recycled.

This task can by resolved by triggering level. In the other words if the evaluation function of Agent goes under the triggering level the Agent will be recycled or terminated. Or it can by resolved by probabilities of termination or recyclation. So the agent with lower value of evaluation function has got greater probability of termination or recyclation.

Table 1: Example of evaluated properties and its score or penalization.

Evaluated property	**score**
Found global best result	+100
Found personal best result	+20
Occurance on already computed point	−15
Criterial function Order of magnitude higher	−10 per order of magnitude
No benefit for algorythm	−5per run
Stayed in global best	−50 each run
Occurance on already computed points near global best	−25

Table 2: Example of agent setting.

Agent's property	**value**
Own speed	0.82
Global best attraction	0.3
Personal best attraction	0.15
Group best attraction	0.23
Randomize of Global best attraction Standard deviation	0.3
Randomize of Personal best attraction Standard deviation	0.3
Randomize of Own speed Standard deviation	0.3
Probability of affecting by random speed	0.05
Velocity increase based on high value of criterial function	0.1

There is demand to preset agent new score after recyclation. Is not wise to set it on constant value. The better way is used average or median score value of agent's population.

4. WAYS OF AGENT'S RECYCLING

There are countless ways of agent's recycling and here is only small enumeration out of it's general principles.

1. New position and speed
   - Simple generation of new random position or speed or both
   - Pre-computed new speed and position
2. New agent's setting
   - The same as all of agnets
   - New random
   - Inherited from the best agents (better Agent has got greater probability of refering of his setting)
   - Inheritance based on genetic algorithm
3. Agent's acquired information (for example personal best position and value)
   - Forgetting of all the informations
   - Forgetting of selected informations
   - Modifying of chosen informations
   - Keep all the information
4. Agent's shared values (for example global best position and value)
   - Unchanged system (values all shared with each of all agents)

- Shared between agent generations
- Exponential moving average

The tested methods are all based on simple generation of new random position and speed, forgetting of all personaly acquired informations and global values shared over all of the Agents. So the only instrument which is varied is modifying of agent's setting. The method of bare recyclation (the same setting as all of Agnets), random generation of setting, the method of inheritance from the best Agents and also the methods based on genetic algorithms will be compared with methods without recyclation.

5. NEW AGENT'S SETTING BASED ON PROBABILITY AND GENETIC ALGORITHM

The method used to obtain new Agent's setting based on probability is quite easy to implement. The Agent will be choosen to refer it's setting to the new Agent with certain probability. This probability is determined by the score of Agents (obtained by evalution function). The Agents with better score have greater probability of to be choosen.

Modified method of obtaining new Agent's setting is based on Genetic algorythm (search heuristic that mimics the process of natural selection). During each successive generation, a proportion of the existing population is selected to breed a new generation [4, 5]. The new parameters are obtain by some of Crossover techniques from two or more parents.

There are used method called Three parent crossover. But it can be easily replaced by another genetic algorithm.

6. METHODIC AND RESULTS OF TESTING

For testing were used four corrected functions for testing of multidimesional optimalization methods in E-Serie's space. There are used Sphere function, Rosenbrock function, Styblinski Tang function and Rastrigin function. Futher can be used Shekel function.

Sphere function and it's minimun

$$f(x) = \sum_{i=1}^{n} x_i^2, f(x_1, \ldots, x_n) = f(0, \ldots, 0) = 0 \tag{1}$$

Rosenbrock function and it's minimun

$$f(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, f(x_1, \ldots, x_n) = f(1, \ldots, 1) = 0 \tag{2}$$

Rastrigin function and it's minimun

$$f(x) = An + \sum_{i=1}^{n} (x_i^2 - Acos(2\pi_i)), f(x_1, \ldots, x_n) = f(0, \ldots, 0) = 0 \tag{3}$$

Styblinski Tang function and it's minimun

$$f(x) = \frac{\sum_{i=1}^{n} x_i^4 - 16x_i^2 + x_i}{2}, f(x_1, \ldots, x_n) = f(-2.903534, \ldots, -2.903534) = -39.16599n \tag{4}$$

Shekel function and it's minimun

$$f(x) = \sum_{i=1}^{m} \frac{1}{c_i + \sum_{j=1}^{n} (x_j - a_{ij})^2}, \quad \textit{given by matrix } A \tag{5}$$

Functions are corrected for usage with E-Series components. Working area is limited by 6 decades of E-Series. And every optimalization process run ten-thousand times for obtain the average duration.

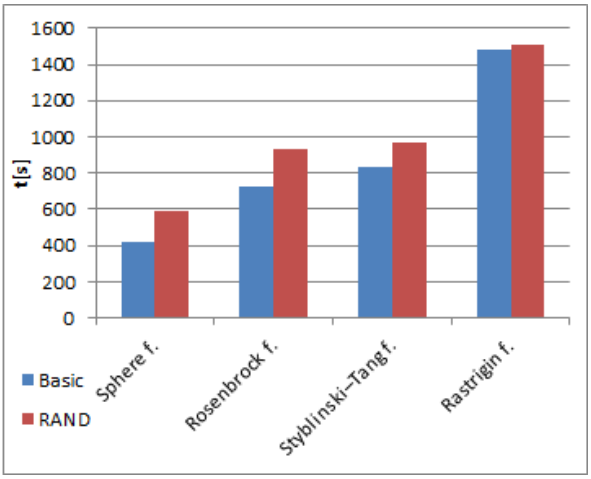

Figure 1: Comparison of the basic PSO and PSO with random setting of recycled Agents.

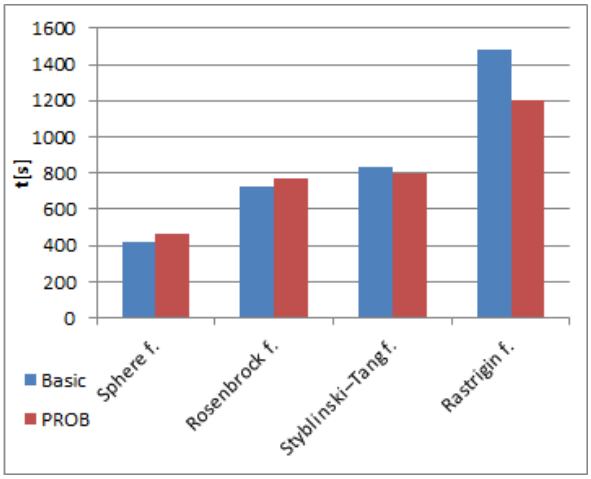

Figure 2: Comparison of the basic PSO and PSO with setting of recycled Agents generated considering probability.

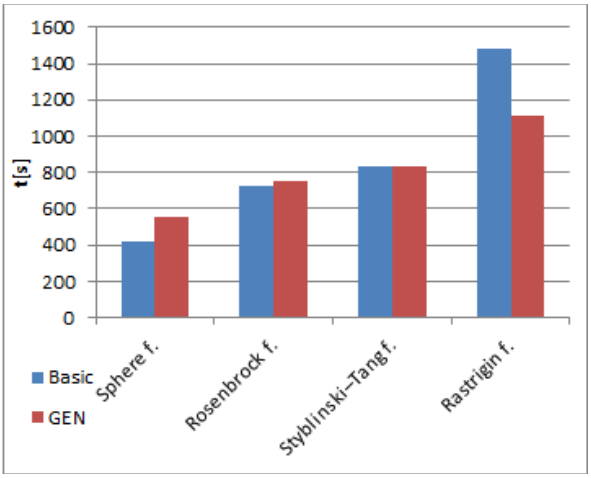

Figure 3: Comparison of the basic PSO and PSO with new Agents setting based on genetic algorith.

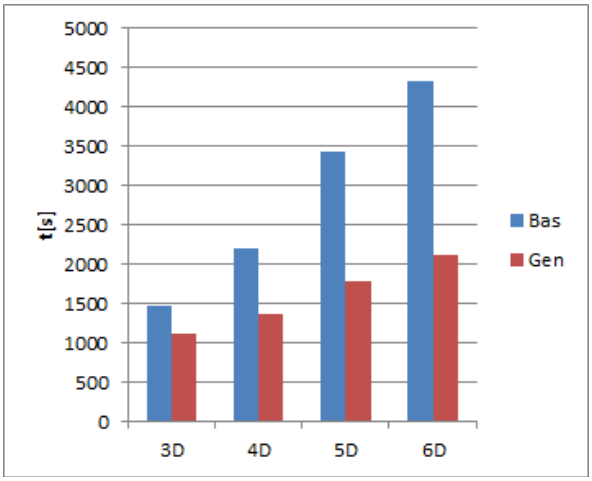

Figure 4: Ilustration of benefits of recyclation Agents with setting inheritance based on genetic algorith versus dimensions.

7. CONCLUSION

Generally, applicable results for the PSO are difficult to obtain because of a large number of the degrees of freedom (the number of Agents, number of dimensions, the Agent setting which cannot be modified by the pre-set function, and other aspects). In the given context, it is also important to point out the lack of test functions for multidimensional optimization in comparison with the count of such functions for one or two dimensions. Although the acquired results are not accurate, they show certain regularities in the behaviour of different settings of the PSO.

It is clear that a correctly set PSO without recycling surpasses PSOs with recycling in case of simple functions, such as the sphere or Rosenbrock ones. However, the more complex - for example, Rastrigin - functions were solved faster by the algorithm with recyclation.

By further extension, we can also point to the fact that, when used with the more complicated functions (the Rastrigin function), the genetic algorithm provides significant benefit; this aspect becomes even more obvious with increasing dimension of the solved function. This state occurs because the genetic algorithm needs enough iterations of the PSO to express itself.

8. GLOSSARY OF TERMS

Agents generation — group of agent with the same generation number of agent's recyclation).

Agents goupe — group of agent whom sharing some parameters or values.

ACKNOWLEDGMENT

The research described in the paper was financially supported by a grant of the BUT science fund, no. FEKT-S-11-5/1012, and the authors also received assistance from projects within the Education for Competitiveness Operative Programme, nos. CZ.1.07.2.3.00.20.0175.

REFERENCES

1. Ibrahim, I., Z. M. Yusof, S. W. Nawawi, M. A. A. Rahim, K. Khalil, H. Ahmad, and Z. Ibrahim, "A novel multi-state particle swarm optimization for discrete combinatorial optimization problems," *Computational Intelligence, Modelling and Simulation*, 18–23, 2012.
2. Zhang, H. and H. Qing, "Hybrid multiagent swarm optimization: Algorithms, evaluation, and application," *Decision and Control*, 5699–5704, 2012.
3. Anantathanavit, M. and M. A. Munlin, "Radius particle swarm optimization," *Computer Science and Engineering Conference*, 126–130, 2013.
4. Pratiwi, L., Y.-H. Choo, N. A. Muda, and A. K. Muda, "Improving ant swarm optimization with embedded vaccination for optimum reducts generation," *Hybrid Intelligent Systems*, 448–454 2011.
5. Wu, X., "A density adjustment based particle swarm optimization learning algorithm for neural network design," *Electrical and Control Engineering*, 2829–2832, 2011.