The Fifteenth International Symposium on Artificial Life and Robotics 2010 (AROB 15th '10),
B-Con Plaza, Beppu,Oita, Japan, February 4-6, 2010

Particle Swarm Optimization with Genetic Recombination - A Hybrid Evolutionary Algorithm

Sam Chau Duong, Hiroshi Kinjo, Eiho Uezato
Faculty of Engineering, University of the Ryukyus
Senbaru 1, Nishihara, Okinawa 903-0213, Japan

Tetsuhiko Yamamoto
Tokushima Technology College
Itano-gun, Tokushima 779-0108, Japan

Abstract

This article presents a hybrid evolutionary algorithm (HEA) based on particle swarm optimization (PSO) and real-coded genetic algorithm (GA). In the HEA, PSO is used to update the solution and a genetic recombination operator is added to produce offspring individuals based on the parents that are selected in proportion to their relative fitness. Through the recombination, new offspring enter the population and the individuals with poor fitness are eliminated. The performance of the proposed hybrid algorithm is compared to those of the original PSO and GA and the impact of the recombination probability to the performance of the HEA is also analyzed. Various simulations of multivariable functions and neural network optimizations are carried out, showing that the proposed approach is superior over the canonical means.

Keywords: Hybrid evolutionary algorithm, Particle swam optimization, Genetic algorithm, Multivariable optimization, Neural network.

1 Introduction

Evolutionary algorithms have been emerged in the growing study and applied pervasively in various areas. Although their competence have been proved to be superior over the conventional methods, an experienced combination of operations (either full or partial) from different approaches may provide a more efficient performance. In fact, the hybridizations of evolutionary algorithms have been the focus of much research recently and they are considered as effective general-purpose tools for the goals of *exploration* and *exploitation* [1]. The hybrids of genetic algorithms (GAs) and particle swarm optimization (PSO) have become a popular and interesting framework with capability of handling several real world problems.

In general, evolutionary algorithms involve strong stochastic basic, they therefore require many generations to obtain good solution. In this article, a hybrid of PSO an real-coded GA is proposed, called the hybrid evolutionary algorithm (HEA). With the focus on the investigation of how quick an algorithm can find a solution, comparison between the proposed HEA and the canonical PSO and GA is carried out with small population size and generation.

The rest of this article is organized as follows. Section 2 is a background on the PSO and real-coded GA used in this study. In Section 3, the proposed hybrid algorithm is introduced. Simulations will be shown in Section 4, where the performance of the proposed method is investigated and compared to those of the canonical PSO and real-coded GA. The optimization problems of multivariable function are first considered. The neural network optimization for the exclusive-or (XOR) problem is then examined. Lastly, Section 5 is the discussion and conclusion of the study.

2 Brief Background on Particle Swarm Optimization and Genetic Algorithm

2.1 Particle Swarm Optimization

As one of the latest evolutionary optimization methods, PSO is a population-based stochastic approach which provides efficient performance with simple operators [2],[3].

Assume that the search space is D-dimensional, the n-th particle (solution) of the swarm is represented by a D-dimensional vector $X_n = [x_{n1}, x_{n2}, \cdots, x_{nD}]^{\mathrm{T}}$. The n-th particle's velocity is also a D-dimensional vector, denoted as $V_n = [v_{n1}, v_{n2}, \cdots, v_{nD}]^{\mathrm{T}}$. The best position of the n-th particle is $PB_n = [pb_{n1}, pb_{n2}, \cdots, pb_{nD}]^{\mathrm{T}}$ (the *local best*, the smallest objective value that the n-th particle has obtained so far), and the best position of the swarm is denoted as gb (the *global best*, the smallest objective value achieved by any particle in the population). After finding the two best values in the k-th iteration, the particle updates its velocity and position in the next iteration $(k+1)$ with following equations:

$$v_{nd}^{k+1} = \lambda v_{nd}^{k} + c_1 r_1 \left(pb_{nd}^{k} - x_{nd}^{k}\right) + c_2 r_2 \left(gb_{d}^{k} - x_{nd}^{k}\right) \quad (1)$$

$$x_{nd}^{k+1} = x_{nd}^{k} + v_{nd}^{k+1} \quad (2)$$

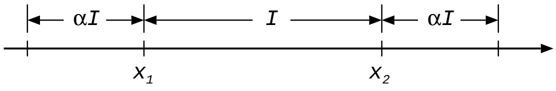


Figure 1: The BLX-α crossover

where $d = 1, 2, \cdots, D$; $n = 1, 2, \cdots, N$ (N is the swarm population size); $k = 1, 2, \cdots, K$ is the iteration (or generation) number; λ is the inertia weight; c_1 and c_2 are positive constants (in this study $c_1 = c_2$, which is usually used); r_1 and r_2 are random values in the range [0, 1].

2.2 Genetic Algorithms

Genetic Algorithms (GAs) are adaptive heuristic search techniques based on the ideas of natural selection and genetics. Generally, the main driving operators of GAs are selection, recombination via crossover, and mutation.

In this study, the recombination of a real-coded GA is used where offspring are produced by the so-called blend crossover (BLX-α) [4]. The BLX-α generates offspring by picking values on an interval that contains two parents and may extend equally on either side of the interval with a range determined based on a range parameter α (see Fig. 1).

3 A Hybrid of Particle Swarm Optimization and Genetic Algorithm

In this paper, a hybrid of PSO and GA is proposed, where the BLX-α crossover is used to produce offspring (M individuals). The operation of the proposed hybrid evolutionary algorithm (HEA) is shown in Fig. 2. In the figure, $(.)_b$ presents the local best value of an individual in PSO and x'_n is the new position of the individual (the updated position by Eqs. (1) and (2)). After being produced by the BLX-α recombination and being evaluated, the offspring x_{n+m} sets itself to be its local best value. A ranking procedure based on the fitness of each individual is performed for the pool of all parents and offspring, from which the individuals with poor fitness are eliminated to maintain a constant population. The individual with highest fitness is set to be the global best of the swarm.

The recombination with selection is the main difference of the HEA, GA compared to PSO. Suppose that the probability of offspring production is μ, the GA and the proposed HEA shall produce $M = \mu \times N$ offspring and also eliminate the same amount (the inferiors) to keep a constant population size of N at each generation. As a result, the number of individuals that is taken into the evaluation at each generation is $(1 + \mu) \times N$.

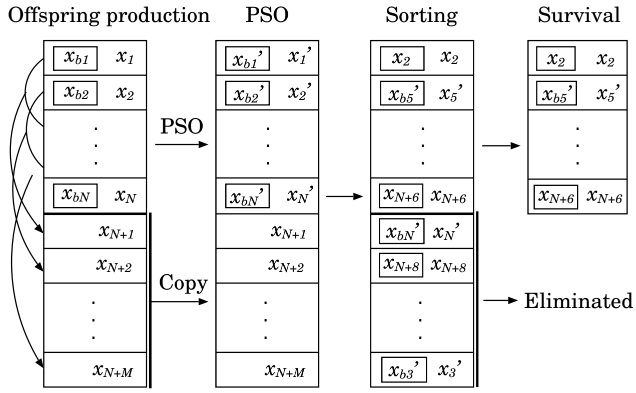


Figure 2: Process of the proposed hybrid algorithm

In the GA recombination, the Roulette wheel technique is used to select parents for reproduction in proportion to their relative fitness, which is defined as:

$$\text{Fitness} = \frac{1}{1 + E} \qquad (3)$$

where E is the error between the obtained cost function value and the optimal cost function value (the optimal solution of the problem).

4 Numerical Simulations

4.1 Multivariable Function Optimization

4.1.1 Parameter and test design

In general, a high recombination probability results in a severe selection of several solutions and thus usually provides good performance of a competitive-selection based EA. Thus, in order to make a fair comparison with the PSO as well as to demonstrate the advantage of the proposed method, a small recombination probability is used, that is $\mu = 0.1$. Also, the generation number K is set to be small and the population size is between 10 and 40 (this range is often used in PSO).

Since evolutionary algorithms are highly dependent on the initial random weights, 100 replications of changing the initial population will be implemented. Also, while the algorithms search for optimal solution in hyperspace, the initial population are drawn randomly from a uniform distribution from the range $[-10.0, 10.0]$, which is intentionally set to be large enough to make the search problem more difficult. The performances of the algorithms are evaluated by the success rate and the mean cost function value. The success rate is the fraction of optimization runs in which an algorithm can achieve small enough error (*i.e.*, $E < E_{suc} = 10^{-4}$).

While using $c_1 = c_2 = 1.0$, which is found to be suitable for the problems being considered, we shall show only the best result (i.e., with highest success rate and/or

Table 1: Test functions

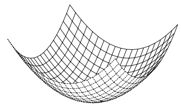
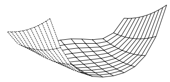
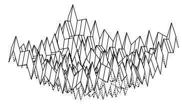
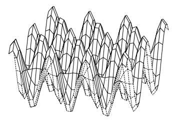
Function	Formula	Optimal solution	Sketch in 2D
F_1: Sphere	$f(x) = \sum_{i=1}^{D} x_i^2$	$F^* = 0, \quad x_i^* = 0$	
F_2: Rosenbroke	$f(x) = \sum_{i=1}^{D-1} \left(100 \left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	$F^* = 0, \quad x_i^* = 1$	
F_3: Rastrigin	$f(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$F^* = 0, \quad x_i^* = 0$	
F_4: Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$F^* = 0, \quad x_i^* = 0$	

Table 2: Parameters resulting in the best performance of the algorithms for specific problems

Function	GA–α	PSO–λ	HEA–(α, λ)
$F1$	$\alpha = 1.0$	$\lambda = 0.2$	$\alpha = 0.9$, $\lambda = 0.4$
$F2$	$\alpha = 1.0$	$\lambda = 0.6$	$\alpha = 1.3$, $\lambda = 0.7$
$F3$	$\alpha = 1.6$	$\lambda = 0.5$	$\alpha = 1.0$, $\lambda = 0.4$
$F4$	$\alpha = 0.1$	$\lambda = 0.7$	$\alpha = 1.6$, $\lambda = 0.6$

smallest mean cost) of each algorithm for specific problem by tuning for the most suitable values of α in the GA, λ in the PSO, and (α, λ) in the HEA. Let us denote them as the BLX–α (or GA–α), PSO–λ, and HEA–(α, λ).

In this section the algorithms are investigated through the optimizations of four functions as shown in Table 1 with the dimension number of $D = 2$. In these functions optimizations, since the optimal solution is $F^* = 0$, we define $E = F$ where F is the value of the cost function obtained after a run of the algorithm.

4.1.2 Simulation result

Using the tuned parameters in Table 2, the results of optimizing the functions are shown in Table 3 for $K = 20$ and $K = 50$. It is clear that the HEA usually obtains higher success rates and lower averaged cost values.

4.2 Neural Network Optimization

4.2.1 Parameter and test design

This section presents the experiment of an NN optimization for the well-known exclusive-or problem (XOR) (see Table 4), which is known as a highly nonlinear multivariable optimization problem.

In the NN, a linear function $f(x) = x$ is kept for the input and output layers while activation for the hidden layer is a sigmoid function, which is:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4}$$

A 2-5-1 structured NN is utilized which results in a 15-variable optimization problem. The initial connection weights are drawn randomly from the range [-5.0, 5.0]. Performances of the algorithms are also evaluated through 100 iterations of changing the initial population. The error function is defined as

$$E = \sum_{p=1}^{4} (T_p - O_p)^2 \tag{5}$$

where T_p is the desired output or teacher signal, O_p is the obtained output value of NN for pattern p, and P is the number of patterns (for the XOR problem, $P = 4$).

4.2.2 Simulation result

Table 5 shows the simulation results of the NN optimization with $K = 50$ and $K = 100$ generations, using the GA–0.1, PSO–0.7, and HEA–$(0.6, 0.8)$. Again, the proposed HEA outperforms the PSO and GA.

5 Discussion and Conclusion

In this research, we have presented a novel hybrid evolutionary algorithm based on a PSO and a real-coded GA. The performance of the HEA is compared with those of the canonical approaches, showing a good performance of the proposed method regardless of the small values of the generation number, population size and the recombination probability μ.

In the tests the GA demonstrated a poor performance. This is due to the small values of generation, population size and the recombination probability μ as well as the fact that GAs are very sensitive to the initial population.

Table 3: Success rate [%] and mean cost value (**succ. rate**/mean cost) with $D = 2$, $K = 20, 50$

Func.	N	K	GA	PSO	HEA
F_1	10	20	**1**/10.96269	**68**/0.16374	**100**/0.0
		50	**1**/10.96269	**68**/0.16360	**100**/0.0
	20	20	**0**/0.75099	**99**/0.00077	**100**/0.0
		50	**2**/0.10052	**99**/0.00076	**100**/0.0
	30	20	**1**/0.49964	**100**/0.0	**100**/0.0
		50	**3**/0.08395	**100**/0.0	**100**/0.0
	40	20	**1**/0.21150	**100**/0.0	**100**/0.0
		50	**12**/0.01293	**100**/0.0	**100**/0.0
F_2	10	20	**0**/3006.9817	**1**/1.69848	**1**/1.51888
		50	**0**/3006.9817	**10**/1.26860	**8**/0.96206
	20	20	**0**/5.92721	**1**/0.46515	**2**/0.56743
		50	**0**/2.92093	**33**/0.21127	**35**/0.15171
	30	20	**0**/4.23923	**4**/0.24810	**4**/0.35501
		50	**0**/2.74650	**53**/0.06008	**63**/0.11816
	40	20	**0**/3.22764	**3**/0.03547	**14**/0.04609
		50	**2**/2.45530	**73**/0.00438	**92**/0.00057
F_3	10	20	**1**/88.29054	**5**/1.26160	**6**/1.81855
		50	**1**/88.29054	**36**/0.99607	**24**/1.63858
	20	20	**0**/30.36538	**10**/0.58992	**37**/0.81628
		50	**0**/11.55541	**54**/0.42886	**60**/0.43778
	30	20	**1**/31.93949	**18**/0.31230	**47**/0.49963
		50	**1**/13.60421	**81**/0.19661	**82**/0.28853
	40	20	**0**/18.76824	**29**/0.21593	**63**/0.39942
		50	**1**/5.64692	**90**/0.06410	**88**/0.13929
F_4	10	20	**1**/0.25129	**7**/0.01322	**18**/0.01153
		50	**1**/0.25129	**26**/0.01095	**27**/0.01047
	20	20	**0**/0.05075	**12**/0.00932	**15**/0.01206
		50	**0**/0.02451	**22**/0.00676	**20**/0.01193
	30	20	**1**/0.04892	**13**/0.00965	**19**/0.01083
		50	**1**/0.02640	**26**/0.00710	**27**/0.01070
	40	20	**0**/0.03243	**14**/0.00801	**35**/0.00779
		50	**1**/0.01320	**33**/0.00583	**38**/0.00649

Table 4: The XOR broblem

Pattern no.	Input		Desired output
p	x_1	x_2	T
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

At the beginning there is a strongly random and diverse population and the crossover tends to explore the search space wildly, therefore resulting in low success rates and several large-error individuals. The PSO appeared to rapidly converge during the beginning of the search, but around global optimum the search process becomes slow. Without a selection operator, individuals tend to follow the best one and get into a neighborhood of the optimum. This results in a low error in average but not so many solutions successfully found. In contrast, the proposed HEA could utilize the widespread search of the BLX at the beginning and the fine-tuning ability of the PSO when it gets near to the optimum. With a selection process to eliminate inferior individuals, the HEA thus balances between finding solution (successfully) and obtaining a low averaged cost, especially when K is small. There are some situations that the HEA is not much better or even worse than the PSO. This is apparently

Table 5: Success rate [%] and mean error (**succ. rate**/mean error) for the NN training with $K = 50, 100$

N	K	GA	PSO	HEA
10	50	**0**/51.09702	**0**/0.72811	**0**/0.89564
	100	**0**/51.09702	**1**/0.43889	**1**/0.38902
20	50	**0**/4.54599	**0**/0.41958	**0**/0.28397
	100	**0**/3.87397	**13**/0.14848	**24**/0.05066
30	50	**0**/4.36849	**0**/0.34127	**1**/0.25869
	100	**0**/2.82969	**23**/0.08107	**43**/0.03092
40	50	**0**/2.41389	**0**/0.21741	**2**/0.10718
	100	**0**/1.79364	**37**/0.05251	**69**/0.00970

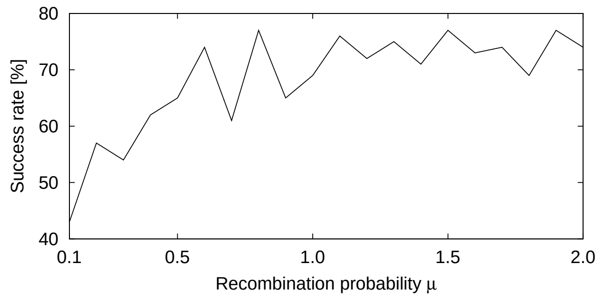


Figure 3: Performance vs. recombination probability μ in the HEA (in the case of the NN optimization by the HEA–$(0.6, 0.8)$ with $N = 30$, $K = 100$)

due to the low recombination probability μ used. The performance of the HEA can be improved when using higher value of μ. The impact of the recombination to its performance is shown in Fig. 3 for the NN optimization, for example. It appears that a high probability μ usually (but not always) provides better performance. For a particular problem, an optimal value of μ is able (and needed) to be determined.

In this study, although the proposed HEA could obtain good performance, it is necessary to validate the algorithm with more complex problems in future work.

References

[1] F. Grimaccia, M. Mussetta, R.E. Zich, "Genetical Swarm Optimization: Self-adaptive hybrid evolutionary algorithm for electromagnetics", *IEEE Trans. Antennas and Propagation* Vol.55, Iss.3, pp.781-785, 2007.

[2] J. Kennedy, R.C. Eberhart, "Particle swarm optimization", *Proc. of IEEE Intl. Conference on Neural Networks*, Piscataway, NJ. pp. 1942-1948, 1995.

[3] K.E. Parsopoulos, M.N. Vrahatis, "On the computation of all global minimizers through Particle swarm optimization", *IEEE Trans. on Evolutionary Computation*, Vol. 8, Iss. 3, pp. 211-224, 2004.

[4] L.J. Eshelman, D.J. Schaffer, "Real-coded genetic algorithms and interval-schemata", in L.D. Whitley (ed.) Foundations of genetic algorithms 2, Morgan Kaufmann, pp 187-202, 1993.