An Island Based Hybrid Evolutionary Algorithm for Optimization

Changhe Li and Shengxiang Yang

Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, UK
{cl160, s.yang}@mcs.le.ac.uk

Abstract. Evolutionary computation has become an important problem solving methodology among the set of search and optimization techniques. Recently, more and more different evolutionary techniques have been developed, especially hybrid evolutionary algorithms. This paper proposes an island based hybrid evolutionary algorithm (IHEA) for optimization, which is based on Particle swarm optimization (PSO), Fast Evolutionary Programming (FEP), and Estimation of Distribution Algorithm (EDA). Within IHEA, an island model is designed to cooperatively search for the global optima in search space. By combining the strengths of the three component algorithms, IHEA greatly improves the optimization performance of the three basic algorithms. Experimental results demonstrate that IHEA outperforms all the three component algorithms on the test problems.

1 Introduction

Evolutionary computation has become an important search and optimization technique for many researchers. The population-based parallel computation, collective learning process, self-adaptation, and robustness are some of the key features of evolutionary algorithms (EAs). EAs have been widely applied for solving important practical problems in engineering, business, and commerce, etc., yet in practice sometimes they deliver only marginal performance. There is little reason to expect that one can find a uniformly optimal algorithm for solving all optimization problems. This is in accordance with the No Free Lunch theorem [1], which explains that for any algorithm, any elevated performance over one class of problems is exactly paid for in performance over another class. Recently, the hybridization of EAs is becoming more and more popular due to their capabilities in handling several real world problems that involve complexity, noisy environments, imprecision, uncertainty, and vagueness.

As reported in the literature, several heuristical techniques have been used to improve the general efficiency of EAs. Zmuda et al. [2] introduced a hybrid evolutionary learning scheme for synthesizing multi-class pattern recognition systems. Wang [3] developed a hybrid approach to improve the performance of EAs for a simulation optimization problem. A hybrid evolutionary Particle Swarm Optimization (PSO) method was proposed by Shi et al. [4]. The hybrid approach

executes two systems simultaneously and selects P individuals from each system for exchanging after the designated N iterations. The individuals with a larger fitness have more opportunities of being selected. A hybrid technique that combines GA and PSO, called genetic swarm optimization (GSO), was proposed by Grimaldi et al. [5] for solving an electromagnetic optimization problem. Li and Wang et al. [6, 7] proposed a hybrid PSO using Cauchy mutation to reduce the probability of trapping local optima for PSO.

In this paper, an island based hybrid evolutionary algorithm(IHEA) is proposed based on Particle Swarm Optimization (PSO), Fast Evolutionary Programming (FEP) and Estimation of Distribution Algorithm (EDA). An island model using different evolutionary strategies is designed for improving the optimization performance of the component algorithms. A set of function optimization benchmark problems are tested in this paper.

The rest of the paper is organized as follows. Section 2 briefly describes PSO, FEP, and EDA optimization techniques. The proposed algorithm, IHEA, is presented in details in Section 3. Experimental results are shown in Section 4. Finally, some conclusions are given in Section 5.

2 Three Different Evolutionary Techniques

2.1 Particle Swarm Optimization

Particle swarm optimization (PSO) was first introduced by Kennedy and Eberhart in 1995 [8, 9]. PSO is motivated from the social behavior of organisms, such as bird flocking and fish schooling. Particles "fly" through the search space by following the previous best positions of their neighbors and their own previous best positions. There are several main versions of the PSO algorithms. The following version modified by Shi and Eberhart [10] is used in this paper. Each particle is represented by a position and a velocity, which are updated as follows:

$$\boldsymbol{V}'_i = \omega \boldsymbol{V}_i + \eta_1 r_1 (\boldsymbol{P}_i - \boldsymbol{X}_i) + \eta_2 r_2 (\boldsymbol{P}_g - \boldsymbol{X}_i) \tag{1}$$

$$\boldsymbol{X}'_i = \boldsymbol{X}_i + \boldsymbol{V}'_i \tag{2}$$

where $\boldsymbol{X}'_i$ and $\boldsymbol{X}$ represent the current and previous positions of particle i, $\boldsymbol{V}_i$ and $\boldsymbol{V}'_i$ are the previous and current velocity of particle i, $\boldsymbol{P}_i$ and $\boldsymbol{P}_g$ are the best-so-far position of particle i and the best position found in the whole swarm so far respectively. $\omega \in (0, 1]$ is an inertia weight which determines how much the previous velocity is preserved, η_1 and η_2 are acceleration constants, and r_1 and r_2 are random numbers generated from the interval [0.0, 1.0].

The framework of the PSO algorithm is given as follows:

Step 1: Generate the initial particles by randomly generating the position and velocity for each particle.
Step 2: Evaluate the fitness of each particle.
Step 3: For each particle i, if its fitness is smaller than the fitness of its previous best position ($\boldsymbol{P}_i$), update $\boldsymbol{P}_i$.

Step 4: For each particle, if its fitness is smaller than the fitness of the best position ($\boldsymbol{p}_g$) of all particles, update $\boldsymbol{P}_g$.
Step 5: Update each particle according to Eqs. (1) and (2).
Step 6: Stop if the stop criterion is satisfied; otherwise, go to Step 3.

From the mathematic theoretical analysis of the trajectory of a PSO particle [11], the trajectory of a particle $\boldsymbol{X}_i$ converges to a weighted mean of $\boldsymbol{P}_i$ and $\boldsymbol{P}_g$. Whenever the particle converges, it will "fly" to the personal best position and the global best particle's position. This information sharing mechanism makes PSO have a very fast speed of convergence. Meanwhile, because of this mechanism, PSO can't guarantee to find the global minimal value of a function.

2.2 Fast Evolutionary Programming

Fast Evolutionary Programming (FEP) was proposed by Yao [12] by introducing Cauchy mutation instead of Gaussian mutation in classical EP (CEP). The framework of FEP is as follows:

Step 1. Generate initial population of μ individuals. Each individual is represented by a pair of real-valued vectors $(x_i, \eta_i), \forall i \subset \{1, 2, ..., \mu\}$.
Step 2. Evaluate the fitness of each individual (x_i, η_i).
Step 3. For each parent $(x_i, \eta_i), \forall i \subset \{1, 2, ..., \mu\}$, create a single offspring (x'_i, η'_i), $\forall i \subset \{1, 2, ..., \mu\}$ by:

$$x'_i(j) = x_i(j) + \eta_i(j)\delta_j \tag{3}$$

$$\eta'_i(j) = \eta_i(j)exp(\tau' N(0,1) + \tau N_j(0,1)) \tag{4}$$

where $x_i(j)$, $x'_i(j)$, $\eta_i(j)$, and $\eta'_i(j)$ denote the $j-th$ component of individual (x_i, η_i) respectively. $N(0,1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one. $N_j(0,1)$ indicates that the random number is generated anew for each value of j. δ_j is a Cauchy random variable with the scale parameter 1 and is generated anew for each value of j. The factor τ and τ' are usually set to $(\sqrt{2\sqrt{n}})^{-1}$ and $\sqrt{2n}^{-1}$ [13, 14].
Step 4. Calculate the fitness of each offspring $(x'_i, \eta'_i), \forall i \subset \{1, 2, ..., \mu\}$.
Step 5. Conduct pairwise comparison over the union of parents (x'_i, η'_i) and offspring $(x'_i, \eta'_i), \forall i \subset \{1, 2, ..., \mu\}$. For each individual, q opponents are chosen uniformly at random from all the parents and offspring. For each comparison, if the individual's fitness is no smaller than the opponent's, it receives a "win".
Step 6. Select μ individuals from the parents and offspring that have the most wins to next generation.
Step 7. If stop criterion is satisfied, then stop; otherwise, go to Step 3.

Yao et al. [15] explains that Cauchy mutation performs better than Gaussian mutation for most tested benchmark problems used in [15] because of its higher probability of making longer jumps (the larger the search step size, the faster the algorithm gets to the global optimum). However, the problem of FEP is that a large step size may not be beneficial at all if the current search point is already very close to the global optimum.

2.3 Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) [16, 17] are non-deterministic, stochastic heuristic search strategies that form part of the evolutionary computation approaches. Within EDAs, a number of solutions or individuals are created every generation according to a distribution model, which evolves generation by generation until a satisfactory solution is achieved. In brief, the characteristic that most differentiates EDAs from other evolutionary search strategies, such as genetic algorithms, is that the evolution from a generation to the next one is done by estimating the probability distribution of the fittest individuals, and afterwards by sampling the induced model. This avoids the use of crossover or mutation operators, and the number of parameters that EDAs require is reduced considerably.

The framework of EDAs is as follows:

Step 1. Select M promising individuals from parent population to form the parent set Q by a selection method (e.g., the truncation selection).
Step 2. Build a probabilistic model $p(x)$ based on the statistical information extracted from the parent set Q.
Step 3. Sample offspring according to the constructed probability model $p(x)$.
Step 4. Fully or partly replace individuals in parent population by sampled offspring to form the next generation.

In this paper, we use truncation selection method as most literatures used. Another major issue in EDA is how to build a probability distribution model $p(x)$. we use Gaussian model with diagonal covariance matrix (GM/DCM) [17]. In GM/DCM, the joint density function of the k-th generation is described as follows:

$$p_k(x) = \prod_{i=1}^{n} \mathcal{N}(x_i; \mu_i^k, \sigma_i^k) \tag{5}$$

where

$$\mathcal{N}(x_i; \mu_i^k, \sigma_i^k) = \frac{1}{\sqrt{2\pi}\sigma_i} e^{\frac{1}{2}(\frac{x_i-\mu_i}{\sigma_i})^2} \tag{6}$$

In Equ(5), the n-dimensional joint probability distribution is factorized as a product of n univariate and independent normal distributions. There are two parameters for each variable required to be estimated in the k-th generation: the mean $\hat{\mu}_i^k$, and the standard deviation, $\hat{\sigma}_i^k$. They can be estimated as follows:

$$\hat{\mu}_i^k = \bar{X}_i^k = \frac{1}{M}\sum_{t=1}^{M} x_{t,i}^k; \hat{\sigma}_i^k = \sqrt{\frac{1}{M}\sum_{t=1}^{M}(x_{t,i}^k - \bar{X}_i^k)^2} \tag{7}$$

where $(x_{1,i}^k, x_{2,i}^k, ..., x_{M,i}^k)$ are values of the i-th variable of the selected M parent individuals in the k-th generation.

Experimental results show that EDA gives distinctive performance on some particular problems. However, it doesn't help for the search on some other problems at all. In EDA, the search is mainly based on global information used for

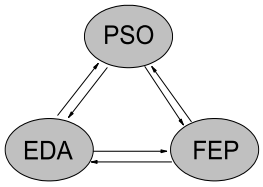

Fig. 1. Migration topology of IHEA.

guiding the exploration. However, too much exploration may cause no convergence problem.

3 Island Based Hybrid Evolutionary Algorithm

The main idea of IHEA is that migration of individuals among different islands can increase diversity of each island, so it can reduce the probability of premature convergence. In IHEA, there are three subpopulation residing in three different islands, which use PSO, FEP and EDA algorithms respectively to search global optima in the whole shared search space. Since different islands use different evolutionary techniques, they probably follow different search directions in the whole shared search space, that is they explore different area in the whole search space. However, they are not independent to search, they exchange their own updated information periodically between each other by migration of promising individuals. The information sharing mechanism is helpful to search unexplored space where probably global optima is.

PSO in IHEA can be regarded as a fast local search operator for exploitation, EDA is used for exploring new promising area in the whole search space. FEP can be taken as mutation operator because of its long jump capability. The cooperative search among the three islands is helpful for them to explore new promising areas. It greatly reduces the probability of premature convergence, hence the global search capability is improved.

The migration topology of IHEA is described in Fig. 1. The main framework of IHEA is as follows:

Step 1. Initialize the three population, set $k = 1$.
Step 2. Use PSO, EDA and FEP to optimize each population.
Step 3. Compare the best individual of each population, if the best individual of population p is better than the best of the other two population, migrate individuals of population p to the other two population whose fitness is better than those individuals of the other two population.
Step 4. Stop if stop criterion is true; otherwise, $k = k + 1$ and goto Step 2.

Another big issue in IHEA is the population resource allocation among different islands. As we know, different problems may have totally different landscapes,

Table 1. Details of the test functions, where n and f_{min} are the dimension and the minimum value of a function respectively and $S \in R_n$.

Test Function	n	S	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$(-5.12, 5.12)$	0
$f_2(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$	2	(-2.048, 2.048)	0
$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos(\frac{x_i - 100}{\sqrt{i}}) + 1$	30	(-300, 300)	0
$f_4(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) -$ $\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	(-30, 30)	0
$f_5(x) = \sum_{i=1}^{n} 100(x_{i+1}^2 - x_i)^2 + (x_i - 1)^2)$	30	(-2.048, 2.048)	0
$f_6(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	(-5.12, 5.12)	0
$f_7(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	(-500, 500)	-12569.5
$f_8(x) = 418.9829 \cdot n + \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	(-500, 500)	0

Table 2. Population size of the three islands in IHEA algorithm

Island	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
PSO	10	15	10	10	15	5	10	10
FEP(q)	5(3)	10(4)	15(5)	5(3)	10(4)	10(4)	15(5)	15(5)
EDA	15	5	5	15	5	15	5	5

such as the number of local or global optima, the difference among local optima, the location of local or global optima, and so on. Different problems may need different algorithms to solve it. This can be observed from the result presented in Section 4.2. Hence, we should allocate different population resources on different islands. For example, we can allocate the most population resources to the island that is more effective than other algorithms to solve the problem.

4 Experimental study

4.1 Test problems and experimental settings

Eight benchmark functions ($f_1 - f_8$) are used in this paper. Functions $f_1 - f_2$ is unimodal functions while $f_3 - f_8$ have many local optima. Table 1 gives the details of these functions. Algorithm parameters are set as follows: For PSO algorithm, the acceleration constants of η_1 and η_2 are both set to be 1.496180 and the inertia weight $\omega = 0.729844$ as suggested by den Bergh [18]. In FEP, the tournament size is 5 for selection and initial standard deviation is 3.0 as used in [12]. For all algorithms , population size is 30. The subpopulation size of FEP, EDA and PSO is different for different problem in IHEA algorithm, details can be seen from Table 2, however, the values of population size are just experimental values not optima values. we run 50 times independently till generation 1500 for all test problems.

Table 3. Comparison of PSO, EDA, FEP and IHEA. The results are mean best function values found at generation 1500 over 50 runs

Test function	PSO	EDA	FEP	IHEA
f_1	1.218e-021	0	1.0219	0
f_2	0	0.3835	3.26304e-005	0
f_3	0.0291024	54.8026	1.29825	0.0559343
f_4	2.09482	4.44089e-016	4.77732	4.44089e-016
f_5	9.09728e-012	2545.73	80.2489	1.71309e-009
f_6	50.7826	0	26.0043	0
f_7	-7431.63	-2343.9	-12307	-12522.4
f_8	5043.08	10348.7	314.947	71.0184

Table 4. The T-test results between IHEA and the other three algorithms,where "***" means the results of each run is the same

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
PSO	-1.17951	***	3.13962	-12.9764	2.03565	-26.3966	-73.2857	-71.1316
EDA	***	-13.8161	-108.133	0	-250.172	***	-137.587	-183.923
FEP	-4.18079	-1.31905	-10.73	-14.7223	-9.02354	-21.3865	-6.90319	-7.33192

4.2 Performance Comparison

The average best results of the four algorithms are shown in Table 3. Table 4 shows the statistical comparison of the IHEA algorithm over the other three algorithms, using the two-tailed T-test with 98 degrees of freedom at a 0.05 level of significance. In all the T-test results of the paper, the performance difference is significant if the absolute value of the T-test result is greater than 1.984.

From Table 3 and Table 4, we can see that PSO, EDA and FEP give quite different performance on different test cases. PSO obtains the best results on function f_2, f_3 and f_5, EDA gives the best results on function f_1, f_4 and f_6, while FEP presents the best results on function f_7 and f_8. However, when compare the three algorithms with IHEA, we can find that IHEA shows the best performance on all test problems except function f_3 and f_5, which is slightly worse than the results of PSO. IHEA greatly improves the performance of the three original algorithms.

Fig 2 also shows the evolutionary process of the four algorithms. IHEA gives the fastest convergence on most test problems. From the results of Table 3 and Fig 2, we can conclude that IHEA reduces the probability of premature convergence for most tested problems, especially for multimodal functions.

Table 5 and Table 6 show the average results of average mean best and T-test results of all the test problems of 10 dimensions except function f_2. Form the results, we can see that all the results obtained by IHEA are better than that of the other three algorithms except on function f_5, where the result of IHEA is slightly worse than that of PSO.

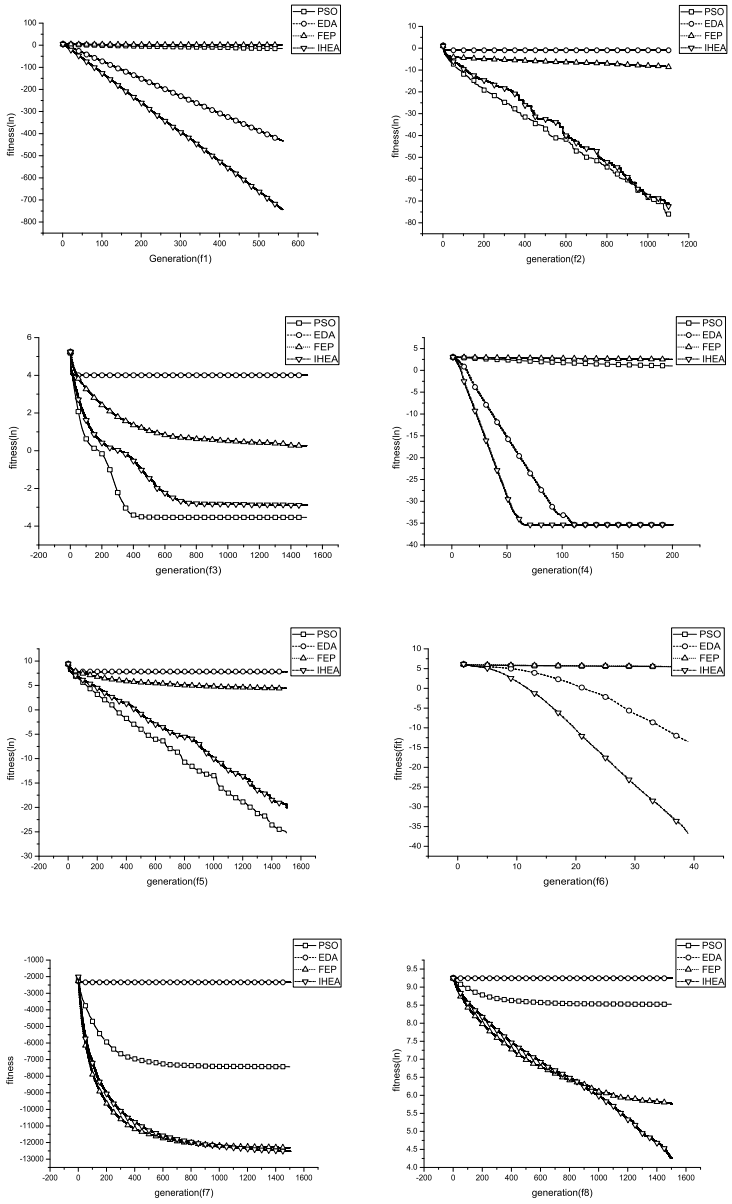

Fig. 2. Evolution process of the average best particle of PSO, EDA, FEP, and IHEA.

Table 5. Comparison of PSO, EDA, FEP and IHEA on problems of 10 dimensions. The results are mean best function values found at generation 1500 over 50 runs

Test function	PSO	EDA	FEP	IHEA
f_1	6.98296e-072	0	0.00744029	0
f_3	0.0866419	11.5354	0.18873	0.073003
f_4	4.06785e-015	4.44089e-016	0.575907	4.44089e-016
f_5	1.97215e-031	599.55	0.0918242	4.19082e-030
f_6	6.18864	0	6.41484	0
f_7	-3431.8	-1317.14	-4177.4	-4189.83
f_8	725.257	2881.37	10.2413	0.000127276

Table 6. The T-test results between IHEA and the other three algorithms on problems of 10 dimensions,where "***" means the results of each run is the same

	f_1	f_3	f_4	f_5	f_6	f_7	f_8
PSO	-1.90455	-1.84457	-50.9999	3.20954	-15.5783	-21.8944	-20.3932
EDA	***	-37.4252	0	-51.6912	***	-80.4453	-66.1889
FEP	-1.94113	-2.24096	-4.50941	-1.67656	-11.619	-2.54557	-1.97234

5 Conclusions

This paper discusses an effective method to reduce the probability of trapping local optima for evolutionary algorithm, which employs hybrid method to improve algorithm performance. An island based hybrid algorithm (IHEA) is proposed based on PSO, EDA and FEP algorithms. In IHEA, three different islands were designed using PSO, EDA and FEP as their local search method respectively. An information sharing mechanism is developed among the three islands. A set of benchmark problems were tested. From the result, it can be seen that IHEA greatly reduces the probability of becoming trapped at a local optimum. The global search capability of IHEA is better than the three original algorithms.

In the future, we will focus on the further study of IHEA in deep level. There are some issues that should be considered, such as, the resources distribution of different islands, the migration topology and algorithms used on different islands.

Acknowledgement

This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/1.

References

1. D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.*, 1(1): 67-82, 1997.

2. MA. Zmuda, MM. Rizki, and LA. Tamburino. Hybrid evolutionary learning for synthesizing multi-class pattern recognition systems. *Applied Soft Computing*, 2(4): 269-282, 2003.
3. L. Wang, A hybrid genetic algorithm-neural network strategy for simulation optimization, *Applied Mathematics and Computation*, 170(2): 1329-1343, 2005.
4. XH. Shi, YC. Liang, HP. Lee, C. Lu, and LM. Wang, An improved GA and a novel PSO-GA-based hybrid algorithm, *Information Processing Letters*, 93(5): 255-261, 2005.
5. EA. Grimaldi, F. Grimacia, M. Mussetta, P. Pirinoli, and RE. Zich. A new hybrid genetical C swarm algorithm for electromagnetic optimization, *Proc. of Int. Conf. on Computational Electromagnetics and its Applications*, pp. 157-160, 2004.
6. C. Li, Y. Liu, L. Kang, and A. Zhou. A Fast Particle Swarm Optimization Algorithm with Cauchy Mutation and Natural Selection Strategy. *ISICA2007, LNCS4683*, pp. 334-343, 2007.
7. H. Wang, Y. Liu, C. Li, and S. Zeng, A Hybrid Particle Swarm Algorithm with Cauchy Mutation, *Proc.of the 2007 IEEE Swarm Intelligence Symposium*, 2007.
8. R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. *Proc. of the 6th Int. Symp. on Micro Machine and Human Science*, pp. 39-43, 1995.
9. J. Kennedy and R. C. Eberhart. Particle Swarm Optimization. *Proc. of the 1995 IEEE Int. Conf. on Neural Networks*. pp. 1942-1948, 1995.
10. Y. Shi and R. C. Eberhart. A Modified Particle Swarm Optimizer. *Proc. of the IEEE Int. Conf. on Evol. Comput.*, pp. 69-73, 1998.
11. M. Clerc and J. Kennedy, The Particle Swarm: Explosion, Stability and Convergence in a Multi-Dimensional Complex Space, *IEEE Trans. on Evol. Comput.*, 6(1): 58-73, 2002.
12. X. Yao and Y. Liu. Fast evolutionary programming, *Proc. of the 5th Annual Conference on Evolutionary Programming (EP'96)*, pp. 451-460, 1996.
13. T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1): 1-23, 1993.
14. D. B. Fogel. An introduction to simulated evolutionary optimization, *IEEE Trans. Neural Networks*, 5(1): 3-14, 1994.
15. X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Trans. on Evol. Comput.*, **3**(1): 82-102, 1999.
16. B.-T. Zhang. A Bayesian framework for evolutionary computation. *Proc. of the 1999 Congress on Evol. Comput.*, pp. 722-728, 1999.
17. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2001.
18. F. van den Bergh. An Analysis of Particle Swarm Optimizers. *PhD thesis*, Department of Computer Science, University of Pretoria, South Africa, 2002.