An EA framework for uncertain optimization problem

M. Bhattacharya

SOBIT, Charles Sturt University, NSW, Australia

Abstract

Many engineering design projects involve solving complex optimization problems. Evolutionary algorithms (EA) have been widely accepted as efficient optimizers for complex real life problems [2]. However, many real life optimization problems involve time-variant noisy environment, which pose major challenges to EA-based optimization. Presence of noise interferes with the evaluation and the selection process of EA and adversely affects the performance of the algorithm [6]. Also presence of noise means fitness function can not be evaluated and it has to be estimated instead. Several approaches have been tried to overcome this problem, such as introduction of diversity (hyper mutation, random immigrants, special operators) or incorporation of memory of the past (diploidy, case based memory) [5]. In this paper we propose a method, DPGA (distributed population genetic algorithm) that uses a distributed population based architecture to simulate a distributed, self-adaptive memory of the solution space. Local regression is used in each sub-population to estimate the fitness. Specific problem category considered is that of optimization of functions with time variant noisy fitness. Successful applications to benchmark test problems ascertain the proposed method's superior performance in terms of both adaptability and accuracy.

Keywords: Optimization, evolutionary algorithm, uncertain environment

1. Introduction

Evolutionary algorithm's applications to optimization problems involving noisy or uncertain fitness functions pose problems that require attention. Not surprisingly, many real world applications actually fall in this category. Two important examples of such applications are:

- Online adaptation of real world systems. In such systems some design parameters can be decided only through experiments and real time.

- Simulation based optimization of large and complex systems that uses random numbers in the simulation. Use of random numbers causes fluctuation of the fitness value obtained by simulation.

Presence of noise in the fitness function could mean 'overestimation' of fitness of inferior candidates and similarly 'underestimation' of fitness of superior candidates at the time of selection. This is likely to result in reduced learning rate, inability to retain learnt information, limited exploitation and absence of gradual monotonous improvement of fitness with generations.

The effects of noise can be reduced by simple methods such as re-sampling and large population size. However, these methods are impractical if there is a strict time limit (interpreted as limits on the total number of fitness evaluations) as fitness evaluation can be highly expensive in many real life problem domains.

Some of the major techniques used to curb the effect of uncertainty in EA are: *Re-sampling* [6, 4, 5], *Conventional EA with increased population size* [4, 6], *Rescaled mutation* [1], *Thresholding* [6, 5], *Fitness value based on neighbouring individuals* [6, 9, 3, 8]

Algorithm 1: *Procedure Distributed Population GA*

1: **begin**
2: $t = 0$
3: Initialize population $P(t)$
4: Evaluate population $P(t)$
5: **while** (not<termination condition>)
6: **begin**
7: $t = t + 1$
8: **if** (reorganizing generation)
9: **begin**
10: Evaluate population $P(t)$ using re-sampling
11: Evolve population $P(t)$ to create new generation with canonical GA mechanism
12: Self-organize new population to create pseudo-populations
13: **end if**
14: **for** (each eligible one of n pseudo-populations $P(t)_{N=1,2,\ldots,n}$)
15: **begin**
16: Evaluate population $P(t)_N$ partly by local regression
17: Evolve canonical as per population GA
18: **end for**
19: **end main**

Fig. 1. The proposed DPGA architecture.

and *Reduced resampling* [7, 6]. A comprehensive survey of various techniques to handle noisy environment with EA, including distributed population approach similar to our current work can be found in [5].

Our current research focuses on uncertain problem domains where sample size for resampling or fitness evaluation time is not of as much concern but accuracy and robustness of solution is. The proposed DPGA framework involves a distributed population based architecture with local regression employed in the sub-populations. DPGA framework aims at generating accurate and robust solution in uncertain environment and is suitable in problem domains where cost of evaluation is not an issue.

The rest of the paper is organized as follows: Section 2 details the proposed framework. Details of experiments are given in Section 3, while Section 4 presents the results and discussions. Finally, Section 5 concludes the paper.

2. The proposed algorithm architecture

The proposed DPGA framework uses distributed population architecture with local regression in the pseudo-populations. Distribution of the population allows tracking multiple peaks in the search space. Each pseudo-population maintains information about a separate region in the search space, thus acting as a distributed self adaptive memory. We use the term pseudo-population instead of subpopulation to emphasize on the fact that each one of these sub groups may not actually act as self-sufficient evolving populations. Two basic ideas guide the functioning of the proposed model:

- Retention of memory with distributed pseudo-populations in the search space i.e., the pseodo-populations should be able to track their moving peaks through time.
- Estimation of time variant noisy fitness by local regression in each subpopulation. Quadratic regression is used in the present work. However, linear regression could be used to reduce computation time.

The basic algorithm structure of DPGA is as described in Figure 1.

Specific features of the proposed distributed population GA are as below:

- Unlike conventional multipopulation EAs [5] the distributed population GA does not maintain a main population alongside the subpopulations. Instead, DPGA switches from single population to multipopulation periodically.
- At *reorganizing generation* the pseudo-populations are merged together to regain the main population and evolution is carried out as per canonical GA mechanism. To correct the adverse effects of noise, actual fitness evaluation along with resampling is used at this stage.
- Next the regenerated main population dissolves into pseudo-populations by self-organization. This is essentially distribution of the candidate solutions into pseudo-populations based on specific criteria. A factor of *fitness* and *population size* decides the eligibility of a pseudo-population to obtain evolution right. An eligible pseudo-population then evolves by the canonical GA mechanism. Local regression is used to estimate the fitness values. Mutation rate depends on a factor of *fitness* and *population size* as well.

- Members of the non-eligible pseudo-populations either survive or eventually disappear after *reorganizing generations*.
- Retention of memory about the *moving peaks* is achieved through the pseudo-populations over specific durations until *reorganizing generations*. This is logical considering the dynamic nature of the phenotypic solution space.

Some simulation details for DPGA are presented in Section 3 and Section 4.

3. Experiment details

In this research, we investigated the performance of the proposed DPGA model as against standard GA, differential evolution, particle swarm optimization for noisy benchmark functions (see Table 1). For analysis purpose, experiments have been conducted on the non-noisy versions of the same set of benchmark functions as well.

3.1. Test functions

We have used the following benchmark functions to test the algorithms: Sphere function (5D), Griewank's function (50D), Rastrigin's function 1 (50D) and the Rosenbrock's function (50d). These benchmark functions have previously been used by researchers to test performance of evolutionary algorithm for noisy optimization problems [3]. All the benchmarks used are minimization problems. The descriptions of the functions are as below:
Sphere function (5 Dimensional):

$$f_{Sphere}(\vec{x}) = \sum_{i=1}^{5} x_i^2 \text{ with } -100 \le x_i \le 100 \quad (1)$$

Griewank's function (50 Dimensional):

$$f_{Griewank}(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{50}(x_i - 100)^2 - \prod_{i=1}^{50}\cos\left(\frac{x_i - 100}{\sqrt{i}}\right) + 1 \text{ with } -600 \le x_i \le 600 \quad (2)$$

Rastrigin's Function 1 (50 Dimensional):

$$f_{RastriginF1}(\vec{x}) = 200 + \sum_{i=1}^{50} x_i^2 - 10 \cdot \cos(2\pi x_i)$$

with $-5.12 \le x_i \le 5.12$ (3)

Rosenbrock's function (50 Dimensional):

$$f_{Rosenbrock}(\vec{x}) = \sum_{i=1}^{49}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$$

with $-50 \le x_i \le 50$ (4)

The noisy versions of the above set of functions are defined as:

$$f_{Noisy}(\vec{x}) = f(\vec{x}) + N(\mu, \sigma^2) \quad (5)$$

where, $N(\mu, \sigma^2)$= Standard Normal (or Gaussian) distribution with mean, $\mu = 0$ and variance, $\sigma^2 = 1$. The probability density function $f(x; \mu, \sigma^2)$ is given as below,

$$f(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (6)$$

3.2. Experiment settings

The parameter settings for the algorithms are as shown in Table 1. For comparison purpose we have used the results reported in [6] for conventional GA, differential evolution (DE) and particle swarm optimization (PSO). Hence similar parameter settings have been used. As Krink et al. in [6] have reported, the parameters of the heuristic algorithms were not tuned for each test problem. The reason stated is that in case of real life problems such tuning can be infeasible due to time constraints.

Let $totalEval$ = total number of function evaluations, $popSize$ = population size, $totalIT$ = total number of iterations, rs = total number of candidate solution resampling, and $totalIUnchanged$ =total number of individuals that remained unchanged (such as the elites) and were not reevaluated in the various generations. Then,

$$totalEval = popSize * totalIT * rs - totalIUnchanged \quad (7)$$

The total number of function evaluations in [14] in case of conventional GA, differential evolution and PSO is kept constant by keeping $totalIT \propto \frac{1}{rs}$. However, this has not been followed for the proposed method. We have used a fixed number of iteration in these experiments. The experiments were run with different values of rs such as rs =1, 5, 20, 50 and

100, to eradicate the effects of noise and find the 'true' fitness. The 'true' fitness here refers to the fitness value obtained by evaluation of the non-noisy versions of the same functions.

4. Results and discussions

Table 2 to Table 9 present the final results obtained for the chosen benchmark functions (both noisy and non-noisy versions). The results for the non-noisy versions of the functions have been reported here mainly as 'standards' to judge the impact of noise. For comparison purpose we have used the results reported in [6] for the following methods: differential evolution, particle swarm optimization (PSO) and conventional EA (CGA). As mentioned earlier, the total number of function evaluations has been kept fixed for the experiments with differential evolution, PSO and conventional EA [6]. The proposed method used variable number of function evaluations.

A comparison of total number of function evaluations is given in Table 10.

Not surprisingly all the heuristics have performed considerably better on the non-noisy benchmark functions compared to the noisy versions of the same functions. As can be observed from the results (Table 2 to Table 9), while the proposed method has performed better for majority of the test cases, the difference is not necessarily significant in all cases of the low dimensional Sphere function. However, for the high dimensional (50D) Griewank, Rastrigin and Rosenbrock functions the differences are clearly significant. In the remaining part of this section we will focus our discussions to performances related to the noisy functions.

Table 1
Parameter Settings* of the Algorithms

Proposed Method	Conventional GA	Differential Evolution	PSO
popSize=100	popSize=100	popSize=50	popSize=20
p_c =1.0	p_c =1.0	CF =0.8	w = 1.0 $\rightarrow$ 0.7
p_m =0.3	p_m =0.3	f =0.5	$\varphi_{\min}$ =0.0
n =10	n =10		$\varphi_{\max}$ =2.0
σ_m =0.01	σ_m =0.01		

*popSize=Population Size, p_c = Crossover rate (EA), p_m = Mutation Rate (EA), n = Number of Elites, σ_m = Mutation Variance, CF = Crossover Factor (in differential Evolution), f =Scaling Factor, w =Inertia Weight (the value is linearly decreased from 1.0 to 0.7 during the run), $\varphi_{\min}$, $\varphi_{\max}$ = Lower and Upper Bounds of the Random Velocity Rule Weights.

Table 2
The results (mean $\pm$ standard error) for f_1 =Sphere function (5D non-noisy), f^*_1 =Sphere function (5D noisy), rs =number of candidate solution resampling.

	f_1	$f_1^*, rs = 1$	$f_1^*, rs = 5$
Proposed Method	4.21334E-75±0	0.00103E-2±0.003	0.00023E-2±0.001
CGA	6.71654E-20±0	0.04078±0.00543	0.02690±0.00363
Differential Evolution	4.12744E-152±0	0.25249±0.02603	0.13315±0.01266
PSO	2.51130E-8±0	0.36484±0.05182	0.16702±0.03072

Table 3
The results (mean $\pm$ standard error) for f_1 =Sphere function (5D non-noisy), f^*_1 =Sphere function (5D noisy), rs =number of candidate solution resampling.

	$f_1^*, rs = 20$	$f_1^*, rs = 50$	$f_1^*, rs = 100$
Proposed Method	0.00018E-2±0.023	0.00001E-2±0.0031	0.00011E-2±0.0009
CGA	0.02205±0.00290	0.01765±0.00233	0.03929±0.00396
Differential Evolution	0.07364±0.00811	0.07004±0.00686	0.08165±0.00800
PSO	0.11501±0.01649	0.06478±0.00739	0.07135±0.00938

Table 4
The results (mean $\pm$ standard error) for f_2 =Griewank function (50D non-noisy), f_2^* =Griewank function (50D noisy), rs =number of candidate solution resampling.

	f_2	$f_2^*, rs =$ 1	$f_2^*, rs =$ 5
Proposed Method	4.00E-7±0.001	0.00211E-1±0.001	0.00211E-1±0.001
CGA	0.00624±0.00138	1.14598±0.00307	1.10223±0.00342
Differential Evolution	0±0	3.31514±0.07388	2.42183±0.03616
PSO	1.54900±0.06695	11.2462±0.50951	16.6429±0.70800

Table 5
The results (mean ± standard error) for f_2 =Griewank function (50D non-noisy), f_2^* =Griewank function (50D noisy), rs =number of candidate solution resampling.

	$f_2^*, rs =$ 20	$f_2^*, rs =$ 50	$f_2^*, rs =$ 100
Proposed Method	0.00011E-1±0.021	0.00200E-1±0.323	0.00301E-1±0.481
CGA	1.44349±0.01381	3.69626±0.13127	18.0858±0.99646
Differential Evolution	2.67093±0.03895	46.8197±0.96449	233.802±6.25840
PSO	85.4865±2.13148	143.021±2.33228	194.188±4.90959

Table 6
The results (mean ± standard error) for f_3 =Rastrigin's function 1 (50D non-noisy), f_3^* =Rastrigin's function 1 (50D noisy), rs =number of candidate solution resampling.

	f_3	$f_3^*, rs =$ 1	$f_3^*, rs =$ 5
Proposed Method	1.219E-7±0.0013	0.03011E-1±0.031	0.01071E-1±0.011
CGA	32.6679±1.94017	30.7511±1.32780	31.4725±2.02356
Differential Evolution	0±0	2.35249±0.06062	14.0355±0.47935
PSO	13.1162±1.44815	55.9704±2.19902	160.500±2.67500

Table 7
The results (mean ± standard error) for f_3 =Rastrigin's function 1 (50D non-noisy), f_3^* =Rastrigin's function 1 (50D noisy), rs =number of candidate solution resampling.

	$f_3^*, rs =$ 20	$f_3^*, rs =$ 50	$f_3^*, rs =$ 100
Proposed Method	0.01171±0.111	0.01092±0.100	0.01232±0.190
CGA	39.1777±2.11529	74.8577±2.69437	147.800±2.93208
Differential Evolution	167.628±2.12569	314.762±2.88650	438.036±3.67504
PSO	313.184±3.93659	380.178±4.88706	418.265±5.35434

Table 8
The results (mean ± standard error) for f_4 =Rosenbrock's function (50D non-noisy), f_4^* =Rosenbrock's function (50D noisy), rs =number of candidate solution resampling.

	f_4	$f_4^*, rs =$ 1	$f_4^*, rs =$ 5
Proposed Method	1.4011E-4±0.0011	1.4201E-4±0.0021	1.1201E-4±0.0091
CGA	79.8180±10.4477	118.940±13.2322	341.788±49.6738
Differential Evolution	35.3176±0.27444	47.6188±0.15811	47.0404±0.13932
PSO	5142.45±2929.47	4884.68±886.599	368512±39755.5

Table 9
The results (mean ± standard error) for f_4 =Rosenbrock's function (50D non-noisy), f_4^* =Rosenbrock's function (50D noisy),, rs =number of candidate solution resampling.

	$f_4^*, rs =$ 20	$f_4^*, rs =$ 50	$f_4^*, rs =$ 100
Proposed Method	1.0001±0.21913	1.0091±0.1393	2.0028±2.1999
CGA	1859.06±261.844	35477.7±4656.17	257488±19371.2
Differential Evolution	7917.46±352.851	1.65E+7±903677	2.98E+8±1.04E+7
PSO	1.61E+7±1.18E+6	5.57E+7±2.38E+6	1.17E+8±7.38E+6

Table 10
Total number of function evaluations used to attain the reported results with the proposed method, canonical GA, differential evolution and PSO for the various test cases.

	totalEval for 5D Sphere	*totalEval* for 50D Griewank	*totalEval* for 50D Rastrigin
Proposed Method	90000	430,000	450,000
CGA	100,000	500,000	500,000
Differential Evolution	100,000	500,000	500,000
PSO	100,000	500,000	500,000

Similar to the observation reported in [6], resampling has prevented stagnation in all the four test functions, regardless of whether the mean final result has been improved. However, the effect of rate of resampling is rather inconclusive as it varied from case to case rather randomly. Only conclusion that can be drawn is that the effect of rate of resampling may be problem dependent. From the results reported in [6] it is obvious that in case of conventional EA, increased number of resampling can not improve the performance when number of iterations is inversely proportional to number of resampling to keep the total number of function evaluations constant. However, interestingly enough, for most of the test cases moderate rate of resampling has helped to improve the solution, while high to very high rate of resampling has rather a deteriorating effect on the solutions.

As can be observed from Table 10, the proposed model has managed to produce far superior solutions with much fewer actual function evaluations.

5. Conclusions

Many industrial projects often involve complex optimization problems. Optimization problems involving uncertain environments are challenging to evolutionary algorithms as they require not only finding the optimum, but also to track the changing optimum over time. Numerous methods have been tried to face this challenge with varied degrees of success [5]. One such approach is that of the multipopulation approach where the subpopulations are used to retain information about the *moving peaks*. The proposed distributed population genetic algorithm or DPGA framework presented in this paper is similar to the multipopulation approach in that it divides the search or solution space into multiple pseudo-populations and retains information about the moving peaks in them. However, the DPGA algorithm applies superior mechanism in that this reduces the computational expense of maintaining the main population along with the subpopulations by switching between main and subpopulations at regular intervals. Also just enough retention of memory is allowed by dissolving the subpopulations periodically. The simulation results do ascertain the superior performance of the proposed algorithm. However, future research need to resolve some of the issues related to the proposed DPGA framework. Mechanism is required to find the optimal number and size of the pseudo-populations. Also the interval for the reorganizing generation is currently based on statistical information about the benchmark test functions. In future research we aim to base this on the characteristics of the evolving phenotypic solution space keeping the computational overhead within acceptable limits.

References

[1] Beyer, H. Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. Computer Methods in Applied Mechanics and Engineering, 186(2-4) , 2000, pp. 239 - 267.

[2] Bhattacharya, M. Exploiting Landscape Information to Avoid Premature Convergence in Evolutionary Search., Proceedings of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, Canada, 0-7803-9487-9/06, IEEE Press, 2006 , pp. 2575-2579.

[3] Branke, J. , Schmidt, C. and Schmeck, H. Efficient fitness estimation in noisy environments. Proceedings of the Genetic and Evolutionary Computation Conference 2001, GECCO2001, Morgan Kaufman, 2001, pp. 243 – 250.

[4] Darwen, P. J. Computationally intensive and noisy tasks: co-evolutionary learning and temporal difference learning on Backgammon. Proceedings of the 2000 Congress On Evolutionary Computation, CEC2000, Volume 1, , 2000, pp. 872 – 879.

[5] Jin, Y. and Branke, J. Evolutionary optimization in uncertain environments - A survey. IEEE Transactions on Evolutionary Computation, 9(3), , 2005, pp. 303 – 317.

[6] Krink, T., Filipic, B. and Fogel, G. B. Noisy optimization problems - a particular challenge for differential evolution?. Proceedings of IEEE Congress on Evolutionary Computation, CEC2004, Volume 1, 2004, pp.332 – 339.

[7] Rudolph, G. A partial order approach to noisy fitness functions. Proceedings of the Congress on Evolutionary Computation, CEC2001, IEEE Press, USA, 2001, pp. 318 – 325.

[8] Sano, Y. and Kita, H. Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation. Proceedings of the

2002 Congress on Evolutionary Computation, CEC2002, IEEE Press, USA, pp. 360 – 365.

[9] Tamaki, H. and Arai, T. A. A genetic algorithm approach to optimization problems in an uncertain environment. Proceedings of International Conference on Neural Information Processing and Intelligent Information Systems, Volume 1, 1997, pp. 436-439.