On the Application of Cohort-Driven Operators to Continuous Optimization Problems using Evolutionary Computation

Arnold L. Patton pattona@cps.msu.edu
Terrence Dexter dexterte@cps.msu.edu
Erik D. Goodman goodman@egr.msu.edu
William F. Punch III Punch@cps.msu.edu

Genetic Algorithms Research and Applications Group (GARAGe)
3115 Engineering Building
Michigan State University
East Lansing, MI 48825
http://GARAGe.cps.msu.edu/

Abstract. Traditional approaches to real-valued function optimization using evolutionary computational methods tend to use either self-adaptive operators (as in the case of evolutionary programming), or population-based operators (as in the case of most real-valued genetic algorithms). However, in general, most population-based operators are limited in scope to the use of at most two or three parent individuals. In this paper we explore an alternative population-based form of adaptation for evolutionary computation, Guided Gaussian Mutation (GGM), which is designed specifically as a localized search operator. This operator is the first of a larger class of Cohort Driven Operators (CDOs) which we define here. Experimental results using GGM in a standard genetic algorithm framework on a series of test problems show impressive improvement over standard evolutionary programming.

Introduction

A number of concepts in evolutionary computation (EC) have arisen from specific philosophic or metaphoric inspirations. Perhaps this is only natural given that the unifying thesis of evolutionary computation is itself based on the metaphor of simulating biological evolution. Many of the techniques employed by various EC systems are direct extensions of the underlying metaphor or philosophy. For example, the GA crossover operator extends naturally from the idea of manipulating solution genomes. Likewise, the mutation operator in EP extends directly from its philosophical base, which focuses attention on phenotypic rather than genotypic effects. Some innovative search techniques have been created through such insights ([Fogel 66], [Holland 75],[Koza 92], to name a few); however, while such metaphors may be useful in developing new techniques, they can also become limiting if they are used to screen out techniques which do not fit within the framework of the underlying metaphor.

In light of the “No Free Lunch” theorems [Wolpert 95][English 97], such conceptual litmus tests make little sense. The generally applicable, most efficient search technique has become a myth. Since we cannot expect any single approach to become uniquely the best in a general sense, perhaps rather than focusing on broadening any single metaphor we should consider blending concepts from the existing philosophies, selecting features of different systems or metaphors as appropriate to the domain and blending them together. With the blending of concepts, the focus becomes shifted from evaluation of the metaphor to evaluation of the individual derived concepts. By incorporating good concepts from separate approaches, we may be able to achieve significant advancement over existing techniques for certain classes of problems.

This paper presents an operator, Guided Gaussian Mutation(GGM), which attempts to blend the phenotypic character of EP mutation with the population-adaptive mechanism typified by a number of GA and ES crossover operators.

Discussion

Generally, the debate over the relative merits of EP and GA involves the evaluation of the operators of the two approaches, most specifically focusing on the reproductive operators. In essence, this becomes a philosophical debate between the inherent value of phenotypic self-adaptive operators, as typified through the EP mutation operator, and genotypic population-adaptive operators, such as crossover. Here, we broaden the topic of population-adaptive operators to consider a larger class of operators, that of Cohort Driver Operators (CDO). Also, a new CDO, the Guided Gaussian Mutation operator is proposed.

Cohort Driven Operators

The GA crossover operator, in its various forms, is in essence, a population-adaptive operator. The net modification produced by an application of crossover is directly dependent on the distribution of the individuals in the population across the problem space. As the population converges, the ability to produce diverse individuals through crossover is greatly reduced.

In general, any operator that relies on information from other individuals in the population in order to transform individual i into a new individual i’ may be classified as a Cohort Driven Operator (CDO). The term cohort here specifies that the selection of the additional individuals from the population may be biased, for example, in an attempt to provide some form of niching. Therefore, the classification of CDO is more encompassing than population-adaptive. Examples of existing CDOs include all forms of GA crossover, including one-point, two-point and uniform crossover, BLX-α [Eshelman 93], and other more recent variants such as uniform normal distribution crossover (UNDX) [Ono 97].

An inherent drawback with the use of CDOs is that since the CDO draws its “driving force” from the population as a whole, any operation that directly affects the composition of the population over time also affects the operation of the CDO. Thus, not only is a CDO somewhat recursive in nature, but the actions of a CDO are directly tied to all other operators, such as selection and mutation. For this reason, the operation of CDOs is often very difficult to analyze, as is the case with standard GA crossover. Clearly a CDO may open a Pandora's box of intertwined relationships and should only be pursued if the potential gain outweighs the potential for instability.

By the same token, CDOs are limited in power by the diversity (or lack thereof) of the population. As the population converges, there is no additional information to be found in a cohort sampling, since all members of the cohort are identical to the solution being modified. Therefore, a requirement when using most CDOs is that some form of "diversity adding" mutation be present or that large populations be used to avert premature convergence.

Guided Gaussian Mutation

EP proponents adhere to the proposition that a mutation operator should have a higher probability of making little or no modification to an individual than making large scale changes. Typically, this mutation is accomplished by addition of samples from a zero-mean probability distribution such as a normal distribution. The action of this operator in a continuous domain is more intuitively tractable than that of crossover or bit mutation. However, in order to apply this technique, we need to choose an appropriate set of mutation magnitudes relative to a given solution. Early EP approaches used the error of the solution to estimate the mutation magnitudes. Modern approaches use self-adaptive encoding of magnitudes directly for each individual [Bäck 93], [Savaranan 95]. However, the former implies some known error limit and a relationship between the magnitudes of parameter values and error, while the latter has the apparent effect of increasing the size of the search space, searching for both the best solution and for the best parameters to continue improving that solution, which would imply a slower or more difficult search process.

Guided Gaussian Mutation (GGM) is a CDO based directly on the EP philosophy focusing on phenotypic rather than genotypic effects. Like EP mutation, it incorporates a zero-centered mutation biased toward smaller mutations; however, instead of encoding the scale of the mutation as a self-adaptive parameter, it is measured directly from the population. This operator blends the basic design and philosophy of the EP mutation operator with the population-adaptive concept found in GA.

For the basic GGM algorithm, a cohort sample is selected from the population for each individual to be mutated. The standard deviation across each parameter is calculated for the sample; these values are then used directly as the standard deviations for the mutative sampling. In the current implementation, the algorithm simply selects individuals from the population without bias (i.e. the cohort is an unbiased sample of the population). Typical sample sizes range from 5 to 20; however, behavioral changes under different sample sizes within this range appear minor. For all experimental data collected here, the sample size was fixed at 10.

The GGM operator explores near the current solution roughly within the same bounds as the population as a whole. Thus GGM mutates convergent parameters at a lower magnitude, while at the same time the diversity of more fluid parameters is maintained by larger magnitude mutation. Of course, this points to one of the drawbacks of GGM: like most CDOs, it becomes useless when the population completely converges, and therefore it requires the use of an additional mutation operator.

Unfortunately, although the motivation behind the creation of GGM was that of hybridization, a reasonable fit between GGM and the EP framework has yet to be realized. Preliminary tests, using GGM in conjunction with and in place of standard lognormal EP mutation in an EP framework show a slight degradation over standard lognormal self-adaptive EP mutation. However, the performance disparity of GGM within GA and EP frameworks is not entirely surprising since GGM is a CDO, and as has been previously discussed, the behavior of all CDOs is directly affected by all other cohort-affecting operators. Therefore, the failure to find a match between GGM and EP may be either due to an incompatibility between some operator in the EP framework and GGM, or due to a dependency between GGM and some element of the GA framework.

Experimental Design

In testing the potential effectiveness of the GGM operator, we selected a number of well regarded optimization test functions from the literature (e.g. [Yang 97], [Savaranan 95], [Salomon 96]) as well as an original function (Sphere-Hull) intended to be difficult for algorithms which tend toward the population mean or have difficulty following non-linear surfaces. All functions were redefined to allow scaling to any number of dimensions. The details of the functions used for this evaluation are outlined in Table 2.

Function	Name	Translated Range
$\sum x_i^{\,2}$	Sphere	$-100 \le x_i \le 100$
$\left\|\sqrt{\sum (x_i - c_i)^2} - 100n\right\|^2 + \sqrt{\sum (x_i - t_i)}$ where $\sqrt{\sum (t_i - c_i)^2} = 100n$	Sphere-Hull	$-500 \le x_i \le 500$
$\sum - x_i \sin(\sqrt{\|x_i\|}$	Schwefel 1	$-500 \le x_i \le 500$
$-20e^{(-0.2*\sqrt{\frac{1}{n}\sum x_i^{\,2}})} - e^{\frac{1}{n}\sum \cos(2\pi x_i)} + 20 + e$	Ackley	$-30 \le x_i \le 30$
$\sum^{n-1} (x_i^{\,2} - 2x_{i+1}^{\,2} - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7)$	Bohachevsky	$-15 \le x_i \le 15$
$10n + \sum (x_i^{\,2} - 10\cos(2\pi x_i))$	Rastrigrin	$-15 \le x_i \le 15$
$\sum^{n-1} (x_i^{\,2} + x_{i+1}^{\,2})^{0.25}[\sin^2(50*(x_i^{\,2} + x_{i+1}^{\,2})^{0.1}) + 1.0]$	Schaffer	$-100 \le x_i \le 100$
$\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j^{\,2}\right)$	Schwefel 2	$-500 \le x_i \le 500$
$1 + \sum \frac{x_i^{\,2}}{4000} - \prod\left[\cos\left(\frac{x_i}{\sqrt{i}}\right)\right]$	Griewangk	$-600 \le x_i \le 600$
$\sum_{i=1}^{n-1}[100*(x_i^{\,2} - x_{i+1}) + (1 - x_i)^2]$	Rosenbrock	$-15 \le x_i \le 15$

Table 1. Test functions. Given ranges are after translation, but before rotation.

These test functions were selected because of their ease of computation and widespread use, which should facilitate evaluation of the results and comparison to similar work. No claim is being made as to the representative qualities of these functions for any larger classes of functions. Several of these functions are highly multimodal; and many are known to be difficult for other search algorithms, especially under high levels of dimensionality (i.e. in a black box form where the search algorithm should not necessarily assume independence of dimensions).

Translation and Rotation

In order to avoid any bias toward a given operator introduced by axial symmetry (such as we might expect to be the case with crossover), all test problems were translated away from the origin by a fixed amount in all dimensions (e.g. a solution of (0,0) is moved to (10,10), etc.) Additionally, in light of recent results in [Salomon 97], all test functions were tested in both their original form and with an intervening 45 degree rotation for each contiguous pair of dimensions. This served to test both against any inherent alignment bias between the function landscape and the operators, and for the effects of pleiotropy on the algorithms tested. Rotation was applied prior to translation. Each tested algorithm was examined using both original and rotated forms of all test functions.

The G-nMRM Algorithm

In an attempt to isolate whether any observed gain while using the GGM operator originates from information gleaned from the cohort sampling, (as opposed to, say, simply the fact that GGM is a zero-mean Gaussian mutation operator), an alternate mutation algorithm was constructed. The Gaussian n-Scaled Multi-range Mutation (G-nMRM) operator allows the scale of a Gaussian mutation to be selected as follows: select uniformly from the range $s = \mathbf{[}0...(r_{max}-r_{min})^{n}\mathbf{)}$, then take the magnitude of the mutation to be $10^{r_{\max}-\sqrt[n]{s}}$. Once the magnitude is selected for an individual, each parameter is mutated by addition of a sample from a zero-mean normal distribution with a standard deviation of the selected magnitude multiplied by the initial range for the given parameter.

Algorithms Tested

For a given GA application, any combination of operator application rates may be chosen. After extensive initial testing, several operator mixtures were chosen for further evaluation. These combinations are listed in Table 3 below. These combinations were chosen explicitly to test various aspects of the interplay between these operators within the GA framework. GA1 represents a “standard” GA approach dominated by application of crossover.

In addition to these five GA variants, a basic EP algorithm using log-normal self-adaptive parameter updates was tested to provide methodology verification and as an external baseline for comparison. For the purpose of comparison, all algorithms tested (both EP and GA variants) encode the problem parameters as 64-bit IEEE floating point reals (doubles). Also, the same random number generators were used in all tested GA and EP implementations (although not the same random seeds), and the identical code for function evaluation was used in each. The GA used was a modified version of GALOPPS [Goodman 96], while Savaranan provided the EP code, which is the same code used in [Savaranan 95]. Both implementations are originally based on Goldberg's implementation of SGA [Goldberg 89].

Results

Results were obtained for all six test algorithms on 10-dimensional versions of the translated test functions, both with and without coordinate rotation, averaged over 100 test runs for each algorithm/function pair. Algorithms were tested with population sizes of 200 and 500. All test runs were tracked for 500 generations, except for GA2 and GA5, which were stopped after 1300 and 2500 generations respectively, since their lower application rate generate fewer evaluations per generation. Tables 4 and 5 list results for selected algorithms on the rotated functions using population sizes of 200 and 500 respectively.

Interpretation

The GA3, GA4, and GA5 variants show an impressive improvement over all other evaluated algorithms for a number of test functions. As expected EP outperformed the "standard" GA1 in nearly all cases. Also, the observed EP performance parallels that reported in [Savaranan 95].

A common observation, which held across almost all tests, is that both GA1 and GA2 variants performed significantly poorer than the GA3 variant. This is intriguing, especially since it implies that, while neither crossover nor GGM is capable of any remarkable performance on its own, the combination of crossover and GGM leads to a larger performance gain than the sum of the gains obtained from using each operator separately. Given that both crossover and GGM are CDOs, and that GGM was designed to an extent to fill the need for local exploration in a GA this is acceptable, even somewhat expected behavior. Still, the emergent properties of the combination are quite striking and pronounced.

Sphere, Sphere-Hull, and Schwefel 1

Except for minor differences, the relative performance of all tested algorithms was consistent across the Sphere, Sphere Hull, and Schwefel 1 functions (both rotated and non-rotated). Figure 1 illustrates the online performance of all tested algorithms using a population size of 500 on the sphere function. Performance across all algorithms varied uniformly under rotation for all three test problems, with only minor performance loss. The low-level mutation and low crossover combination of GA5 worked very well on these broad, smooth, symmetric functions.

GA Parameters

Tournament Selection, w/ Tournament Size of 2
Single Best Elitism
Field-Based 2-Point Crossover
GGM Cohort Sample size of 10
Scale Factor of 2 for G-*n*MRM, with $r_{min} = 0$, and $r_{max} = 20$

EP Parameters

Log-Normal Self-Adaptive Parameter Updates
Self-Adaptive Parameter Initialization (initialrange)/(6*sqrt(n))
1 Child per Parent per Generation
EP Tournament Ranking Selection w/ Voting Pool of 10

Table 2. EP and GA Parameter settings

Algorithm Name	*Crossover Rate*	*G-2MRM Rate*	*GGM Rate*
GA1	90%	15%	0%
GA2	0%	0%	15%
GA3	90%	0%	15%
GA4	90%	50%	10%
GA5	5%	5%	10%

Table 3. GA variants tested.

Function	EP			GA1			GA3			GA5		
	Evals.	Log Mean Best	Log S.Dev.	Evals.	Log Mean Best	Log S.Dev.	Evals.	Log Mean Best	Log S.Dev.	Evals.	Log Mean Best	Log S.Dev.
Sphere	99837	-3.00	-2.19	84621	-0.31	0.07	84588	-7.28	-6.49	93160	-29.53	-28.90
Sphere-Hull	99354	0.35	0.88	84577	1.57	1.35	84561	-1.41	-0.76	93195	-14.05	-14.47
Schwaef.1	99860	0.99	1.88	84608	2.76	2.95	84644	-1.74	-0.87	93475	-15.08	-14.37
Ackerman	99069	0.17	0.15	84595	0.71	0.14	84349	-1.73	-0.78	93058	0.37	0.16
Bohachev.	99623	0.36	0.14	84562	0.95	0.55	83240	-0.56	-0.32	93216	0.51	0.19
Rastrigin	99295	1.42	0.97	83956	1.31	0.93	83105	0.94	0.65	93007	1.49	1.25
Schaffer	94992	1.22	0.89	84254	1.40	0.63	84130	0.42	0.51	93430	1.50	1.12
Schwaef.2	97080	3.09	2.49	84631	3.12	2.41	83712	3.05	2.48	92920	3.20	2.53
Griewank	99796	-0.69	-0.62	84605	0.15	0.00	84533	-2.09	-2.01	93075	-0.52	-0.55
Rosenbrock	99871	1.35	1.78	84621	2.07	2.69	84133	0.96	1.15	93583	1.35	1.78

Table 4. Comparison of selected algorithms with population size of 200 on rotated functions.

Function	EP			GA1			GA3			GA5		
	Evals.	Log Mean Best	Log S.Dev.	Evals.	Log Mean Best	Log S.Dev.	Evals.	Log Mean Best	Log S.Dev.	Evals.	Log Mean Best	Log S.Dev.
Sphere	249503	-4.22	-3.23	211584	-3.53	-3.08	211380	-INF	-INF	240816	-30.15	-29.54
Sphere-Hull	249516	-0.71	-0.14	211558	0.25	0.38	211446	-14.21	-29.40	241118	-14.15	-15.07
Schwaef.1	249294	-0.82	0.16	211496	1.40	1.81	211444	-27.78	-27.00	242338	-24.96	-24.78
Ackerman	249444	-0.11	-0.05	211616	0.43	-0.07	211415	-INF	-INF	241341	-0.52	-0.22
Bohachev.	247441	0.19	0.04	211578	0.65	0.23	202778	-1.19	-0.79	241106	-0.21	-0.16
Rastrigin	244889	1.36	0.99	211563	1.02	0.65	206755	0.69	0.45	241321	1.40	1.07
Schaffer	237537	1.18	0.81	204734	1.24	0.52	210014	-0.22	-0.04	242055	1.31	1.09
Schwaef.2	249297	3.03	2.41	207937	3.03	2.48	203954	2.91	2.48	241032	3.10	2.61
Griewank	249478	-0.76	-0.98	211611	-0.44	-0.60	211415	-2.64	-2.31	241693	-0.82	-1.06
Rosenbrock	249702	1.46	2.08	211492	1.02	1.01	211637	0.84	-0.14	242571	0.91	1.23

Table 5. Comparison of selected algorithms with population size of 500 on rotated functions.

Figure 1. : Performance on sphere function (10 Dimensions)

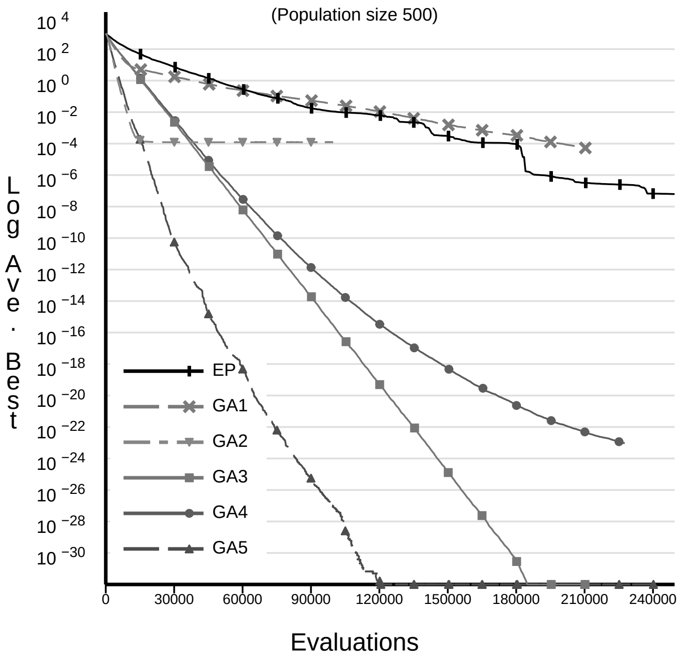


Figure 2. : Performance on rotated Ackley function (10 Dimensions)

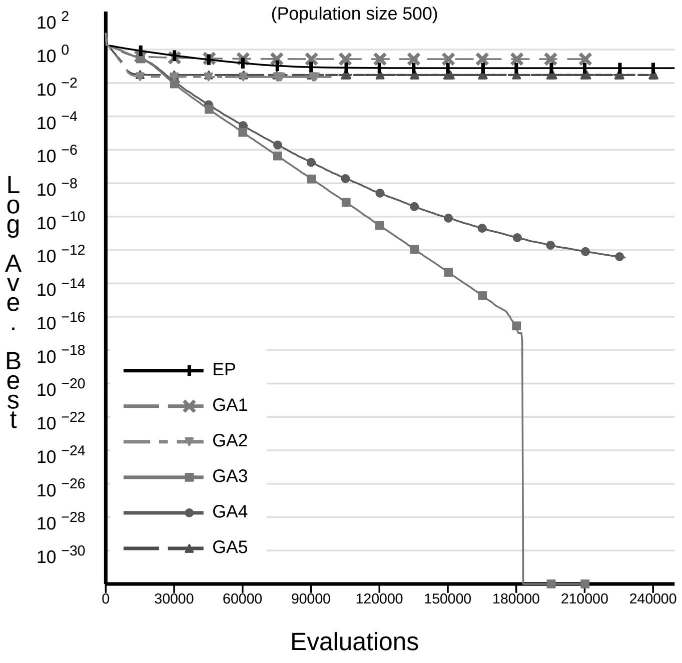
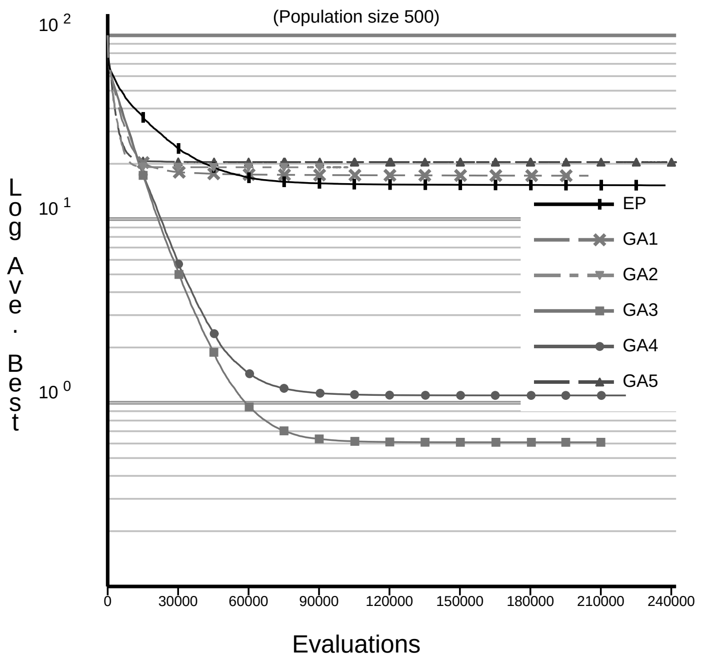


Figure 3. : Performance on rotated Schaffer function (10 Dim.)



Population Size Effects

As expected, variants GA2 and GA3 show the largest gain as the population size is increased (since all of the operators in these variants are useless under premature convergence); however, the difference in behavior for GA3 between population sizes is the most striking. This behavior holds constant for the Sphere, Sphere-Hull, and Schwefel 1, as well as Ackley and Bohachevsky functions. In all these cases except the Bohachevsky function, this behavior was also invariant under rotation (i.e. rotation had no effect except for a minor modification of the slope and convergence points). In-depth analysis shows that, in fact, approximately 95% of the runs using GA3 with the smaller population size show the same sort of direct exponential error reduction as is observed using a larger population size; however, the remaining 5% converge prematurely fairly early in the run.

Ackley and non-rotated Bohachevsky

On these functions, GA3 and GA4 outperform all other tested algorithms, especially as the population size is increased. As expected, GA5, which performed so well on earlier functions, becomes much less effective. Also, the dramatic population size effects for GA3 and GA4 are clearly visible. Figure 2 illustrates the online performance of all tested algorithms using a population size of 500 on the rotated Ackley function.

Schaffer, Rastrigin, and Griewangk

In general for these functions, algorithms GA3 and GA4 show improvement over the remaining algorithms (though not as strikingly as before), and the other algorithms are relatively indistinguishable. This separation is more distinct under rotation. Also, the population size effects for GA3 and GA4 are not as dramatic. Figure 3 illustrates online performance of all tested algorithms using a population size of 500 on the rotated Shaffer function.

Conclusion

GGM is a successful blending of both GA and EP philosophies. It is apparent that for several functions, a system using GGM and parameter-based crossover can achieve significantly better performance than both standard EP and GA approaches. In all test cases, GA3 performed at least as well as EP, and often performed significantly better. The apparent success of GGM appears to be related to an interplay between GGM and parameter-based crossover; hence we may expect some difficulties in achieving similar success in using GGM within a standard EP framework.

Acknowledgements

The authors would like to thank N. Savaranan for providing the EP code and aiding in reproducing EP results. Also, thanks to Kumar Chellapilla and David Fogel for answering numerous, tangled EP questions and for offering feedback and encouragement in equal portions.

Bibliography

Bäck, T. and H. P. Schwefel (1993), "A Survey of Evolutionary Algorithms for Parameter Optimization," Evolutionary Computation, 1 (1), pp.; 1-23.

English, T. M. (1997), "Evaluation of Evolutionary and Genetic Optimizers: No Free Lunch", Proceedings of the Fifth Annual Conference on Evolutionary Programming, MIT Press, pp. Pp. 163-169.

Eshelman, L. (1991), "The CHC Adaptive Search Algorithm," Foundations of Genetic Algorithms and Classifier Systems 1, Morgan-Kaufmann, pp. 265-283.

Eshelman, L. (1993), "Real-Coded Genetic Algorithms and Interval-Schemata," Foundations of Genetic Algorithms 2, Morgan-Kaufmann, pp. 187-202.

Fogel, D.B. and J. W. Atmar (1990), "Comparing Genetic Operators with Gaussian Mutations in Sumulated Evolutionary Processes Using Linear Systems," Biological Cybernetics, 63, pp. 111-114.

Fogel, L.J., A.J. Owens, and M.J. Walsh. (1966), Artificial Intelligence through Simulated Evolution. Wiley, New York.

Goldberg, D. E. (1989), Genetic Algorithms in Search, Optimization and machine Learning, Addison Wesley Publishing Company.

Goodman, E. D. (1996), "An Introduction to GALOPPS – the Genetic ALgorithm Optimized for Portability and Parallelism System, Release 3.2", Technical Report 96-07-01, Genetic Algorithms Research and Applications Group (GARAGe) and Case Center for Computer-Aided Engineering and Manufacturing, Michigan State University.

Gordon, V. S. and D. Whitley (1993), "Serial and parallel Algorithms as Function Optimizers," Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan-Kaufmann., pp. 177-183.

Holland, J. H. (1975), Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI.

Janikow and Z. Michalewicz, (1991), "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms," Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan-Kaufmann., pp. 31-36.

Koza, J. R. (1992) Genetic Programming, The MIT Press, Cambridge, MA.

McClave, J. T. and F. H. Dietrich, II (1985), Statistics, Third Ed., Dellen Publishing Company.

Michalewicz, Z. (1992), Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin.

Ono, I. And S. Kobayashi (1997), "A Real-Coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover," Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan-Kaufman, pp. 246-253.

Salomon, R. (1996), "Performance Degradation of Genetic Algorithms under Coordinate Rotation," Proceedings of the Fifth Annual Conference on Evolutionary Programming, MIT Press., pp. 153-161.

Saravanan, N., D.B. Fogel and K.M. Nelson (1995), "A Comparison of Methods for Self-Adaptation in Evolutionary Algorithms," BioSystems, 36, pp 157-166.

Wright, A.H., (1991), "Genetic Algorithms for Real Parameter Optimization Foundations of Genetic Algorithms and Classifier Systems, Morgan-Kaufmann., pp. 205-218.

Wolpert, D. and W. Macready (1995), "No Free Lunch Theorems for Search," Santa Fe Institute, SFI-TR-02-010.

Yang, J., J. Horng, and C. Kao (1997), "A Continuous Genetic Algorithm for Global Optimization," Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan-Kaufmann., pp. 230-237.