An Evolutionary Programming Algorithm for Automatic Engineering Design

An Evolutionary Programming Algorithm for Automatic Engineering Design

Andrew Lewis[1], David Abramson[2] and Tom Peachey[2]

[1]Division of Information Services, Griffith University, Brisbane, Queensland, Australia
[2]Dept. of Computer Science and Software Engineering, Monash University, Melbourne, Victoria, Australia
a.lewis@griffith.edu.au, davida@csse.monash.edu.au, tcp@csse.monash.edu.au

Abstract: This paper describes a new Evolutionary Programming algorithm based on Self-Organised Criticality. When tested on a range of problems drawn from real-world applications in science and engineering, it performed better than a variety of gradient descent, direct search and genetic algorithms. It proved capable of delivering high quality results faster, and is simple, robust and highly parallel.

Introduction

In the engineering design process there is a demand for the capability to perform automatic optimisation, minimising or maximising some derived quantity, a measure of "fitness" of the design. For real-world problems these objective function values are often computed using sophisticated and realistic numerical simulations of physical phenomena. This is an extremely computationally intensive process when models must be run tens, or maybe hundreds, of times to effectively search design parameter space. Current High Performance Computing systems mostly derive their capacity from parallel architectures so for optimisation methods to be practical and effective the algorithms used must preferably have a large degree of concurrency. Figure 1 offers a simple taxonomy of methods that can be used for this automatic optimisation.

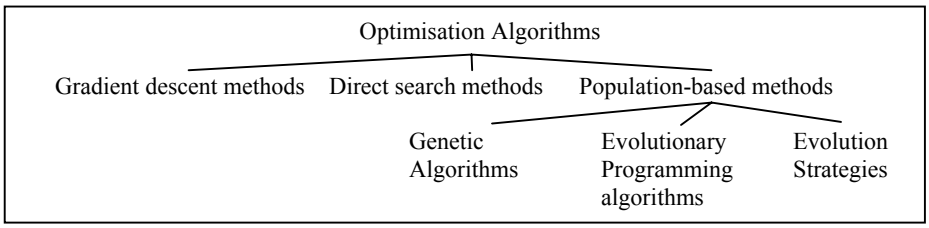

Fig. 1. a taxonomy of optimisation algorithms

Gradient descent algorithms, based on classical, Newtonian methods of analysis, generally exhibit rapid local convergence [1] but due to the iterative steps of gradient determination and some form of line search most employ, they are inherently sequential in nature. In many problems arising in real-world design processes there are limits

imposed on problem parameters by their nature; it does not make sense, for example, for a physical component to have negative size. The authors have previously exploited this simple bounding of engineering problems to enhance the parallelism of line searching to some extent [2].

Of greater potential concurrency are **direct search methods**, such as the Nelder-Mead Simplex algorithm [3]. These methods do not explicitly use approximate gradient information, but instead take objective function values from a set of sample points, and use that information to continue sampling. Use of the Nelder-Mead simplex algorithm remains current, largely because on a range of practical engineering problems it is capable of returning a very good result [4].

The early direct search methods, however, exhibit very limited concurrency. The Multidirectional Search (MDS) method of Dennis & Torczon [5] has greatly enhanced parallelism. The authors have explored hybrids and variants of Nelder-Mead Simplex and MDS in order to try and increase the amount of concurrency.

Offering the greatest concurrency are **population-based methods**, in which large numbers of objective function evaluations are typically performed in parallel, evolving solutions over several "generations". It is only over the past decade that the computational capacity available to scientists and engineers has increased to the point where population-based methods of optimisation have become practical for the solution of real-world problems. Moreover, in engineering design, the evaluation of the objective function is so much slower than the rest of the algorithm, that such codes demonstrate excellent speedup in spite of the need for global communication on each iteration. **Genetic Algorithms** (GA) now may be frequently encountered in application to engineering problems (for examples, see Alander [6]). These have inherent, easily exploitable parallelism, and attractive global convergence probabilities, but have been considered generally slow to converge [7].

In the general case, evolutionary computation seeks to use insight into natural processes to inform population-based computational methods. **Evolutionary Programming** refers to that class of methods in evolutionary computation that apply a random mutation to each member of a population, generating a single offspring. However, unlike Genetic Algorithms, no recombination operators are applied. Selection takes place, and half the combined population of parents and offspring enter the next generation. Population members are considered as representative of species, rather than individuals. Such methods are generally simple, robust and highly parallel.

The theory of self-organised criticality gives an insight into emergent complexity in nature [8]. Bak contended that the critical state was "the most efficient state *that can actually be reached dynamically"*. In this state, a population in an apparent equilibrium evolves episodically in spurts, a phenomenon known as *punctuated equilibrium.* Local change may affect any other element in the system, and this delicate balance arises without any external, organising force. We have utilised these concepts in developing a new Evolutionary Programming algorithm that, in addition to the conventional mutation and selection operations, implements a further selection operator to encourage the development of a self-organised critical system. The algorithm is evaluated on a range of test cases drawn from real-world problems, and compared against a selection of algorithms, including gradient descent, direct search methods and a genetic algorithm.

EPSOC: an Evolutionary Programming algorithm using Self-Organised Criticality

The general optimisation problem can be stated as:

$$\text{Minimize } f(\mathbf{x}) \text{ where: } f \text{ is an arbitrary non-linear function and } \mathbf{x} = \{x_0,\ldots,x_i,\ldots,x_n\} \quad (1)$$

For a population-based method, the population, p, consists of a set of parameter vectors, $\mathbf{x}$: $p = \{\mathbf{x}_0,\ldots,\mathbf{x}_i,\ldots,\mathbf{x}_m\}$. For real-world engineering design problems, values for $f(\mathbf{x})$ generally can only be derived from execution of complex numerical simulations requiring considerable computation time.

As stated in the introduction, EPSOC is built on the basis of an Evolutionary Programming algorithm, with an additional selection operator following the method of Bak. The steps of the algorithm are outlined below:

1. Initialise a random, uniformly-distributed population, p, and evaluate each trial solution, $\mathbf{x}_i \; \forall \; i$.
2. Sort the population by objective function value, $f(\mathbf{x})$.
3. Select a set, B, of the *nbad* worst members of the population. For each member of B, add to the set its two nearest neighbours in parameter space that are not already members of the set, *or from the best half of the sorted population*.
4. Apply a random, uniformly-distributed mutation to the selected set, B, i.e. re-initialise them. For all other members of the population, generate a "child" by applying a small (~10% of parameter range), random, uniformly-distributed mutation to the "parent" member.
5. Evaluate each new trial solution, $f(\mathbf{x})$.
6. If a child has a better objective function value than its parent, replace the parent with the child.
7. Repeat from step 2 until a preset number of iterations have been completed.

As each set of parameters defining a trial solution is independent of all others, it is immediately apparent that the evaluation of trial solutions at steps 1 and 5 can be performed concurrently. Since the evaluation of the objective function completely dominates the execution time, from Amdahl's Law we can expect extremely high parallel efficiency.

Critical states of evolutionary models exhibit a power-law distribution of the size and frequency of extinction events. This is apparent when the number of events involving given numbers of members of the population approximate a straight line in a double-log statistical plot of the phenomena. In earlier applications of self-organised criticality to optimisation, it has been proposed that a separately computed power-law extinction rate be imposed on a spatial diffusion model, or cellular GA [9]**.** In Krink and Thomsen's model, population members are chosen at random for extinction, and the algorithm is greedy to the extent of maintaining the single best member, an elite of one. Their algorithm apparently does not attempt to evolve a population in a critical state, but indirectly imposes the observed behaviour of such a population.

In contrast, EPSOC is a straightforward implementation of Bak's nearest-neighbour, punctuated equilibrium model as an optimisation algorithm. By considering the trial

solution parameter vectors as defining a location in an *n*-dimensional parameter space, the spatial behaviour of Bak's model is realized naturally. We also chose to apply a high degree of greediness to the algorithm. Maintaining a large "elite" (in EPSOC, half the total population) can be viewed as a "constructive" operator. Like "Maxwell's demon", it accretes information and encourages a gradual improvement in the better half of the population. This achieved as re-initialised or mutated population members move into the "protected" half of the population, and the median of the population moves toward better objective function values.

The Case Studies

We have assembled a number of case studies drawn from interesting and challenging scientific and engineering applications. These were used to test and assess the performance of the individual algorithms. Generally, the case studies fall into 2 sets:

- Smooth, with a dominant global minimum (Laser 1, Crack 1, Aerofoil - Figures 2(a), 2(c) & 2(e), and Rosenbrock's function)
- Multiple local minima, non-convex (Laser 2, Crack 2, Bead - Figures 2(b), 2(d) & 2(f))

Laser 1 and 2:

A two-dimensional test surface was derived from the computation of a quantum electrodynamical simulation of a laser-atom interaction experiment [10]. The base case, Laser 1, is quite a smooth surface, the dataset containing only 4 minima, of which the global minimum is quite dominant, as can be seen in Figure 2(a). Additive fractal noise was overlaid on this dataset to develop a "noisier", more challenging surface to test the algorithms. This dataset, Laser 2, contained 1157 local minima of varying severity, and is illustrated in Figure 2(b).

Crack 1 and 2:

Finite element analysis of a thin plate under cyclic loading, with a cutout specified by parameters, was used to generate the Crack datasets [11]. Common practice in damage tolerant design has been to minimise the maximum stress under load. Isosurfaces of these stress values are shown in Figure 2(c). This dataset, Crack 1, was reasonably smooth, with only 26 local minima. A new approach in modeling stressed components is to attempt to maximize durability. The finite element model of the crack-stress test case included fatigue cracks at a number of locations, and the objective of this test case, Crack 2, was to maximise the life of the part as determined by the time taken for fatigue crack growth to a defined length. Isosurfaces at a number of values are shown in Figure 2(d). In contrast to Crack 1, this dataset was "noisy", with 540 local maxima, and discontinuous isosurfaces.

Aerofoil:

This test case models the aerodynamic properties of a two dimensional aerofoil. The objective function to be minimised is the lift-drag ratio [12], and this is computed by executing a Computational Fluid Dynamic model of the object. Figure 2(e) shows a

number of isosurfaces in the parameter space investigated. The dataset was generally smooth, with only 12 local minima and a dominant global minimum.

Bead:

The application from which this case study was drawn used a ceramic bead to minimise distortion of the radiation pattern of a mobile telecommunications handset during testing [13]. The objective function value, derived from an FDTD full-wave analysis of the cable structure, was a measure of transmission strength through the bead at 1 GHz. The dataset for the Bead case study, of which isosurfaces for a particular value are shown in Figure 2(f), is quite complex and rich in structure. It contains 298 local minima.

Rosenbrock's function:

In order to provide a point of comparison, the well-known Rosenbrock's function in two dimensions was included. The objective function values for this test case were directly computed from $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ for $x_i \in [-2,2]$ which has one local minimum at $f(1,1) = 0$.

Results of Numerical Experiments

For purposes of comparison a set of algorithms were tested, including EPSOC, a GA (Genesis 5.0 [14]), a parallel gradient descent method (P-BFGS), Dennis and Torczon's MDS, the Simplex method of Nelder and Mead, a variant of Simplex operating concurrently on multiple vertices with a search direction defined by a reducing set of vertices (RSCS), and variants of Simplex and RSCS utilizing line-searching.

The population-based algorithms, EPSOC and Genesis, used a population of 64 members. Other operational parameters of the algorithms, such as mutation and crossover rates for the GA, and extinction rate and mutation range for EPSOC, were tuned for each test case and the best results obtained are reported in this paper. Both algorithms are ideally suited to a master-slave, gather-scatter parallel system.

Each of the algorithms tested was run on each of the test cases from 10 randomly distributed start points. To allow direct comparison, in a given test case the same set of start points were used for each algorithm.

The starting simplices for the direct search methods were right simplices aligned with the coordinate axes. By default they were reasonably large, scaled to 10% of the parameter range for each coordinate. The convergence criterion for most cases was a fractional step-wise gradient of 10^{-3}, except for EPSOC and Genesis, which used fixed limits on iteration count and function evaluations, respectively.

Table 1 shows the best objective function value obtained in 10 runs for each algorithm on each test case, and the Equivalent Serial Function Evaluations (ESFE) required to obtain that result. ESFE, derived from the number of steps involving concurrent evaluation of objective functions, is a measure of the actual time taken to find an optimal solution. For each test case, the best objective function value obtained by any algorithm, and the fastest ESFE to obtain that value, are highlighted.

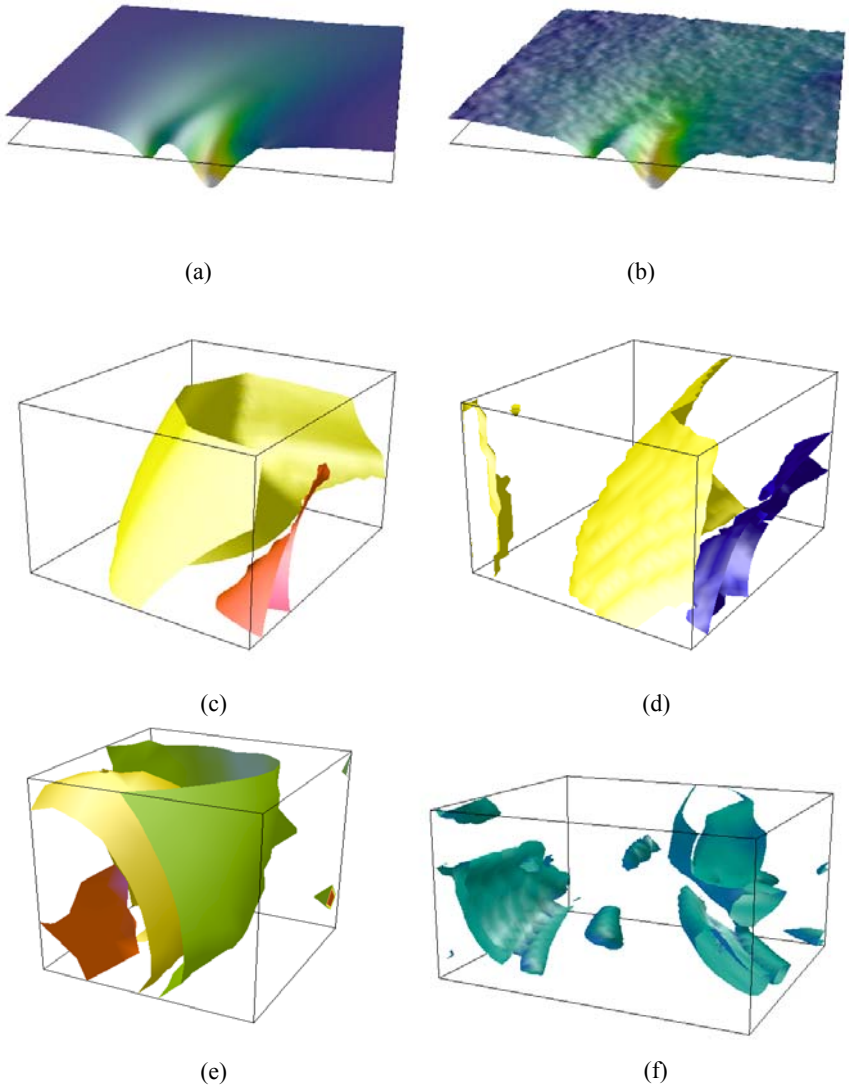

Fig. 2. test case isosurfaces

The ESFE results shown for EPSOC and the GA are the generation counts at which the minimum result was obtained, not the count for termination, which was fixed.

It may be noted from Table 1 that EPSOC effectively found the global minimum in 6 out of 7 cases (counting the result on the Crack 2 test case as successful, since it was within 0.03% of the global minimum value). In half of these cases, it also achieved this result faster than other algorithms. Its performance on Rosenbrock's function was equivalent to that of the other algorithms. In general, it achieved results as good as, or better, than all other algorithms, and its rate of convergence was highly competitive.

Algorithm	*Laser 1*		*Laser 2*		*Crack 1*		*Crack 2*		*Aerofoil*		*Bead*		*Rosenbrock*	
	Obj.	ESFE	Obj.	ESFE	Obj.	ESFE	Obj.	ESFE	Obj.	ESFE	Obj.	ESFE	Obj.	ESFE
EPSOC	**-0.48**	**10**	**-0.56**	**12**	**187.5**	**5**	5356	20	**-68.64**	20	**-39.85**	14	1e-3	14
GA	**-0.48**	14	**-0.56**	19	188.5	16	5346	11	-68.51	9	**-39.85**	**7**	2e-4	23
P-BFGS	**-0.48**	37	**-0.56**	39	**187.5**	26	5347	12	-67.90	26	-29.71	18	7e-2	10
Simplex	**-0.48**	24	**-0.56**	24	187.6	25	5353	23	**-68.64**	20	-26.98	16	**0**	54
SimplexL1	**-0.48**	15	**-0.56**	24	**187.5**	13	5331	12	-68.63	18	**-39.85**	13	5e-6	48
MDS	**-0.48**	16	**-0.56**	**12**	187.6	13	**5357**	1000	**-68.64**	14	-16.12	7	3e-4	1000
RSCS	**-0.48**	22	**-0.56**	**12**	187.6	9	5347	42	**-68.64**	**7**	-26.91	12	**0**	**39**
RSCSL	**-0.48**	22	**-0.56**	24	**187.5**	15	**5357**	**27**	-67.21	18	-26.98	11	6e-6	88
RSCSL1	**-0.48**	13	**-0.56**	17	187.7	11	5348	8	-68.62	15	-26.98	5	8e-4	33

Table 1. Best objective functions values obtained in 10 runs, and time taken

Conclusions

In this paper we have described a simple new Evolutionary Programming algorithm that utilizes concepts of Self-Organised Criticality. Tested on a range of cases drawn from real-world problems, against a representative set of direct search, gradient descent and genetic algorithms, it has been demonstrated to exhibit superior performance.

Examination of the results from the numerical experiments demonstrates population-based methods, EPSOC and the GA, are competitive against classical gradient descent and direct search methods providing they are executed on a parallel machine with sufficient processors to evaluate all of the population members in one iteration.. The quality of results obtained, and their speed in achieving them were clearly better in a majority of cases. Allowing EPSOC a larger number of iterations may further improve the result obtained on some problems (for example, on the challenging Bead test case, after 100 generations the **median** result from 10 runs lay at the global minimum).

Within the population-based methods, EPSOC outperformed the GA in finding the global minimum on all but one test case. Even for that case, the median result across multiple runs for EPSOC was better than that of the GA.

Like the GA, EPSOC is highly parallel, evaluating 1280 trial solutions in the same time it took Simplex, for example, to evaluate 81, and to generally better effect. The parallelism and completion time of the algorithm are set independent of problem size. The resulting pre-determined execution behaviour, in terms of resources required and time taken, its simplicity and its easily implemented master-slave parallelism make it well-suited for practical application to real-world problems.

Given the intended use of these algorithms in practical engineering design, they have all, with the exception of Genesis 5.0, been integrated into the Nimrod/O framework [12]. Nimrod/O is a design optimisation toolset. It makes it possible to describe a design scenario using a simple, declarative language, then makes use of parallel or distributed computers seamlessly to execute the experiment. Unlike other systems Nimrod/O combines optimisation, distributed computing and rapid prototyping in a single tool.

Further work is required on EPSOC to determine the efficacy of the method on problems of high dimensionality, and the link between problem dimensionality and optimal population size, extinction rates and mutation factors.

References

[1] Fletcher, R., "Practical Methods of Optimization", Wiley, New York, 1987.

[2] Lewis, A., Abramson, D., and Simpson, R., "Parallel Non-Linear Optimization: Towards the Design of a Decision Support System for Air Quality Management", *Proc. ACM/IEEE SC97 Conf.,* San Jose, CA, 1997.

[3] Nelder, J.A., and Mead, R., "A simplex method for function minimization", *Comput. J.*, 7, pp. 308-313, 1965.

[4] Wright, M.H., "Direct Search Methods: Once Scorned, Now Respectable", In Numerical Analysis 1995 *(Proc. 1995 Dundee Biennial Conference in Numerical Analysis)*, Griffiths, D.F. and Watson, G.A. (eds.), Harlow, UK, pp. 191-208, 1995.

[5] Torczon, V., "Multidirectional Search", Ph.D. thesis, Rice University, Houston, TX, 1989.

[6] Alander, J.T., "An Indexed Bibliography of Genetic Algorithms in Computer Aided Design", Report 94-1-CAD, Department of Information Technology and Production Economics, University of Vaasa, Finland, 1995.

[7] Durand, N., and Alliot, J-M., "A Combined Nelder-Mead Simplex and Genetic Algorithm", *GECCO'99: Proc.Genetic and Evolutional Computation Conf.*, Orlando, FL, 1999.

[8] Bak, P., "How Nature Works", Springer-Verlag, New York, 1996.

[9] Krink, T., and Thomsen, R., "Self-Organized Criticality and Mass Extinction in Evolutionary Algorithms", *Proceedings of the 2001 Congress on Evolutionary Computation (CEC2001)*, Seoul, Korea, 2001.

[10] Abramson D., Sosic R., Giddy J. and Hall B., "Nimrod: A Tool for Performing Parametised Simulations using Distributed Workstations", *The 4th IEEE Symposium on High Performance Distributed Computing*, Virginia, August 1995.

[11] Peachey, T., Abramson, D., Lewis, A., Kurniawan, D. and Jones, R., "Optimization using Nimrod/O and its Application to Robust Mechanical Design", Submitted for publication, *Fifth International Conference on Parallel Processing and Applied Mathematics (PPAM 2003),* Czestochowa, Poland, 2003.

[12] Abramson, D., Lewis, A., Peachey, T., Fletcher, C., "An Automatic Design Optimization Tool and its Application to Computational Fluid Dynamics", *Proc. ACM/IEEE SC2001 Conf.,* Denver, CO, 2001.

[13] Lewis, A., Saario, S., Abramson, D. and Lu, J., "An Application of Optimisation for Passive RF Component Design", *Conference on Electromagnetic Field Computation*, Milwaukee, June 4-7th, 2000.

[14] Grefenstette, J.J., "GENESIS: A system for using genetic search procedures", In *Proceedings of the 1984 Conference on Intelligent Systems and Machines,* pp. 161-165, 1984.