

**Southern California Earthquake Center
Annual Report, 1998
January 15, 1999**

Jean-Bernard Minster, Principal Investigator, Nadya P. Williams, and Heming Xu
Institute of Geophysics and Planetary Physics, 0225
Scripps Institution of Oceanography, UCSD, La Jolla, Ca, 92037-0225

Intermediate-term earthquake prediction algorithms

This SCEC research was targeted at SCEC Task 4: *Intermediate-Term Earthquake Prediction*. In support of the activities of Working Group A (*Master Model*). Our goal was to advance and finalize our ongoing worldwide test of the "M8" algorithm, originally developed for intermediate-term prediction of large events, which uses a catalog of mainshocks to identify large scale seismicity patterns before large earthquakes in a given region (e.g. *Gabrielov et al.*, 1986; *Keilis-Borok et al.* 1988; *Keilis-Borok et al.*, 1990; *Updyke et al.*, 1989; *Healy et. al*, 1992; *Kossobokov et al.*, 1992, *Keilis Borok and Rotwain*, 1994; *Kossobokov and Mazhkenov*, 1994). Unfortunately, the principal researcher on this problem was severely incapacitated for the majority of the grant period, so that this work was stopped for nearly a six-month period. We have recently restarted it, and are conducting a complete reprocessing of the worldwide seismicity for the period 1963-1998. We have evidence that our previous results suffered from a bias favorable to the M8 algorithm, due to our sampling strategy. The calculations are underway and will be reported at a SCEC seminar in the near future.

Separately, we have initiated a set of applications of *Evolutionary Strategies* to search systematically for geographical regions which lead to a most successful application of an arbitrary algorithm. This work was presented at the 1998 annual conference on *Evolutionary Programming*, and at the SCEC annual meeting. The attached report summarize the state of our progress to date, which is essentially limited to the achievements reached as of March, 1998, for the reasons mentioned above.

The events which are not predicted by the algorithm did not appear to be clearly segregated from those which are predicted; the focal mechanisms for both populations were quite similar. We propose to extend our examination through the end of 1996 and search further whether there are any differences between predictable and unpredictable events in this populations. We would like to apply our method to other applications, such as SEISMOLAP: a quantification of seismic quiescence and clustering (*Zschau*, 1995).

Strong motion and nonlinear wave propagation

Research on nonlinear wave propagation in soils in the strong-motion regime by Graduate Student Heming Xu resulted in a Ph.D. dissertation that was successfully defended in November, 1998. This research was partially supported by SCEC. The research, conducted jointly with Prof. s. Day of SDSU pertains to the application of the so-called *endochronic* nonlinear rheology. The Table of contents of Dr. Xu's thesis includes:

1. Introduction
2. Model for nonlinear wave propagation derived from rock hysteresis measurements
3. Comparison of perturbation and numerical solutions for one-dimensional nonlinear wave propagation

- 4.Hysteresis and two-dimensional nonlinear wave propagation in Berea Sandstone
- 5.Two-dimensional linear and nonlinear wave propagation in a half-space
- 6.Three-dimensional linear and nonlinear wave propagation
- 7.Hysteresis and nonlinear soil response
- 8.Concluding remarks

All but the last two chapters have been published or submitted for publication. Chapter 6 deals explicitly with very large scale nonlinear three-dimensional wave propagation problems, using a massively parallel computing environment.

References

- Bäck, T., *Evolutionary Algorithms in Theory and Practice*, Oxford, 1996.
- Bäck, T., Hammel, U., Schwefel, H.-P., *Evolutionary Computation: Comments on the History and Current State*, *IEEE Transactions on Evolutionary Computation*, vol. 1:1, 3-17, april 1997.
- Bowman, D. and Sammis, C., *SCEC Annual Report*, 1996.
- Fogel, D.B., *System Identification through Simulated Evolution*, Needham MA, Ginn Press Heights.
- Foley, J.D. and Van Dam, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing company, 1984.
- Gabrielov A.M. et al., *Algorithms of long-term earthquake prediction*, Int. School for research oriented to earthquake prediction algorithms, software and data handling, CERESIS, Lima, 1986
- Gardner J., and L. Knopoff, Is the sequence of earthquakes in southern California with aftershocks removed Poissonian? Yes, *Bull. Seismol. Soc. Amer.*, **64**, 1363-1367, 1974.
- Healy, J.H., V.G. Kossobokov, and J.W. Dewey, *A test to evaluate the earthquake prediction algorithm*, M8, USGS Open-file report, in press, 1992
- Keilis-Borok, V.I., L. Knopoff, V.G. Kossobokov, and I.M. Rotvain, Intermediate-term prediction in advance of the Loma Prieta earthquake, *Geophys. Res. Lett.*, **17**, 1461-1464, 1990.
- Keilis-Borok, V.I., and I.M. Rotvain, Diagnosis of times of increased probability (TIPs) for strong earthquakes in Northern Appalachians, *Comp. Seismol. and Geodyn.*, **1**, 1-5, 1994.
- Kossobokov, V.G. and M.G. Mazhkenov, Times of increased probability of large earthquake in the eastern Tien Shan diagnosed by the M8 algorithm, *Comp. Seismol. and Geodyn.*, **1**, 16-19, 1994.
- Kossobokov, V.G., J.H. Healy, and V.I. Keilis-Borok, Loma Prieta earthquake prediction by the M8 algorithm, (Preprint) 1992
- Kossobokov V.G., and J.H. Healy, Testing an earthquake prediction algorithm, *Science* submitted, 1994
- Keilis-Borok, V.I., L. Knopoff, and I.M. Rotvain, Bursts of aftershocks long term precursors of strong earthquakes, *Nature*, **283**, 259-263, 1980.
- Minster, J.B., and N.P. Williams, The "M8" intermediate-term earthquake prediction algorithm: an independent assessment, *EOS Am. Geophys. Union Trans.* Oct. 1992.
- Minster, J.B., and N.P. Williams, A systematic approach to assess M8-class intermediate-term earthquake prediction algorithms, *IASPEI 27th General Assembly, Wellington, N.Z. Abstr. S6-21* 1995a.
- Minster, J.B., and N.P. Williams, Performance of the M8 Intermediate-term earthquake prediction algorithm, assessed from bootstrap estimates, *IUGG XXI Gen. Assembly, Abstr. A-401*. 1995
- Minster, J.B., and N.P. Williams, Worldwide performance of a seismicity-based intermediate-term prediction algorithm: The M8 algorithm, $M > 7.5$, 1985-94, *SCEC Annual Meet.* p.74 Oct. 1995
- Molchan G.M., and O.E. Dmitrieva, Aftershock identification: methods and new approaches, *Geophys. J. Intern.*, **109**, 501-516, 1992.
- Updyke et al., *Proc. NEPEC June 6-7, 188, Reston VA*, USGS Open-file report 89-144, 1989
- Rothman, D.H., 1985, Nonlinear inversion, statistical mechanics, and residual statics estimation, *Geophysics*, **50**, 2784-2796.
- Sornette, D. and C. Sammis, Complex critical exponents from renormalization group theory of earthquakes: Implications for earthquake predictions, *J. Phys. L.*, 607-619, 1995.
- Sornette, D. and C.G. Sammis, Universal log-periodic correction to renormalization group scaling for regional seismicity: Implications for earthquake prediction. *SCEC annual meeting, 1994, preprint*.

Use of Evolutionary Programming to Identify Regions of Specific Seismicity

Abstract

Analysis of space-time seismicity patterns is an important tool used by many researchers who attempt to progress towards earthquake prediction. Typically, the selection of the best time and space windows for a particular application is made “by hand” using an empirical knowledge of possible criteria. It is, therefore, a common question: what is the best space-time seismicity sample in which the analyses can be performed, and how to identify it? This problem falls into a class of global optimization problems for which the fitness functions are not continuous, and where Evolutionary Algorithms can therefore provide an effective approach. As an example, we are using Evolutionary Programming (EP) to find the smallest rectangle in a given geographical area such that the earthquake population it contains possesses certain characteristics.

Problem Description

Consider the following problem: Given an earthquake catalog, find the smallest rectangle in a given space such that it contains N epicenters from the catalog, and includes a selected target point E (point for which we might make a prediction by further extending the algorithm from this simple search to a “predictive stage”).

The basic idea that we want ultimately to pursue is that for some earthquake data the release of regional energy (prior to a large earthquake) appears to exhibit log-periodic fluctuations in the vicinity of the critical point. We calculate energy as a function of time, and an objective measure of fitness is the quality of fit of energy to a power law which can help to determine the time occurrence of the mainshock [Sornette and Sammis].

Before going into the calculation of energy an efficient algorithm has to be found that will allow a fast choice of the space-time windows. To study this issue we look first at the elementary problem of just finding rectangles with a desired earthquake count. We can extend it later to any complexity of energy calculation. Usually, one chooses a critical time window and spatial region manually based on prior knowledge of the seismicity and physical processes that govern it.

Any function calculated from the earthquake catalog (consisting of hundreds of thousands of events) is a discontinuous (usually non-linear) function of a point process in space and time. Iterative optimization techniques such as least-squares are unfeasible here since functions constructed from the catalog are typically not differentiable. Methodical step searches in time and space (grid searches such as Monte Carlo methods) are impractical because of the size of the problem. In contrast, Evolutionary Programming is a stochastic optimization technique that can be applied to a problem of finding a global extremum of non-linear function, and this project is to examine its applicability to the analysis of seismicity.

Implementation

A generic EP Algorithm normally consists of the following components [Bäck]:

- data representation
- fitness-proportional reproduction algorithm
- set of genetic operators

We are using $(\mu+)$ -EP to implement the algorithm. $(\mu+)$ means that μ parents generate μ offsprings (so far we have used one offspring per parent) through mutation at each generation. The best μ offsprings are selected deterministically from the combined population of parents and offsprings, and replace the parents. The algorithm can be expressed in the following steps.

- (1) Construct the initial population of N rectangles. Each is a 5 dimensional random vector P_i uniformly distributed over a feasible range in each dimension. All rectangles are subject to a single constraint - to contain a target point. This is the initial “parent” population.
- (2) For each P_i create a new member of the population P_{N+i} by altering the vector’s P_i parameters via random mutation, e.g. each component of the vector P_i is perturbed by a Gaussian random variable with mean 0 and adaptable standard deviation.
- (3) For each member of total population P_{2N} assign a fitness score F_i , where fitness is an objective function of arbitrary complexity (depending on the problem). The fitness function is evaluated based on the values of the vector P_i and possibly some other parameters. The fitness function measures how poorly or how well the member satisfies the optimization criterion.
- (5) For each member compute a score value S_i based on a probabilistic tournament where each solution is placed in competition against M randomly selected opponents. The solution receives a “winning score” if it outperforms the opponent. Assign a score between $0 \leq S_i \leq M$ based on the number of competitions that were won.
- (6) Select N members with the highest scores as a new “parent” population, discard the remaining ill-fitted solutions.

Steps (3)-(6) are iterated until a suitable solution is found or until available computer time is exhausted.

The main algorithm is thus straightforward. A set of difficulties comes from the fact that each time a rectangle is generated we have to go through the catalog of events and find what events are contained in this rectangle. Thus some operations are repeated over and over again, and need to be optimized via preprocessing and usage of appropriate data structures that would minimize the execution time.

Parameterization of rectangles.

1. Creation of rectangles can be expressed in terms of applying translation, scaling and rotation operations to a single initial rectangle. The efficiency is important since we have to create many new rectangles at each generation. Homogeneous coordinates [Foley] allow us to do these transformations easily and in terms of only multiplication and addition operations. In homogeneous coordinates a point $P(x, y)$ is represented as $P(W \ x, W \ y, W)$ for any scale factor $W > 0$, and where x and y are cartesian coordinates. Given a point $P(X, Y, W)$ in homogeneous coordinates, we can find the corresponding cartesian coordinate representation by $x = X/W$ and $y = Y/W$. If $W = 1$ then division is never required. Each rectangle can be described by five spatial dimensions: length a , width b , translation coordinates (x, y) , μ , and azimuth θ (angle of rotation counterclockwise). These five parameters represent the vector P_i that is mutated at step two of

evolutionary algorithm. These five components of P have the following ranges:

$$\begin{matrix} 0 & a \\ 0 & b \\ -1 & & 1 \\ -1 & \mu & 1 \\ - & & < \end{matrix}$$

To generate an arbitrary rectangle that contains a target point E we start with a unit rectangle in homogeneous coordinates, such that its center is at the origin [0, 0] and the sides are equal to 2, using the following sequence of operations:

- translation by (,μ); the center of rectangle goes to [, μ].
- scaling by (a, b); the sides of rectangle become length a, and width b.
- rotation of scaled rectangle about the origin by an angle .
- translation of scaled and rotated rectangle to the target point E.

The compound transformation for any point $P = [x, y, 1]$ is $C = P \cdot T \cdot S \cdot R \cdot T_E$ where $C = [X, Y, 1]$, X and Y are the cartesian coordinates of the translated point, and T, S, R, T_E are the matrices for the translation, scaling, and rotation operations. The coordinates of the four corners of any rectangle can now be easily found using 4 multiplications and 4 additions:

$$\begin{aligned} X &= a \cos \quad (x +) - b \sin \quad (y + \mu) + X_E \\ &\text{and} \\ Y &= a \sin \quad (x +) + b \cos \quad (y + \mu) + Y_E \end{aligned}$$

Optimization of the earthquak e count

Since the earthquake catalog consists of hundreds of thousands of events, and even a small area of interest of a size of a few degrees in latitude and longitude can include tens of thousands of events, the operation of finding the earthquakes inside the rectangle can become very expensive in terms of computer time. Thus it is desirable to minimize the time of this operation.

To do this we combine the preprocessing of the catalog with the application of Bresenham’s algorithm. First, the earthquakes coordinates are converted using an equal area sinusoidal projection. In the sinusoidal projection the latitude and longitude are transformed into new coordinates u and v as:

$$\begin{aligned} u &= \cos \\ v &= \end{aligned}$$

This allows us to use a grid imposed upon an area of interest, and have equal grid steps in both directions. The area of interest (latitude and longitude range) is converted using the same sinusoidal projection, and is divided into cells (the size of which can be adjusted) of equal areas. Each cell has integer coordinates (i, j), and is assigned a variable that holds the earthquake count for it.

For each earthquake from the catalog we find the integer coordinates of the cell inside which this earthquake falls, and update the variable count for that cell. The procedure is executed only once before any other calculations.

Each rectangle has four corners for which we can find the integer coordinates on the grid. Now the lines connecting the corners (or the sides of rectangle) can be evaluated by Bresenham's line algorithm where the algorithm "draws" the line on a discrete set of points (which are presented by the cells of the grid). The line would appear jagged if one were to picture it, due to the integer approximation. To count the earthquakes inside this approximated rectangle, we simply do a summation of the cell earthquake counts using only the cells that are between the lines, including those cells that represent the lines (sides) of the rectangle.

The combination of a grid presentation and the Bresenham algorithm allow us to use only integer arithmetic; no real variables are used. The arithmetic needed to find the lines and count the earthquakes on the subset of interior cells is minimal: it involves only addition and subtraction. This implementation gives us a considerable speed-up since we avoid multiple comparison operations that otherwise would be needed to find if each earthquake was inside the given rectangle.

Details of EP implementation

1. Parent and Offsprings Populations.

For the present investigation, we have used parent populations sizes of $N=100$ to $N=200$. Too small a population can lead to a "premature convergence." We have allowed only one offspring for each parent. Experiment with variable number of offsprings per parent as a function of fitness will be considered in the future.

2. Fitness functions.

We defined the *fitness function* F in terms of the *score function* S , which, in turn, is defined by up to three independent *penalty functions* P_e, P_a , and P_r . These terms are dealing with the number of earthquakes in the rectangle, the area, and the aspect ratio. All these functions can be constructed arbitrarily because Evolutionary Programming doesn't require differentiability of the fitness function.

The penalty function for the number of the earthquakes inside the rectangle is

$$P_e = \exp(- (N^2/N_0 - N)) \text{ for } N \leq N_0$$

and

$$P_e = \exp(- (N - N_0)) \text{ for } N > N_0$$

where N_0 is desired number of earthquakes inside the rectangle, N is the actual number of the earthquakes inside the rectangle, and α is a parameter, $0 < \alpha < 1$. The rectangles with fewer earthquakes than N_0 are penalized more than rectangles with more earthquakes than N_0 .

The penalty function for the area (or size) of the rectangle is

$$P_a = \exp(-A/A_e)$$

where A_e is the area per event in the model box, A is the area of the rectangle, and is a parameter, $0 < < 1$.

The penalty function for the aspect ratio is

$$P_r = \exp[-(r-1)]$$

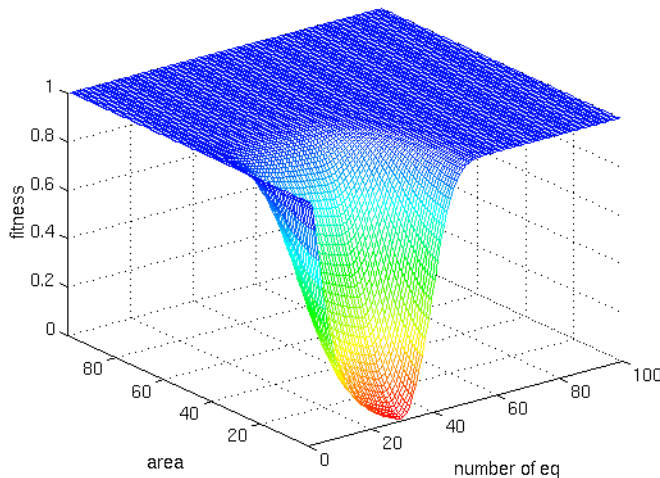
where r is the aspect ratio (width/length) of the rectangle defined as $r = b/a$ for $b < a$, and $r = a/b$ for $b > a$, and is a parameter, $0 < < 10$.

The parameters , , are used to define the relative importances of the three penalty terms. The following score functions have been tried:

$$S = (P_e + P_a + P_r)/3$$

$$S = (P_e + P_a)/2$$

$$S = P_e P_a$$



The *fitness function* is then defined as:

$$F = (1 - S)/(1 + S)$$

so that as small a fitness function as possible is desirable. The latter of three score functions gives a sharper minimum for the fitness functions (fig. 1).

Figure 1. Example of the fitness function.

3. Offsprings Generation.

We generate a single offspring per parent by applying Gaussian perturbations to each of the five parameters that define the rectangle. The perturbations are of the general form

$$(F - F_r)^{0.4} N(0, 1)$$

where F is the fitness value of the parent, F_r is a *reference fit*, and $N(0,1)$ is a random number drawn from a Gaussian distribution with zero mean and variance 1. Thus models with small fitness values are only slightly perturbed, while large fitness values will allow for large perturbations, and thus for the sampling of the space in “remote” areas compared to the parent rectangle.

The reference fit is defined via the fitness of the two best models of the population as $F_r = 2 \text{ BestFit} - \text{NextBestFit}$. Since the fitness function does not converge to zero, the *reference fit* fitness differential to which we scale the perturbations applied to generate offsprings is adjusted according to the tail of the fitness distribution for the best members of the population.

4. Competition Coupled with Simulated Annealing

The number of the opponents M in the competition must also be chosen, and [Fogel] recommends M to be between 10 and 20% of the total population. Each of the $2N$ members of the parents-and-offsprings population is competed against M other members chosen at random. Using the competition algorithm described in Fogel (1991), we have sometimes observed the population to stagnate in a neighborhood representing a local minimum of the fitness function, but clearly an incorrect solution. In order to help prevent such instances of premature convergence to a local minimum, we have found it effective to combine EP with a simulated annealing schedule, by defining the score S_A of population member A as follows (e.g. Rothman 1985):

- (1) Select a random competitor B and let $S = S_A - S_B$ be the difference in their fitness values.
- (2) If $S \leq 0$, then increase the score S_A by one.
- (3) If $S > 0$, then increase the score S_A by one (i.e. declare A to be a winner over B) with probability given by a Gibbs distribution:

$$P(S) = e^{-s/T_0^j}, \text{ where } j \text{ is the generation index.}$$

The initial temperature T_0 is set to the median of the fitness values for the initial population. The cooling parameter is then set to a value such that the temperature becomes small over 30 to 50

generations. This approach means that in early generations, even relatively poor solutions are given a chance to survive, and their offspring are generated through relatively large perturbations, thereby improving the sampling of the solution space. On the other hand, in later generations, the temperature is low and the competition is reduced to a rather straightforward comparison of fitness values, favoring solutions that provide even a minor improvement in the fit.

5. Time considerations

The data structure that is used by the “evolutionary” part of the algorithm has to access each member of the population at each generation for setting the parameters defining the rectangles, for the calculation of fitness values and scores, and for the generation of offsprings.

The modification of the data structure at each generation involves cutting half of the population based on the ordered (descending) scores values.

Since the access operation is the most common, it would be preferable to use a structure that uses $O(1)$ time for this operation, and balance the time for the modification as well. We are using an array of rectangles, where each rectangle has its own structure described by a class (in C++ terms). The access of any array member for any function consideration is $O(1)$, giving a total $O(N)$ for a population of size N , and the worst time for sorting using QuickSort is $O(N\log N)$. The total running time of the algorithm can be viewed as follows:

Initialization step

function	timing
Set Initial Population of a size N	$O(N)$
Find Population’s Fitness	$O(N)$
Set Annealing Temperature, which is a median fitness value of the whole population. Do sorting by fitness	$O(N\log N)$
Find Initial BestFit, and NextBestFit (array is already sorted from the previous step)	$O(1)$
Competition	$O(N)$

Evolutionary step, repeat while reaching the determined number of Generations G

function	timing
Create new generation	$O(N)$
Choose the best half of the population	$O(N\log N)$
Generate offsprings	$O(N)$
Find Population fitness	$O(N)$
Competition	$O(N)$

At this point we assume that the calculation of the fitness function, adjusting the temperatures,

and creating the offsprings takes some constant time, a reasonable assumption since the population size is not allowed to vary from one generation to the next. Then the estimate for the total running time of the algorithm is $O(GN\log N)$ for G generations and N -size of the population.

Even though there are quite a few papers in the literature which offer applications of Evolutionary Algorithms to numerous fields, not many works include theoretical discussions of the performance. So far, the performance is typically described by empirical results for a particular instance of a problem, and “the theoretical foundations are to some extent still weak” [Bäck, Hammel, Schwefel].

However, the efforts invested in this field are justified by gain in flexibility and adaptability to solve the problem at hand, combined with the robust performance when the approach is successful. Another advantage of Evolutionary Algorithms is that they can easily be ported to parallel computer architectures giving a considerable speed-up linear with the number of processing units p as long as p does not exceed the population size N .

Summary

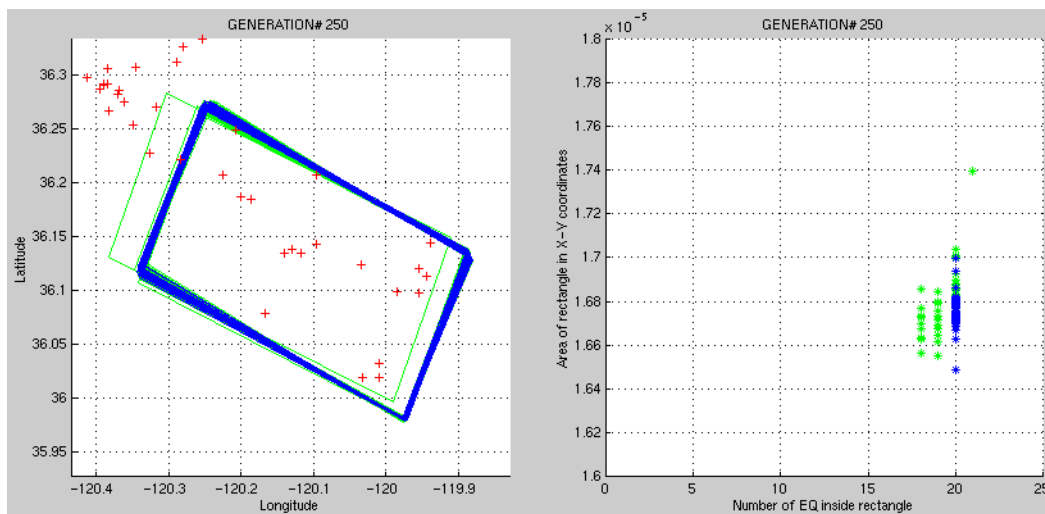


Figure 2. Example of the program output for 250 generations.

The results of one program execution are shown in figure 2. The target point is 36° latitude and -120° longitude, the desired number of the earthquakes inside the rectangle is 20. The score function is of the form $S = P_e P_a$, and the parameters are $P_e = 0.02$, and $P_a = 0.001$. The choice of the parameters gives more weight (in the score function) to the number of earthquakes inside the rectangle than to the area of rectangle. The left subplot shows the last generation of parents (blue) and offsprings (green), and the earthquakes (red crosses). The right subplot shows the values of the fitness function vs. the number of the earthquakes inside the rectangles, for parents (blue), and offsprings (green). As one can see, the perturbations at the last generation are still large enough to create ill-fitted offsprings, but the solution found is quite acceptable.

This implementation of the algorithm still has unsolved problems such as numerous local optima, and lack of differentiability of the fitness function. The creation of the offsprings is not scaled perfectly (we arrive at the solutions where the algorithm finds the rectangle with the desired properties but not the best geometry (when this rectangle can still be further minimized in terms of size). We need to establish the criteria to quit the execution automatically when the “best” solution is found instead of giving a rigid limit which can be insufficient for some instances, and “overkill” for others.

Nevertheless, in spite of remaining uncertainties concerning these issues, the combination of evolutionary programming and simulated annealing schedule leads to an acceptable solution with sufficient reliability to be practical. We are therefore encouraged to continue the development and to go on to compute more complex functions of the seismicity, in particular the release of energy with time inside each rectangle. The fitness function will then be changed to reflect how well the characteristics of the history of energy release matches a theoretical pattern (e.g. log-periodic oscillations superposed on a power-law growth curve [Bowman and Sammis].)

References

- Bäck, T., *Evolutionary Algorithms in Theory and Practice*, Oxford, 1996.
- Bäck, T., Hammel, U., Schwefel, H.-P., *Evolutionary Computation: Comments on the History and Current State*, *IEEE Transactions on Evolutionary Computation*, vol. 1:1, 3-17, april 1997.
- Bowman, D. and Sammis, C., *SCEC Annual Report*, 1996.
- Fogel, D.B., *System Identification through Simulated Evolution*, Needham MA, Ginn Press Heights.
- Foley, J.D. and Van Dam, A., *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing company, 1984.
- Rothman, D.H., 1985, Nonlinear inversion, statistical mechanics, and residual statics estimation, *Geophysics*, 50, 2784-2796.
- Sornette, D. and C. Sammis, Complex critical exponents from renormalization group theory of earthquakes: Implications for earthquake predictions, *J. Phys. I.*, 607-619, 1995.