Visualization Tool for a Terrain-Based Genetic Algorithm

V. Scott Gordon
CSU, Sacramento
gordonvs@ecs.csus.edu

James Thein
Genetech, Inc.
jthein@gene.com

Abstract

We describe and implement a visualization tool applet for a Terrain-Based Genetic Algorithm (TBGA). The TBGA is a self-tuning version of the Cellular Genetic Algorithm (CGA), wherein various combinations of parameter values appear in different physical locations of the population. The TBGA is useful for solving optimization problems as well as for finding good CGA parameter values. By tallying the number of times a new best individual is found for each location in the population, the applet illustrates the progress of evolution as a gradually evolving terrain map showing effective locations as having increasing altitude. We contrast two methods for using the TBGA to determine good parameter settings. The tool can also help educate users unfamiliar with the TBGA and how it works.

1. Introduction

The Terrain-Based Genetic Algorithm (TBGA) is a self-tuning version of the Cellular Genetic Algorithm (CGA). In a TBGA, various combinations of parameter values appear in different physical locations of the population, forming a sort of terrain in which solutions evolve. In a previous study [8], the TBGA was shown to perform better and with less parameter tuning than a CGA on a suite of test problems, when the CGA was set to parameter values used in an earlier set of experiments [7] and thought to be good.

The TBGA was also used to automatically determine good parameter settings for the CGA. The resulting CGA produced even better results than were achieved by the TBGA that found those settings. Thus, the TBGA not only displayed good potential as a function optimizer, it also was a powerful tool for extracting better performance from a CGA, and required less parameter tuning.

The nature of the TBGA algorithm lends itself to visualization. In particular, a visualization tool is useful both in assisting new users in understanding the algorithm, and as a research aid for studying how best to utilize the TBGA for extracting aggregate parameter information. In this study, we describe a visualization method and Java applet tool that supports these goals.

2. Background - CGA / TBGA

Selecting good parameter values (e.g., mutation rate) can be complicated, and is affected not only by the the genetic algorithm itself, but also the nature of the optimization problem [1, 3, 4, 10, 13]. A *self-tuning* genetic algorithm attempts to determine suitable parameter values without user intervention. In the TBGA, this is accomplished by spreading a range of parameter values along the population axes of a CGA.

Cellular genetic algorithms (CGA) - sometimes called massively-parallel GA's - assign one individual per processor and limit mating to within demes (neighborhoods). CGAs are usually simulated on a single processor with a 2-dimensional matrix. There are many types of CGA's, and by 1989, numerous researchers had developed the same concept independently [9, 11, 12]. Whitley's work with a cellular automata model of these algorithms [15] gave rise to the term CGA.

The TBGA is based on a common CGA described in a previous study [6] called a *fixed-topology Deme-4* CGA. Each individual is processed at every generation, and an individual's mate is selected from the best of the four strings located above, below, left, and right (see Figure 1).

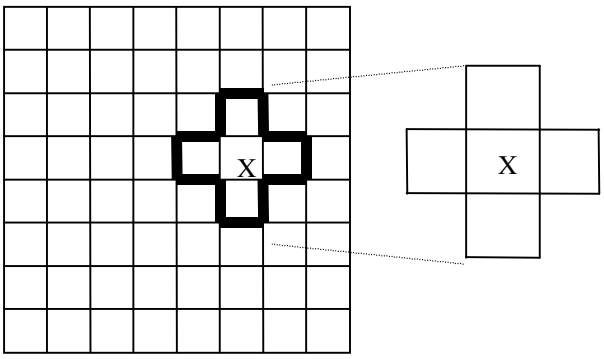

Figure 1. The Deme for Node X in a Deme-4 CGA

Crossover is always performed, yielding two offspring, and mutation is then applied to each offspring. If either resulting offspring has a better or equal fitness than the original node, then that node is replaced by the most fit offspring. Edge elements wrap around, forming a torus. We use reduced-surrogate crossover.

Next, parameters are chosen as *terrain variables*. Since the CGA described has two dimensions and therefore two axes, we chose two parameters as terrain variables: *mutation rate* and *number of crossover points.* A range of values for those parameters is spread along the axes of the CGA, so that strings residing in different physical locations in the population structure are subjected to different combinations of settings (although the parameter values in neighboring cells are similar). The variances in parameter values over the space form a sort of terrain, and that is why such an algorithm is called a *terrain-based* genetic algorithm.

Consider the 8x8 CGA in Figure 2. In this example, mutation rates are spread along the X-axis. The maximum mutation rate is 35%, and the minimum rate is 0%. Mutation rates for each cell are shown along the x-axis at the bottom of the grid. Similarly, settings for the number of crossover points are spread along the Y-axis, where the values range from 1 to 8. Note that every cell in a given column has the same mutation rate, and every cell in a given row has the same number of crossover points. To achieve a smooth distribution of parameters, while still retaining the advantages of a toroidal CGA topology, parameters are arranged using a process called *sifting*, in which the maximum value is placed in the center, then each decreasing value is placed on alternating sides.

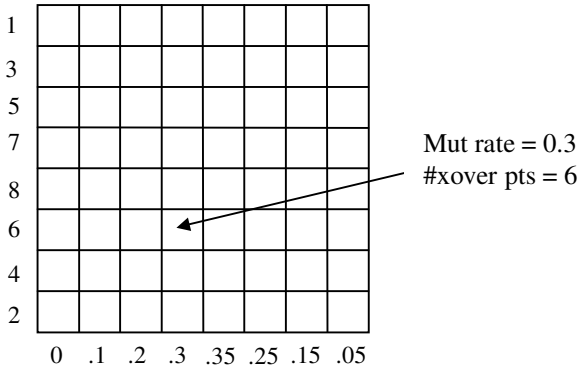

Figure 2. Mutation Rate (along x-axis) and Number of Crossover Points (along y-axis), after sifting

Even though several cells share the same mutation rate (for example), every cell has a different combination of parameters. Thus, in a TBGA with parameters shown as in Figure 2, cells in the far upper left have low mutation rates and few crossover points; cells in the left center area have low mutation rates and many crossover points, etc. In theory, a TBGA could utilize a wide variety of parameter combinations, by extending the CGA to more dimensions as has been proposed by some researchers [2].

3. Finding Good Parameter Settings

The TBGA proved to be a useful tool in finding good parameter settings for use in a *standard* CGA. In the original study, we did this by identifying the physical locations in the population structure where the best solutions evolved [8].

The TBGA reports the location of each newly found current best string. Sometimes, these locations cluster around specific areas, but many times they do not. When there is clear clustering, it is natural to suspect that a good set of parameters is found at the *highest physical peak* in the terrain. When there is not clear clustering around a peak, a *weighted average* of the parameter settings at each location can be used.

For example, suppose we were using a 3x3 grid, and say that we ran the TBGA for 10 generations, tallying for each cell how often a new current best string appeared there. Since each cell also has its own value for mutation we might tally data similar to that shown in Table 1.

Table 1. Example Tally for Finding Mutation Rate

CELL	Mutation Rate	#bests
(1,1)	.00	0
(1,2)	.00	0
(1,3)	.00	2
(2,1)	.01	0
(2,2)	.01	5
(2,3)	.01	0
(3,1)	.02	3
(3,2)	.02	0
(3,3)	.02	0

The most good solutions were found in cell (2,2). If we use the *highest peak* method for finding parameter settings, we would propose using .01 as the mutation rate, since that is the mutation rate associated with cell (2,2). If, however, we use the *weighted average*

method, the suggested mutation rate would then be the sums of the mutation rates, weighted by the tallies, or:

```
[5*(.01) + 3*(.02) + 2*(0)]/10 = .011
```

In the original study, the weighted average technique was used to compute mutation rate and crossover points, and the resultant CGA performed markedly better than both TBGA and the original CGA [8]. However, there was no particular reason to believe that *weighted-sum* was necessarily the most effective approach. If solutions clearly cluster around one good setting, *highest peak* may be a better choice.

Furthermore, consider a scenario in which good solutions cluster around *two* peak areas of the graph, wherin a weighted sum would reflect a location between the two peaks, where **no** good solutions have evolved. In this instance, it is not clear whether weighted sum or highest peak would be preferable.

Finally, there is the possibility that certain parameter settings are useful at different times during search. If so, then perhaps the tallies could be used to generate a schedule for dynamic parameter settings.

4. Visualization Tool

The terrain model described in the previous sections lends itself to 3-D visualization. We now describe our TBGA visualization Java applet.

At each generation, the best string or strings reside at particular locations on the grid. As evolution proceeds, the TBGA counts the number of times each grid location contains an individual that has the new best fitness. These tallies are displayed as an altitude corresponding to each grid point, producing a 3D terrain map where the highest peaks represent locations where the best strings have evolved.

It is important not to confuse the various terrain analogies being employed here. The distribution of parameter values through the grid can be considered one such terrain. The efficacy of each grid location, and thus each set of parameters, as illustrated during the course of evolution, is another terrain. Our current visualization efforts are concerned with the latter.

The tool builds and displays the evolution terrain map described above in real time, and renders it both as a wire-frame and as a solid model within a standard browser. The tool also allows the user to select an optimization problem from a list, the size of the grid, the number of generations, and the number of experiments to run. As the resulting terrain map grows over time, the user can rotate it so as to view it from various perspectives. Figure 3 shows the tool and a sample of the visualization it provides.

Currently, we have only entered a few problems into the prototype, taken from our previous study [8], including the *Rastrigin* function, the *Ugly 4-bit Deceptive* function, *DeJong's F2*, and a *20-object knapsack* problem. The first two are frequently used as GA test problems. The Rastrigin function was described in [12], and is characterized by a large search space and many local minima. The Ugly 4-bit Deceptive problem [14] is a 40-bit artificially-constructed problem in which ten fully-deceptive 4-bit subproblems are interleaved. Whitley's problem is based on a similar 3-bit problem introduced in [5].

Figure 3 shows the terrain map as displayed by the applet after 30 experiments of 500 generations each, for the Rastrigin function. Each point on the grid represents a location in the CGA population, and the altitude illustrates how many individuals with a *new best-fitness* have appeared at that location. The higher the altitude, the more effective the parameters at that location have been at generating better solutions.

Figure 3 also illustrates a run in which two peaks are clearly visible in the solid model. Mutation rate is roughly the same for each peak, but the setting for crossover points is different for each peak.

We ran similar sets of experiments on each of the other three functions. The wire-frame terrain shown in Figure 4 corresponds to the Ugly-4-bit function, and is representative of the terrains generated for the remaining functions. The black dots shown on the grid are the locations of the *highest peak*, and *weighted-sum* locations. Note that there is not as clear a peak as there is for the Rastrigin function.Also note that the highest-peak method and the weighted sum method do not agree on which parameter settings are optimal. The parameter settings suggested by each method, for each of the four problems, are shown in Table 2.

Table 2. Optimal parameter settings as determined by TBGA

Problem	method	mutation	xover pts
Ugly 4-bit	highest peak	.0469	4
	weighted avg	.0165	7
DeJong F2	highest peak	0	7
	weighted avg	.0371	5
Knapsack 20	highest peak	.1219	2
	weighted avg	.0372	4
Rastrigin	highest peak	.0028	7
	weighted avg	.0028	10

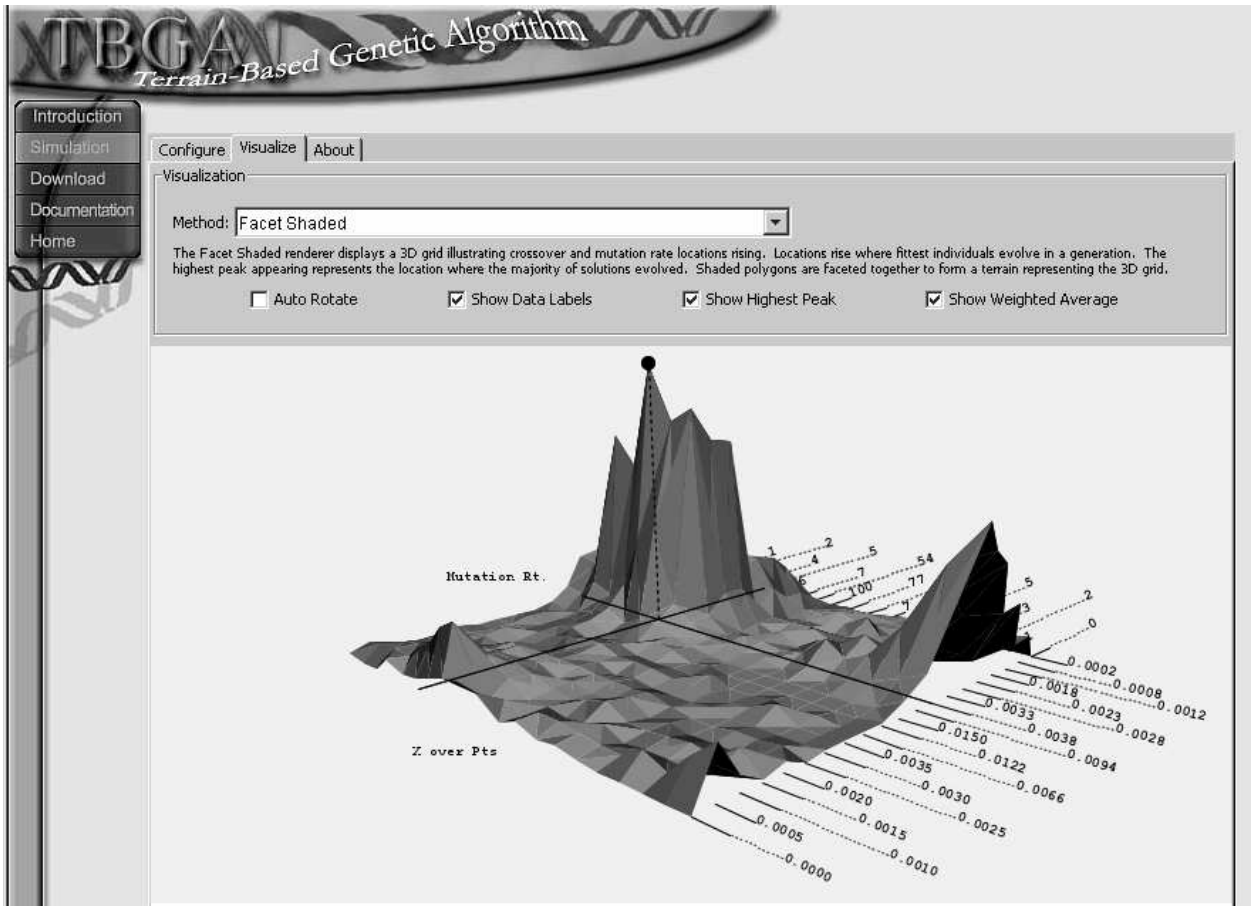

Figure 3. TBGA Visualization Applet, solid model.
Highest peak and corresponding parameter values found for the Rastrigin function.

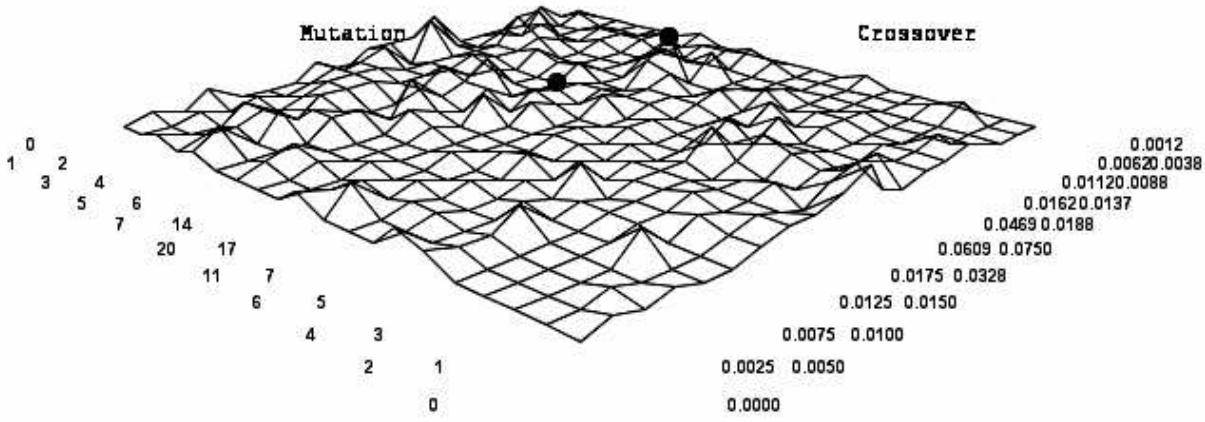

Figure 4. Wire-mesh parameter visualization for the *Ugly 4-bit Deceptive function*

5. Evaluation and First Assessment

The parameters settings suggested by the TBGA vary considerably depending on the method used (highest peak versus weighted average). In order to determine which recommendation(s) tended to be the best, we compared them using a standard CGA for each of the parameters settings suggested from Table 2. The results are shown in figure 5.

The weighted-sum method is slightly better on the Ugly 4-bit problem and the Rastrigin function, but the highest peak method works the best on F2 and the Knapsack problems. Furthermore, contrasting the two methods in light of the differing shaped terrains is also a bit inconclusive. The convergence graphs agree very strongly on the Rastrigin function, presumably because the mutation rates proposed were identical - mutation rate is likely the more critical of the two parameters.

The terrain shown in figure 3 is markedly different than the terrain in figure 4. Perhaps we should not limit the TBGA to 30 experiments, but instead to run it until a peak emerges. This would be a particularly useful test for the Ugly 4-bit problem, since the inferior performance of the highest-peak method reflected a peak that was, in actuality, not much of a peak.

It is important to note that, in all cases, performance of the CGA remains significantly better than for parameter settings identified previously (without the TBGA) [7,8]. In that sense, it appears that *both methods work well.* Further study is needed on a broader set of problems to determine which method usually performs the best, or is more consistent.

6. Other Uses for Visualization

The TBGA is unique and the terrain analogy is sometimes not easily grasped by users not already expert in the field of evolutionary computation. Visualization will help elucidate and educate new users considering utilizing the TBGA for their particular applications. Since a Java applet is unlikely to perform as well on difficult problems as, say, our original C++ implementation, the visualization can be accompanied by production code that can be downloaded if the applet proves insufficiently powerful.

In addition to the educational benefits of visualization, the tool would greatly facilitate additional study in several areas:

- The clustering behavior of the TBGA needs further study. Perhaps solutions would tend to cluster around particular good parameter settings if averaged over significantly more experiments.
- If a pattern of parameter utilization can be discerned by observing the nature of terrain growth, it may lead to the development of scheduling methods for dynamic parameter settings.
- Observing the resultant terrains may suggest alternative methods for parameter determination, hopefully superior to the weighted-sum or highest-peak methods.

7. Conclusions

We described a visualization tool for a terrain-based genetic algorithm (TBGA), and implemented it as a Java applet. The TBGA is a self-tuning version of the traditional Cellular Genetic Algorithm (CGA), in which various combinations of parameter values appear in different physical locations of the population, forming a sort of terrain in which individual solutions evolve. It is useful both for solving optimization problems, and also for finding good parameter values to use in a standard CGA.

By tallying the number of times a new best individual is found for each location in the population, we illustrated the progress of evolution as a gradually evolving terrain map showing effective locations as having increasing altitude. Visualization using both solid and wireframe rendering was described including several sample outputs for various test problems. Two methods were described and tested for identifying parameter values: *highest peak* and *weighted sum*, but it is not yet clear which method is preferable.

Many uses for the tool were described, including applications for education and research. In the area of education, the tool can help educate new users unfamiliar with the TBGA and how it works. It is anticipated that the tool will help us refine our techniques for using the TBGA to identify good CGA parameter settings, as well as to better understand the nature of evolution within a TBGA.

8. Acknowledgement

The visualization tool described in this paper was developed as part of a team project at CSU Sacramento. We thank students Michael Pope, Matthew Santa Cruz, Robert Carlin, Reuben Fauth, and James Sanguinetti for their work in this capacity, and Dr. Gopal Rao for his supervisorial assistance.

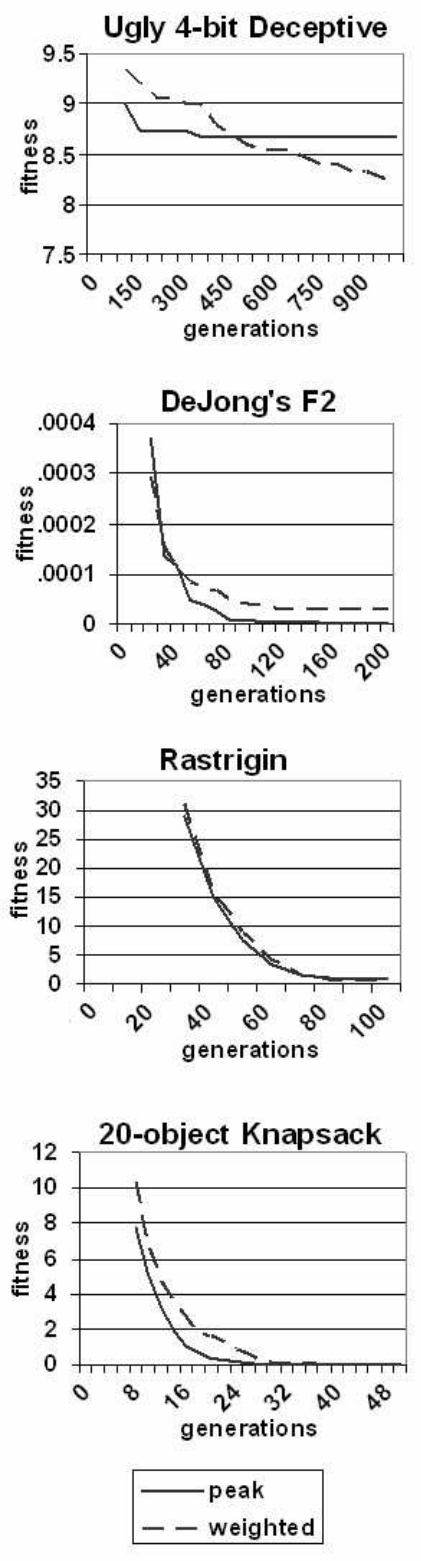

Figure 5. Convergence of CGA using TBGA-determined parameters, *peak* vs. *weighted-sum*

9. References

[1] Back, T. *Optimal Mutation Rates in Genetic Search*. Proc. of the 5th Int. Conf. on Genetic Algorithms (1993) 2

[2] Baluja, S. *Structure and Performance of Fine-Grain Parallelism in Genetic Search*. Proceedings of the 5th International Conference on Genetic Algorithms (1993) 155

[3] Davidor, Y. and Ben-Kiki, O. *The Interplay Among the Genetic Algorithm Operators: Information Theory Tools used in a Holistic Way*. PPSN2 (1992) 75

[4] Goldberg, D. *Sizing Populations for Serial and Parallel Genetic Algorithms*. Proceedings of the 3rd International Conference on Genetic Algorithms (1989) 70

[5] Goldberg, D., Korb, B., and Deb, K. *Messy Genetic Algorithms: Motivation, Analysis, and First Results*. Complex Systems 3 (1989)

[6] Gordon, V., Mathias, K., and Whitley, D. *Cellular Genetic Algorithms as Function Optimizers: Locality Effects.* ACM Symposium on Applied Computing (SAC'94) (1994)

[7] Gordon, V. and Whitley, D. *Serial and Parallel Genetic Algorithms as Function Optimizers*. Proceedings of the 5th International Conference on Genetic Algorithms (1993) 177

[8] Gordon, V., Pirie, R., Wachter, A., and Sharp, S. *Terrain-Based Genetic Algorithm (TBGA): Modeling Parameter Space as Terrain*. Proceedings of the Genetic and Evolutionary Computation Conf. (GECCO-99) (1999) 229

[9] Gorges-Schleuter, M. *ASPARAGOS - An Asynchronous Parallel Genetic Optimization Strategy*. Proceedings of the 3rd International Conf. on Genetic Algorithms (1989) 422

[10] Grefenstette, J. *Optimization of Control Parameters for Genetic Algorithms*. IEEE Transactions on Systems, Man, and Cybernetics, Vol SMC-16, No.1, January/February 1986

[11] Manderick, B. and Spiessens, P. *Fine-Grained Parallel Genetic Algorithms*. Proceedings of the 3rd International Conference on Genetic Algorithms (1989) 428

[12] Muhlenbein, H., Schomisch, M., and Born, J. *The Parallel Genetic Algorithm as Function Optimizer*. Proc. of the 4th International Conf. on Genetic Algorithms (1991) 271

[13] Schaffer, J.D., Caruana, R., Eshelman, L., and Das, R. *A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization*. Proc. of the 3rd International Conf. on Genetic Algorithms (1989) 51

[14] Whitley, D. *Fundamental Principles of Deception in Genetic Search*. FOGA. Morgan Kaufmann (1991)

[15] Whitley, D. *Cellular Genetic Algorithms*. Proceedings of the 5th International Conference on Genetic Algorithms (1993) 658