Studying Parallel Evolutionary Algorithms: The Cellular Programming Case

Mathieu Capcarrère,[1] Andrea Tettamanzi,[2] Marco Tomassini,[3] and Moshe Sipper[1]

[1] Logic Systems Laboratory, Swiss Federal Institute of Technology, 1015 Lausanne, Switzerland.
E-mail: name.surname@di.epfl.ch, Web: lslwww.epfl.ch

[2] Department of Computer Science, University of Milano, Via Comelico 39/41, 20135 Milano, Italy.
E-mail: tettaman@dsi.unimi.it, Web: eolo.usr.dsi.unimi.it/~tettaman/.

[3] Institute of Computer Science, University of Lausanne, 1015 Lausanne, Switzerland.
E-mail: Marco.Tomassini@iismail.unil.ch, Web: www-iis.unil.ch.

Abstract. Parallel evolutionary algorithms, studied to some extent over the past few years, have proven empirically worthwhile—though there seems to be lacking a better understanding of their workings. In this paper we concentrate on cellular (fine-grained) models, presenting a number of statistical measures, both at the genotypic and phenotypic levels. We demonstrate the application and utility of these measures on a specific example, that of the cellular programming evolutionary algorithm, when used to evolve solutions to a hard problem in the cellular-automata domain, known as synchronization.

1 Introduction

Parallel evolutionary algorithms have been studied to some extent over the past few years. A basic tenet of such parallel algorithms is that the population has a spatial structure. A number of models based on this observation have been proposed, the two most important being the *island* model and the *grid* model. The coarse-grained island model features geographically separated subpopulations of relatively large size. Subpopulations exchange information by having some individuals migrate from one subpopulation to another with a given frequency and according to various migrational patterns. This can work to offset premature convergence, by periodically reinjecting diversity into otherwise converging subpopulations. In the fine-grained grid model individuals are placed on a toroidal d-dimensional grid (where $d = 1, 2, 3$ is used in practice), one individual per grid location (this location is often referred to as a *cell*, and hence the fine-grained approach is also known as *cellular*). Fitness evaluation is done simultaneously for all individuals, with genetic operators (selection, crossover, mutation) taking place locally within a small neighborhood. From an implementation point of view, coarse-grained island models, where the ratio of computation to communication is high, are more adapted to multiprocessor systems or workstation clusters, whereas fine-grained cellular models are better suited for massively parallel machines or specialized hardware. Hybrid models are also possible, e.g., one might consider an island model in which each island is structured as a grid of locally interacting individuals. For a recent review of parallel evolutionary algorithms (including several references) the reader is referred to [16].

Though such parallel models have empirically proven worthwhile [1, 4, 7, 8, 10, 15, 17], there seems to be lacking a better understanding of their workings. Gaining insight into the mechanisms of parallel evolutionary algorithms calls for the introduction of statistical measures of analysis. This is the underlying motivation of our paper. Specifically, concentrating on cellular models, our objectives are: (1) to introduce several statistical measures of interest, both at the genotypic and phenotypic levels, that are useful for analyzing the workings of fine-grained parallel evolutionary algorithms, and (2) to demonstrate the application and utility of these measures on a specific example, that of the cellular programming evolutionary algorithm [12]. Among the few theoretical works carried out to date, one can cite Mühlenbein [9], Cantú-Paz and Goldberg [2], and Rudolph and Sprave [11]. The latter treated a special case of fine-grained cellular algorithms, studying its convergence properties; however, they did not present statistical measures as done herein.

We begin in Section 2 by describing the cellular programming evolutionary algorithm and the synchronization task. Section 3 introduces basic formal definitions, and various statistical measures used in the analysis of cellular evolutionary algorithms. In Section 4, we apply the statistics of Section 3 to analyze the cellular programming algorithm when used to evolve solutions to the synchronization problem. Finally, we conclude in Section 5.

2 Evolving Cellular Automata

2.1 Cellular automata

Our evolving machines are based on the cellular automata model. Cellular automata (CA) are dynamical systems in which space and time are discrete. A cellular automaton consists of an array of cells, each of which can be in one of a finite number of possible states, updated synchronously in discrete time steps, according to a local, identical interaction rule. The state of a cell at the next time step is determined by the previous states of a surrounding neighborhood of cells. This transition is usually specified in the form of a *rule table*, delineating the cell's next state for each possible neighborhood configuration [12]. The cellular array (grid) is d-dimensional, where $d = 1, 2, 3$ is used in practice; in this paper we shall concentrate on $d = 1$. For such one-dimensional CAs, a cell is connected to r local neighbors (cells) on either side, where r is a parameter referred to as the radius (thus, each cell has $2r + 1$ neighbors, including itself).

The model investigated in this paper is an extension of the CA model, termed *non-uniform cellular automata* [12, 14]. Such automata function in the same way as uniform ones, the only difference being in the cellular rules that need not be identical for all cells. Our main focus is on the *evolution* of non-uniform CAs to perform computational tasks, using the cellular programming approach. Thus, rather than seek a *single* rule that must be universally applied to all cells in the grid, each cell is allowed to "choose" its own rule through evolution.

2.2 The cellular programming algorithm

A cell's rule table is encoded as a bit string (the "genome"), containing the next-state (output) bits for all possible neighborhood configurations. In our case, the CAs are of radius $r = 1$, and thus the genome consists of 8 bits: the bit at position 0 is the state to which neighborhood configuration 000 is mapped to and so on until bit 7 corresponding to neighborhood configuration 111. Rather than employ a population of evolving, uniform CAs, as with genetic algorithm approaches, our algorithm involves a single, non-uniform CA of size n, where the population of cell rules is initialized at random. Initial configurations are then generated at random, in accordance with the task at hand, and for each one the CA is run for M time steps. Each cell's fitness is accumulated over $C = 300$ initial configurations, After every C configurations evolution of rules occurs by applying crossover and mutation. This evolutionary process is performed in a completely *local* manner, where genetic operators are applied only between directly connected cells. It is driven by $nf_i(c)$, the number of fitter neighbors of cell i after C configurations. For a fuller description see [12, 13].

2.3 The synchronization task

The one-dimensional synchronization task was introduced by Das *et al.* [5] and studied by Hordijk [6], and Sipper [12, 13], the latter using non-uniform CAs. In this task the CA, given any initial configuration, must reach a final configuration, within M time steps, that oscillates between all 0s and all 1s on successive time steps. The synchronization task comprises a non-trivial computation for a small-radius CA.

3 Statistical Measures

3.1 Basic definitions and notation

In this section we formally define the basic elements used in this paper. A *population* is a collection of individuals, each represented by a genotype. A genotype is not necessarily unique—it may occur several times in the population. In addition, as the population considered has a topology, the spatial distribution of the genotypes is of interest. Let n be the number of individuals in the system. Let R_i, $1 \leq i \leq n$ be the genome of the ith individual. Let Γ be the space of genotypes and $G(\Gamma)$ be the space of all possible populations. Let $f(\gamma)$ be the fitness of an individual having genotype $\gamma \in \Gamma$. When the cells are arranged in a row, as is the case in the example of Section 2, a population can be defined as a vector of n genotypes $x = (R_1, \ldots, R_n)$; then we have $G(\Gamma) = \Gamma^n$.

For all populations $x \in G(\Gamma)$, an occupancy function $n_x\colon \Gamma \to N$ is defined, such that, for all $\gamma \in \Gamma$, $n_x(\gamma)$ is the number of individuals in x sharing the same genotype γ, i.e., the occupancy number of γ in x. The size of population x, $\|x\|$, is defined as $\|x\| \equiv \sum_{\gamma \in \Gamma} n_x(\gamma)$.

We can now define a share function $q_x\colon \Gamma \to [0,1]$ giving the fraction $q_x(\gamma)$ of individuals in x that have genotype γ, i.e., $q_x(\gamma) = n_x(\gamma)/\|x\|$.

Consider the probability space $(\Gamma, 2^\Gamma, \mu)$, where 2^Γ is the algebra of the parts of Γ and μ is any probability measure on Γ. Let us denote by $\tilde{\mu}$ the probability of generating a population $x \in G(\Gamma)$ by extracting n genotypes from Γ according to measure μ. It can be shown that it is sufficient to know either of the two measures—μ (over the genotypes) or $\tilde{\mu}$ (over the populations)—in order to reconstruct the other.

The fitness function establishes a morphism from genotypes into real numbers. If genotypes are distributed over Γ according to a given probability measure μ, then their fitness will be distributed over the reals according to a probability measure ϕ obtained from μ by applying the same morphism. This can be summarized by the following diagram:

$$\begin{array}{ccc} \Gamma & \xrightarrow{f} & \mathbb{R} \\ \wr & & \wr \\ \mu & & \phi \end{array} \tag{1}$$

The probability $\phi(v)$ of a given fitness value $v \in [0, +\infty)$ is defined as the probability that an individual extracted from Γ according to measure μ has fitness v (or, if we think of fitness values as a continuous space, the probability density of fitness v): for all $v \in [0, +\infty)$, $\phi(v) = \mu(f^{-1}(v))$, where $f^{-1}(v) \equiv \{\gamma \in \Gamma : f(\gamma) = v\}$.

An evolutionary algorithm can be regarded as a time-discrete stochastic process

$$\{X_t(\omega)\}_{t=0,1,2,\ldots}, \tag{2}$$

having the probability space $(\Omega, \mathcal{F}, P)$ as its base space, $(G(\Gamma), 2^{G(\Gamma)})$ as its state space, and the natural numbers as the set of times, here called *generations*. Ω might be thought of as the set of all the evolutionary trajectories, $\mathcal{F}$ is a σ-algebra on Ω, and P is a probability measure over $\mathcal{F}$.

The transition function of the evolutionary process, in turn based on the definition of the genetic operators, defines a sequence of probability measures over the generations.

Let $\tilde{\mu}_t$ denote the probability measure on the state space at time t; for all populations $x \in G(\Gamma)$,

$$\tilde{\mu}_t(x) = P\{\omega \in \Omega : X_t(\omega) = x\}. \tag{3}$$

In the same way, let μ_t denote the probability measure on space $(\Gamma, 2^\Gamma)$ at time t; for all $\gamma \in \Gamma$,

$$\mu_t(\gamma) = P[\kappa = \gamma | \kappa \in X_t(\omega)]. \tag{4}$$

Similarly, we define the sequence of probability functions $\phi_t(\cdot)$ as follows: for all $v \in [0, +\infty)$ and $t \in N$,

$$\phi_t(v) = \mu_t(f^{-1}(v)). \tag{5}$$

We shall now introduce several statistics pertaining to cellular evolutionary algorithms in the next two subsections: first, genotypic statistics, which embody aspects related to the genotypes of individuals in a population, and secondly phenotypic statistics, which concern properties of individual performance (fitness) for the problem at hand. Keeping in mind the synchronization problem studied herein, we concentrate on a one-dimensional spatial structure. We present a more complete set of measures as well as detailed proofs of the propositions given below in [3].

3.2 Genotypic statistics

One important class of statistics consists of various genotypic diversity indices (within the population) whose definitions are based on the occupancy and share functions delineated below.

Occupancy and share functions. At any time $t \in N$, for all $\gamma \in \mathit{\Gamma}$, $n_{X_t}(\gamma)$ is a discrete random variable with binomial distribution

$$P[n_{X_t}(\gamma) = k] = \binom{n}{k} \mu_t(\gamma)^k [1 - \mu_t(\gamma)]^{n-k}; \tag{6}$$

thus, $E[n_{X_t}(\gamma)] = n\mu_t(\gamma)$ and $\mathrm{Var}[n_{X_t}(\gamma)] = n\mu_t(\gamma)[1 - \mu_t(\gamma)]$. The share function $q_{X_t}(\gamma)$ is perhaps more interesting, because it is an estimator of the probability measure $\mu_t(\gamma)$; its mean and variance can be calculated from those of $n_{X_t}(\gamma)$, yielding

$$E[q_{X_t}(\gamma)] = \mu_t(\gamma) \quad \text{and} \quad \mathrm{Var}[q_{X_t}(\gamma)] = \frac{\mu_t(\gamma)[1 - \mu_t(\gamma)]}{n}. \tag{7}$$

Structure. Statistics in this category measure properties of the population structure, that is, how individuals are spatially distributed.

Frequency of transitions. The frequency of transitions $\nu(x)$ of a population x of n individuals (cells) is defined as the number of borders between homogeneous blocks of cells having the same genotype, divided by the number of distinct couples of adjacent cells. Another way of putting it is that $\nu(x)$ is the probability that two adjacent individuals (cells) have different genotypes, i.e., belong to two different blocks.

Formally, the frequency of transitions $\nu(x)$ for a one-dimensional grid structure can be expressed as

$$\nu(x) = \frac{1}{n} \sum_{i=1}^{n} \left[R_i \neq R_{(i \bmod n)+1} \right], \tag{8}$$

where $[P]$ denotes the indicator function of proposition P.

Diversity. There are a number of concievable ways to measure genotypic diversity, two of which we define below: population entropy, and the probability that two individuals in the population have different genotypes.

Entropy. The entropy of a population x of size n is defined as

$$H(x) = \sum_{\gamma \in \Gamma} q_x(\gamma) \log \frac{1}{q_x(\gamma)}. \tag{9}$$

Entropy takes on values in the interval $[0, \log n]$ and attains its maximum, $H(x) = \log n$, when x comprises n different genotypes.

Diversity indices. The probability that two individuals randomly chosen from x have different genotypes is denoted by $D(x)$.

Index $D(X_t)$ is an estimator of quantity

$$\sum_{\gamma \in \Gamma} \mu_t(\gamma) \left(1 - \mu_t(\gamma)\right) = 1 - \sum_{\gamma \in \Gamma} \mu_t(\gamma)^2, \tag{10}$$

which relates to the "breadth" of measure μ_t.

Proposition 1 *Let x be a population of n individuals with genotypes in Γ. Then,*

$$D(x) = \frac{n}{n-1} \sum_{\gamma \in \Gamma} q_x(\gamma)(1 - q_x(\gamma)). \tag{11}$$

Proof. See [3].

We observe that for all populations $x \in G(\Gamma)$,

$$D(x) \geq \frac{H(x)}{\log n}. \tag{12}$$

One can observe that $D(x)$ rises more steeply than entropy as diversity increases.

An interesting relationship between D and ν is given by the following proposition.

Proposition 2 *Given a random one-dimensional linear population x of size n, the expected frequency of transitions will be given by*

$$E[\nu(x)] = D(x). \tag{13}$$

Proof. See [3].

3.3 Phenotypic statistics

Phenotypic statistics deal with properties of phenotypes, which means, primarily, fitness. Associated with a population x of individuals, there is a fitness distribution. We will denote by ϕ_x its (discrete) probability function.

Performance. The performance of population x is defined as its average fitness, or the expected fitness of an individual randomly extracted from x, $E[\phi_x]$.

Diversity. The most straightforward measure of phenotypic diversity of a population x is the variance of its fitness distribution, $\sigma^2(x) = \mathrm{Var}[\phi_x]$.

Structure. Statistics in this category measure how fitness is spatially distributed across the individuals in a population.

Ruggedness. Ruggedness measures the dependency of an individual's fitness on its neighbors' fitness. For a one-dimensional population, x, of size n, $x \in \Gamma^n$, ruggedness can be defined as follows:

$$\rho^2(x) = \frac{1}{n} \sum_{i=1}^{n} \left[1 - \frac{1 + 2f(R_i)}{1 + f(R_{(i \bmod n)+1}) + f(R_{(i-2 \bmod n)+1})} \right]^2 . \qquad (14)$$

Notice that $\rho^2(x)$ is independent of the fitness magnitude in population x, i.e., of performance $E[\phi_x]$.

4 Results and Analysis

Using the different measures presented in the previous section we analyzed the processes taking place during the execution of the cellular programming algorithm presented in the Section 2. This was carried out for the synchronization task for CAs of size 150. The results are based on 75 runs. (Additional tasks are studied in [3].)

The evolutionary dynamics of the synchronization task were found to exhibit at most three fitness phases: a low-fitness phase, followed by a rapid-increase phase, ending with a high-fitness phase. Note that for this task a successful run is considered to be one where a perfect fitness value of 1.0 is attained. The evolutionary runs can be classified into four distinct classes, two of which represent successful runs (Figures 1a and 1b), the other two representing unsuccessful runs (Figures 1c and 1d). The classification is based on the number of phases exhibited during evolution. Let us now present some general trends, and then detailed results of our experiments according to these three fitness phases.

In all runs the entropy (H) falls from a high of approximately 0.8 to a low of approximately 0.7 within the first 20 generations, and from then on generally tends to decline. Though this decline is not monotonous the entropy always ends below 0.4. This fall in entropy is due to two factors. First, we can observe in all runs a steep drop in the transition frequency (ν) in the first few generations, followed by an almost continuous drop in the subsequent generations. Though it may be intuitive that, given the possibility of rule replication between neighboring cells after each generation, blocks will tend to form, our measures now provide us with quantitative evidence. Note that the transition frequency (ν)

progresses towards an oscillatory state about values below 0.3. The second factor involved in the lower entropy is the number of rules. One can see directly that a low ν implies few rules. This is corroborated by the diversity (D) measure decreasing trend.

For the the task studied herein the objective is to reach a high average fitness over the entire population, rather than consider just the highest-fitness individual cell. Thus, intuitively we can expect that the phenotypic variance will tend to be minimized, and we can factually check that both the fitness variance (σ^2) and ruggedness (ρ^2) are always very low towards the end of an evolutionary run. Usually the evolved CA had less than 10 different rules out of the 256 possible ones. We now detail the fitness phases.

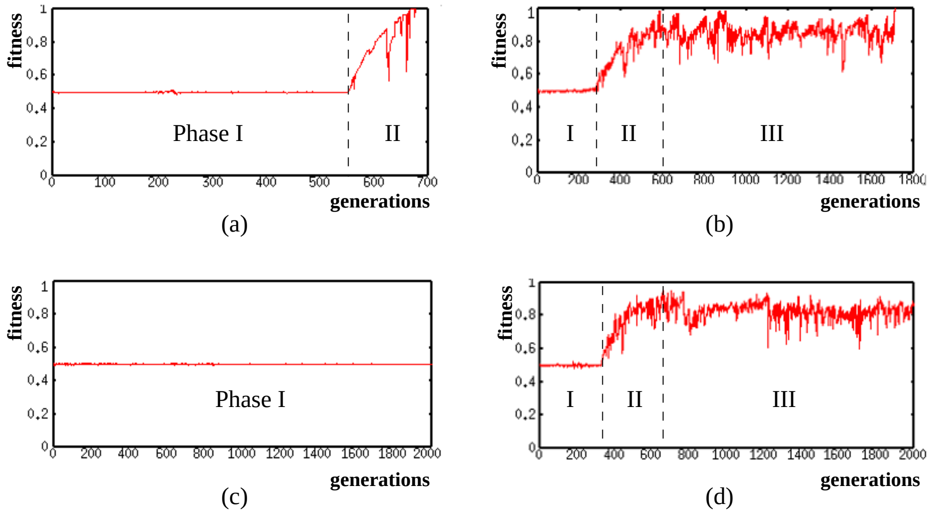

Fig. 1. The evolutionary runs for the synchronization task can be classified into four distinct classes, based on the three observed fitness phases: phase I (low fitness), phase II (rapid fitness increase), and phase III (high fitness). (a) Successful run, exhibiting but the first two phases. The solution is found at the end of phase II. (b) Successful run, exhibiting all three phases. The solution is found at the end of phase III. (c) Unsuccessful run, "stuck" in phase I. (d) Unsuccessful run, exhibiting all three phases. Phase III does not give rise to a perfect solution.

Phase I: Low fitness. This phase is characterized by an average fitness of 0.5, with an extremely low variance. However, while exhibiting phenotypic (fitness) "calmness," this phase is marked by high underlying genotypic activity: the entropy (H) steadily decreases, and the number of rules strongly diminishes. An unsuccessful type-c run (Figure 1c) results from "genotypic failure" in this phase. To explain this, let us first note that for the synchronization task, only rules with neighborhoods 111 mapped to 0 and 000 mapped to 1 may appear in a successful solution. Let us call this the "good" quadrant of the rule space, and define the "bad" quadrant to be the one that comprises rules mapping 111 to 1 and 000 to 0. In some experiments, evolution falls into the bad quadrant, possibly due to a low fitness variance. Only the mutation operator can possibly hoist the evolutionary process out of this trap. However, it is usually insufficient in itself, at least with the mutation rate used herein (0.001). Thus, in such a case the algorithm is stuck in a local minimum, and fitness never ascends beyond 0.53 (Figure 1c).

Phase II: Rapid fitness increase. A rapid increase of fitness characterizes this phase, its onset marked by the attainment of a 0.54 fitness value (at least). This comes about when a sufficiently large block of rules from the good quadrant emerges through the evolutionary process. In a relatively short time after this emergence (less than 100 generations), evolved rules over the entire grid all end up in the good quadrant of the rule space; this is coupled with a high fitness variance (σ^2). This variance then drops sharply, while the average fitness steadily increases, reaching a value of 0.8 at the end of this phase. Another characteristic of this phase is the sharp drop in entropy. On certain runs a perfect CA was found directly at the end of this stage, thus bringing the evolutionary process to a successful conclusion (Figure 1a).

Phase III: High fitness. The transition from phase II to phase III is not clear cut, but we observed that when a fitness of approximately 0.82 is reached, the fitness average then begins to oscillate between 0.65 and 0.99. During this phase the fitness variance also oscillates between approximately 0 and 0.3. While low, this variance is still higher than that of phase I. Whereas in phases I and II we observed a clear decreasing trend for entropy (H), in this phase entropy exhibits an oscillatory pattern between values of approximately 0.3 and 0.5. We conclude that when order (low entropy) is too high, disorder is reinjected into the evolutionary process, while remaining in the good quadrant of the rule space; hence the oscillatory behavior. On certain runs it took several hundred generations in this phase to evolve a perfect CA—this is a success of type b (Figure 1b). Finally, on other runs no perfect CA was found, though phase III was reached and very high fitness was attained. This is a type-d unsuccessful run (Figure 1d) which does not differ significantly from type-b successful runs.

5 Concluding Remarks

In this paper we introduced several statistical measures of interest, both at the genotypic and phenotypic levels, that are useful for analyzing the workings of fine-grained parallel evolutionary algorithms in general. We then demonstrated their application and utility on a specific example, that of the cellular programming evolutionary algorithm, which we employed to evolve solutions to the synchronization problem.

We observed the notable difference between activity at the genotypic level and at the phenotypic level, which we were able to study quantitatively. The synchronization task was seen to undergo (at most) three fitness phases, the nature of which (or the absence of which) served to distinguish between four types of evolutionary runs.

Parallel evolutionary algorithms have been receiving increased attention in recent years. Gaining a better understanding of their workings and of their underlying mechanisms thus presents an important research challenge. We hope that the work presented herein represents a small step in this direction.

References

1. D. Andre and J. R. Koza. Parallel genetic programming: A scalable implementation using the transputer network architecture. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming 2*, Cambridge, MA, 1996. The MIT Press.
2. E. Cantú-Paz and D. E. Goldberg. Modeling idealized bounding cases of parallel genetic algorithms. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 353–361, San Francisco, 1997. Morgan Kaufmann Publishers.
3. M. Capcarrère, A. Tettamanzi, M. Tomassini, and M. Sipper. A statiscal study of a class of cellular evolutionary algorithms. *Submitted*, 1998.
4. J. P. Cohoon, S. U. Hedge, W. N. Martin, and D. Richards. Punctuated equilibria: A parallel genetic algorithm. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, page 148. Lawrence Erlbaum Associates, 1987.
5. R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving globally synchronized cellular automata. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343, San Francisco, CA, 1995. Morgan Kaufmann.
6. Wim Hordijk. The structure of the synchonizing-ca landscape. Technical Report 96-10-078, Santa Fe Institute, Santa Fe, NM (USA), 1996.
7. A. Loraschi, A. Tettamanzi, M. Tomassini, and P. Verda. Distributed genetic algorithms with an application to portfolio selection problems. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 384–387. Springer-Verlag, New-York, 1995.
8. B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, page 428. Morgan Kaufmann, 1989.
9. H. Mühlenbein. Evolution in time and space–the parallel genetic algorithm. In Gregory J. E. Rawlins, editor, *Foundations Of Genetic Algorithms I*. Morgann Kaufmann Publishers, 1991.
10. M. Oussaidene, B. Chopard, O. Pictet, and M. Tomassini. Parallel genetic programming and its application to trading model induction. *Parallel Computing*, 23:1183–1198, 1997.
11. G. Rudolph and J. Sprave. A cellular genetic algorithm with self-adjusting acceptance threshold. In *First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 365–372, London, 1995. IEE.
12. M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag, Heidelberg, 1997.
13. M. Sipper. The evolution of parallel cellular machines: Toward evolware. *BioSystems*, 42:29–43, 1997.
14. M. Sipper. Computing with cellular automata: Three cases for nonuniformity. *Physical Review E*, 57(3), March 1998.
15. T. Starkweather, D. Whitley, and K. Mathias. Optimization using distributed genetic algorithms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, page 176, Heidelberg, 1991. Springer-Verlag.
16. A. Tettamanzi and M. Tomassini. Evolutionary algorithms and their applications. In D. Mange and M. Tomassini, editors, *Bio-Inspired Computing Machines: Toward Novel Computational Architectures*. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998. (to appear).
17. M. Tomassini. The parallel genetic cellular automata: Application to global function optimization. In R. F. Albrecht, C. R. Reeves, and N. C. Steele, editors, *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 385–391. Springer-Verlag, 1993.