A Cellular Genetic Algorithm with Self–Adjusting Acceptance Threshold

Günter Rudolph
Informatik Centrum Dortmund e.V
Joseph-von-Fraunhofer-Str. 20
D–44227 Dortmund
Rudolph@LS11.Informatik.Uni-Dortmund.de

Joachim Sprave
Universität Dortmund
Fachbereich Informatik XI
D–44221 Dortmund
Sprave@LS11.Informatik.Uni-Dortmund.de

Abstract

We present a genetic algorithm (GA) whose population possesses a spatial structure. The GA is formulated as a probabilistic cellular automaton: The individuals are distributed over a connected graph and the genetic operators are applied locally in some neighborhood of each individual. By adding a self–organizing acceptance threshold schedule to the proportionate reproduction operator we can prove that the algorithm converges to the global optimum. First results for a multiple knapsack problem indicate a significant improvement in convergence behavior. The algorithm can be mapped easily onto parallel computers.

1 Introduction

Evolutionary algorithms (EAs) form a class of stochastic optimization algorithms in which principles of organic evolution are regarded as rules for optimization. They are often applied to parameter optimization problems [1] when specialized techniques are not available or standard methods fail to give satisfactory answers due to multimodality, nondifferentiability or discontinuities of the problem under consideration. Here we focus on pseudoboolean optimization and a special class of EAs, namely genetic algorithms (GAs) [6]. Since GAs use bit strings to encode elements of the search space, they are natural candidates for pseudoboolean optimization.

In traditional realizations of GAs the population of individuals is just a multiset of feasible trial points in the search space. The exchange of information between two individuals by imitating the principle of inheritance may occur everywhere within the population, i.e., the population does not possess a spatial structure.

Recent experiences, however, reveal that GAs with a spatial population structure are not only easily to map onto massively parallel computers but also offer a better solution quality than traditional GAs [11, 5, 17, 12, 16]. Recently, this approach was also used for continuous search spaces in the framework of evolution strategies [13, 14, 18] as well as for hybrid parallel versions of evolutionary algorithms and simulated annealing [10, 14]. It was recognized several times that these fine–grained parallel algorithms may be regarded as cellular automata [17, 19, 21]. To provide a theoretical framework to study the differences we formally present a GA as a probabilistic cellular automaton, in which all genetic operators are applied locally in a certain neighborhood. This requires a modification of the proportionate reproduction or selection mechanism of traditional GAs.

Since proportionate reproduction prevents global convergence [15] we added a self–adjusting acceptance threshold which is related to the *Great Deluge algorithm* presented in [3]. This rising threshold ensures that the algorithm converges to the global optimum in finite expected time regardless of the objective function and the initialization of the algorithm.

The remainder of the paper is organized as follows: Section 2 presents a description of the cellular genetic algorithm as well as its formal model. The proof of convergence to the global optimum can be found in Section 3. First computational results are given in Section 4. Finally, we draw our conclusions in Section 5.

2 Cellular Genetic Algorithms

In principle, evolutionary algorithms can be designed to operate on arbitrary search spaces. But to facilitate theoretical considerations in later sections we restrict the search space to the finite binary space. Consequently, GAs are appropriate candidates to tackle the resulting pseudo-boolean optimization problems of the type

$$\max\{f(x) \,:\, x \in \mathbb{B}^l\} \quad , \tag{1}$$

where the real–valued function $f(.)$ denotes the *objective function* and $\mathbb{B}^l = \{0,1\}^l$ the *search space*. In general, a transformation of the objective function, the *fitness function* $F = g \circ f$, is used for the selection process in a GA.

2.1 Description of the Algorithm

A common feature of EAs is that they maintain a *population* of n *individuals*. In case of a GA each individual is represented by a binary vector $x \in \mathbb{B}^l$. The essential difference to traditional GAs is that each individual is regarded to 'live' on a node of a connected graph and that interactions between the individuals are restricted to their nearest neighbors in the graph. Clearly, the graph of a population of a traditional GA is fully connected. Here, the population is arranged on a ring, so that each individual has at least three neighbors: a left neighbor, a right neighbor and itself.

At each iteration (*generation*) of the algorithm all individuals are modified simultaneously by three genetic operators: *reproduction*, *crossover* and *mutation*. Since these operators are applied locally to the neighborhood of each individual, some of them have to be modified.

2.1.1 Local Reproduction

With regard to parallelism, the need of global knowledge should be kept small to allow efficient and scalable implementations. The most important global operator in traditional GAs is reproduction, because the sum of all fitness values $F(x_k)$, $k = 1, 2, \ldots, n$, in the population is used to calculate the *relative fitness* $p(x_k)$ of individual k. Assuming $f(x) > 0$ for all $x \in \mathbb{B}^l$, we may set $F(x) = f(x)$, so that the relative fitness of individual k is defined by

$$p(x_k) := \frac{F(x_k)}{\sum_{i=1}^{n} F(x_i)}$$

and the *cumulative relative fitness* CRF(x_k) as

$$\text{CRF}(x_k) := \sum_{i=1}^{k} p(x_i) \quad .$$

The reproduction probability of each individual is made proportionate to its relative fitness by picking a uniformly distributed random number $\xi \in [0,1)$ and choosing the individual number k with

$$\text{CRF}(x_k) = \min_{1 \le i \le n} \{\, \text{CRF}(x_i) : \text{CRF}(x_i) \ge \xi \,\}.$$

There are two major problems in proportionate reproduction: first, only positive objective function values are allowed. Second, when the population is near an optimum and relatively close together, all relative fitnesses are nearly the same. A common solution is the introduction of *windowing techniques* and *scaling*.

Windowing means that a history is tracked of the worst individuals' objective function value over a certain number of generations in the past, the so-called *evolution window*. The fitness now is defined as the objective function value reduced by the worst value from the history. This technique, however, does not solve the problem completely, because the current objective value may be worse than the worst value from the history, so that a negative relative fitness may occur. This can be avoided by restricting the evolution window to the current generation.

Proportionate reproduction even with windowing does not prefer good individuals enough to get satisfactory results in numerical optimization problems. Therefore, sometimes the selection pressure is increased by applying a scaling function to the evaluations, e.g. $F(x) = \exp(f(x))$.

In case of a neighborhood approach reproduction is slightly different: only individuals from the neighborhood of an individual can be selected as parents to produce its successor. Therefore, all calculations are restricted to the neighborhood.

Let $r \geq 1$ denote the neighborhood radius, i.e., the neighborhood size is $2r+1$. Then the fitness value of individual k at generation t is obtained by a normalized linear scaling[1]

$$F(x_k^{(t)}) := 1 + c \cdot \frac{f(x_k^{(t)}) - m_k^{(t)}}{M_k^{(t)} - m_k^{(t)}} \quad ,$$

where $c > 0$ and

$$m_k^{(t)} = \min\{\, f(x_i^{(t)}) \,:\, i = k-r, \ldots, k+r \,\}$$

and

$$M_k^{(t)} = \max\{\, f(x_i^{(t)}) \,:\, i = k-r, \ldots, k+r \,\}.$$

Here and in the sequel all index calculations are performed modulo n. The local CRF of individual k can now be defined as

$$\mathrm{CRF}_r(x_k) := \sum_{i=k-r}^{k+r} p_r(x_i) \quad ,$$

where

$$p_r(x_k) := \frac{F(x_k)}{\sum_{i=k-r}^{k+r} F(x_i)} \quad . \qquad (2)$$

2.1.2 Crossover and Mutation

The crossover and mutation operator do not need a modification. Crossover operates on two individuals (parents), that are selected from the neighborhood, by picking a position $\chi \in \{\, 1, 2, \ldots, l-1 \,\}$ at random and creating a new individual by taking the first χ bits from the first parent and the remaining bits from the second.

Each bit position of the new created individual is mutated by inverting the actual bit position, if the fixed mutation probability $p_m \in (0,1)$ is larger than a random number uniformly distributed over $[\,0,1)$, which is drawn anew for each bit position.

[1] We define $0/0 := 0$.

2.1.3 Threshold acceptance

Since the current generation also belongs to the evolution window, there is no lower boundary for the quality of new offspring, so if you don't use an elitist strategy the algorithm will not be globally convergent [15].

To achieve global convergence a threshold technique is introduced in this GA: each individual has to be better than the *tidal value*, which is defined as the maximum of the worst evaluation a certain number in the past and the tidal value of the last generation. As one can see, the tidal value is monotone rising by its definition. To keep the population size constant, for each place in the population the generation of offspring is iterated until either the generated offspring is above the current tide or a certain number of unsuccessful trials is exceeded. In this case the individual in this place remains unchanged.

Again, in order to avoid a global knowledge, a tidal value exists for each place in the population, as well as a history of the evaluations in the past. Let be t the generation number, k the position in the population, δ the window size. Then the tide of individual k at generation t is defined as

$$\tau_k^{(t)} := \begin{cases} f(x_k^{(0)}) & \text{, if } t < \delta \\ \max\{\tau_k^{(t-1)}, f(x_k^{(t-\delta)})\} & \text{, else.} \end{cases} \qquad (3)$$

2.1.4 Outline of the Algorithm

The following pseudo code gives a sketch of the algorithm:

```
Initialize
REPEAT
  FOR EACH node
    Select two neighbors
    Recombine them
    Mutate resulting offspring
    Evaluate offspring
```

```
    IF F(offspring) > threshold
    THEN
       accept offspring
    ENDIF
    Update local threshold
  ENDFOR
UNTIL max. number of generations
```

Because all offspring is generated simultaneously, an implementation of must manage a second population to store the accepted offspring. The above outline shows a sequential implementation, but the body of the FOR-loop can be evaluated in parallel, with a synchronization point at the start of the FOR-loop.

2.2 Modeling the cellular genetic algorithm

Locally interacting systems can be studied in the general framework of probabilistic automata networks (PAN). Special cases of PANs are cellular automata, neural networks [4] and, as we shall demonstrate, locally interacting evolutionary algorithms. The following definition is extracted from [20]:

DEFINITION 1:
Let V denote the set of nodes of an undirected graph $G = (V, E)$ with edges $E \subseteq V \times V$. Each node $v \in V$ is called an *automaton*, which possesses a finite *state space* $\mathcal{S}_v$. The product space $\mathcal{S} = \bigotimes_{v \in V} \mathcal{S}_v$ is called the *system state space*. Each $s \in \mathcal{S}$ denotes a *system state*, whereas s_v denotes the *state of automaton* v. The set $\mathcal{N}_a = \{v \in V : (v, a) \in E\}$ is called the neighborhood of automaton a. For each automaton a there exists a well–defined *transition matrix* P_a of size $\prod_{v \in \mathcal{N}_a} |\mathcal{S}_v| \times |\mathcal{S}_a|$, that gathers the probabilities that state s_a with neighborhood $\mathcal{N}_a$ transitions to a state $s'_a \in \mathcal{S}_a$. The new system state $s(t+1)$ at step $t+1$ depends on the previous system state $s(t)$ and the transition matrices $P_v, v \in V$:

$$s(t+1) = g(s(t), (P_v : v \in V)) ,$$

where $g(.)$ symbolizes the synchronous *update rule*. A *probabilistic automata network* is completely determined by the tuple $(V, (\mathcal{S}_v : v \in V), (\mathcal{N}_v : v \in V), (P_v : v \in V), s(0))$. •

The algorithm presented in this paper is a special case of a PAN and may be viewed as a probabilistic cellular automaton. The graph of a PCA is more "regular" than the graph of a PAN, which we like to express as follows:

DEFINITION 2:
Let $G = (V, E)$ be a graph with $V \subseteq \mathbb{Z}^d, d \in \mathbb{N}$. The neighborhood $\mathcal{N}_v$ of vertex v is determined by the *neighborhood structure* $\mathcal{N}^s \subset \mathbb{Z}^d$, which is a finite set of offsets:

$$\mathcal{N}_v = v + \mathcal{N}^s = \{v + a : a \in \mathcal{N}^s\} .$$

The cardinality of $\mathcal{N}^s$ is called the *neighborhood size*. If $V = \bigotimes_{i=1}^d \mathbb{Z}/\mathbb{Z}_{m_i}, m_i \in \mathbb{N}$, then the offsets are added by modulo arithmetic. •

Wolfram [22, p. 1] summarized the basic characteristics of cellular automata. We differ from his definition only by allowing probabilistic update rules:

DEFINITION 3:
A probabilistic automata network with a graph as in Definition 2, where each automaton possesses the same neighborhood structure $\mathcal{N}^s$, the same state space $\mathcal{S}_0$ and the same transition matrix P is called a *probabilistic cellular automaton* (PCA). A PCA is completely determined by the tuple $(V, \mathcal{S}_0, \mathcal{N}^s, P, s(0))$. •

Now we can formulate the cellular genetic algorithm in the framework of cellular automata: The graph of the network is defined by the set of nodes $V = \mathbb{Z}/\mathbb{Z}_n$ and the neighborhood structure $\mathcal{N}^s = \{-r, \ldots, -1, 0, 1, \ldots, r\}$, where $r > 0$ denotes the *neighborhood radius*. The state space of each individual is $\mathcal{S} = \mathbb{B}^{(2r+1)\cdot l} \times \mathcal{H} \times \mathcal{T}$, where $\mathcal{T} = \{f(x) : x \in \mathbb{B}^l\}$ denotes the set of possible threshold values and $\mathcal{H} = \mathcal{T}^\delta$ denotes all possible histories of size δ. Since $|\mathcal{T}| \leq 2^l$ the state space is finite.

The transition matrix P can be decomposed into a probabilistic and a deterministic part. The probabilistic part describes the probabilities to generate an intermediate individual $y_k^{(t+1)} \in \mathbb{B}^l$ from the neighborhood $(x_{k-r}^{(t)}, \ldots, x_k^{(t)}, \ldots x_{k+r}^{(t)}) \in \mathbb{B}^{(2r+1)\cdot l}$. Let transition matrix $G : 2^{(2r+1)\cdot l} \times 2^l$ gather these probabilities. Matrix G may be decomposed into the product $G = S \cdot C \cdot M$ of transition matrices, where $S : 2^{(2r+1)\cdot l} \times 2^{2l}$ contains the probabilities to select two specific individuals from the neighborhood, $C : 2^{2l} \times 2^l$ contains the probabilities for the outcome of crossover and $M : 2^l \times 2^l$ contains the probabilities for the outcome of mutation. Clearly, matrices S, C and M are stochastic matrices, i.e., each entry is non-negative and the entries in each row sum up to one.

The probability to generate a bit string $x' \in \mathbb{B}^l$ from $x \in \mathbb{B}^l$ by mutation is

$$P\{x \to x'\} = p_m^{H(x,x')}(1-p_m)^{l-H(x,x')} > 0$$

if $p_m \in (0,1)$ and where $H(x,x')$ denotes the Hamming distance between string x and x'. Therefore, all entries in matrix M are positive. This leads to:

LEMMA 1:
All entries of transition matrix $G = S \cdot C \cdot M$ are positive.

PROOF:
Since matrix multiplication is associative we first consider the product $C \cdot M$. As previously mentioned, the nonnegative entries of each row of matrix C sum up to one. Therefore, there is at least one positive entry in each row. Since matrix M is positive, multiplication of a row of C with a column of M gives a positive value. Consequently, the product $C \cdot M$ is positive. The same argumentation can be applied to the product $G = S \cdot (C \cdot M)$. •

Although it is possible to derive formulas for the entries of the transition matrices S and C, we omit these here, because these expressions are not necessary for our global convergence proof in the next section.

It remains to model the deterministic part of the update rule. Let $(h_{k,1}^{(t)}, h_{k,2}^{(t)}, \ldots, h_{k,\delta}^{(t)})$ be the history and $\tau_k^{(t)}$ be the threshold value of individual k at step t. Then

$$(h_{k,1}^{(t+1)}, h_{k,2}^{(t+1)}, \ldots, h_{k,\delta}^{(t+1)}) = (f(x_k^{(t)}), h_{k,1}^{(t)}, \ldots, h_{k,\delta-1}^{(t)})$$

and

$$\tau_k^{(t+1)} = \begin{cases} \max\{\tau_k^{(t)}, h_{k,\delta}^{(t+1)}\} & \text{, if } t \geq \delta \\ f(x_k^{(0)}) & \text{, otherwise.} \end{cases}$$

3 Global Convergence Proof

At first, we have to define a criterion to decide whether the cellular genetic algorithm converges to the global optimum. Clearly, the best objective function value within a population should converge to the maximum value. Since the best objective function value is a random variable, it is useful to distinguish between the different modes of convergence of random sequences:

DEFINITION 4:
If $\{ D_t, t \geq 0 \}$ are random variables on a probability space $(\Omega, \mathcal{A}, P)$, then the random sequence $(D_t)_{t\geq 1}$ is said to
(a) *converge in probability* to D_0, denoted $D_t \xrightarrow{P} D_0$, if $\forall \epsilon > 0$

$$\lim_{t\to\infty} P\{ |D_t - D_0| \leq \epsilon \} = 1;$$

(b) *converge almost surely* to D_0, denoted $D_t \xrightarrow{a.s.} D_0$, if

$$P\{ \lim_{t\to\infty} D_t = D_0 \} = 1 \ ;$$

(c) *converge completely* to D_0, denoted $D_t \xrightarrow{c} D_0$, if $\forall \epsilon > 0$

$$\sum_{t=1}^{\infty} P\{ |D_t - D_0| > \epsilon \} < \infty \ . \quad \bullet$$

The following Lemma 2 summarizes some dependencies between the different modes of stochastic convergence.

LEMMA 2:
(a) $D_t \xrightarrow{c} D_0 \Rightarrow D_t \xrightarrow{a.s.} D_0 \Rightarrow D_t \xrightarrow{P} D_0$. The converse is not true in general.
(b) If Ω countable, then $D_t \xrightarrow{P} D_0 \Leftrightarrow D_t \xrightarrow{a.s.} D_0$.

PROOF: (a) [9]; (b) [2]. •

Now we can state the criterion for global convergence:

DEFINITION 5:
Let $f^* = \max\{ f(x) : x \in \mathbb{B}^l \}$ and $D_t = f^* - \max\{ f(x_k^{(t)}) : k = 0, 1, \ldots, n-1 \}$. The cellular genetic algorithms *converges to the global optimum* if $D_t \xrightarrow{P} 0$. •

We note that convergence in probability should be the minimum requirement. A Markov chain analysis leads to the following result:

THEOREM 1:
The cellular genetic algorithm converges to the global optimum regardless of the initialization. Even more: $D_t \xrightarrow{c} 0$.

PROOF:
It is sufficient to show that $f(x_k^{(t)}) \xrightarrow{c} f^*$ for any $k = 0, 1, \ldots, n-1$. We therefore choose one specific k and omit the subscript in the sequel.

Since we are only interested in the objective function value, we may condense the state space of each individual to $\mathcal{T}^{\delta+2}$, so that we have to investigate the behavior of the random sequence $(f(x^{(t)}), \tau^{(t)}, h_1^{(t)}, h_2^{(t)}, \ldots, h_\delta^{(t)})$.

We shall demonstrate that the above sequence may be described as a homogeneous absorbing Markov chain, whose only absorbing state is $(f^*, f^*, f^*, \ldots, f^*)$. In this case it is known (see e.g. [7]) that the probability to transition to the absorbing state at step t can be bounded from below by $1 - C \cdot \beta^t$, where $C > 0$ and $\beta \in (0,1)$. That means, that $P\{f(x^{(t)}) - f^* > \epsilon\} \le C \cdot \beta^t$. Thus,

$$\sum_{t \ge 0} P\{f(x^{(t)}) - f^* > \epsilon\} \le C \cdot \sum_{t \ge 0} \beta^t < \infty \; .$$

Consequently, we get $D_t \xrightarrow{c} 0$ and the proof is completed.

It remains to prove that the above sequence forms a homogeneous absorbing Markov chain. First, let us define some subsets of the state space $\mathcal{T}^{\delta+2}$:

$$\begin{aligned}
A_1 &= \{(a, b, *, *, \ldots, *) \;:\; a, b < f^*\}\,, \\
A_2 &= \{(f^*, c, *, *, \ldots, *) \;:\; c \ge b\}\,, \\
A_3 &= \{(*, d, f^*, *, \ldots, *), : \; c \le d \le f^*\}\,, \\
A_4 &= \{(*, f^*, *, *, \ldots, *)\}\,, \\
A_5 &= \{(f^*, f^*, *, *, \ldots, *)\}\,, \\
A_6 &= \{(f^*, f^*, f^*, f^*, \ldots, f^*)\}\,,
\end{aligned}$$

where the symbol $*$ is used to denote any value from the set $\mathcal{T}$. Second, note that it follows from Lemma 1 that there exists a minimal positive probability p_{min} to generate any state in one step. This probability is just $p_{min} = \min\{g_{ij}\} > 0$, where g_{ij} denote the entries of the transition matrix G in Lemma 1.

Now assume that the chain at step[2] $t \ge \delta$ is in a state that belongs to A_1. Either the chain remains in class A_1 or it transitions to A_2 with a probability of at least $p_{min} > 0$. If the latter occurs then the next state belongs to the set A_3, because the optimal value f^* is saved in the history. After at most δ steps the optimal value f^* becomes the threshold value, so that the chain reaches a state in A_4. Now the chain either remains in A_4 or it transitions to A_5 with a probability not less than $p_{min} > 0$. From now on individuals with lower objective function value than the optimal one (below the threshold) are not accepted. Therefore, the chain remains in the set A_5. After at most δ steps all history values become f^* and the chain has reached its only absorbing point. This always happens regardless of the initialization of the algorithm. Consequently, the Markov chain is absorbing and the proof is completed. •

4 First Computational Results

Just to achieve a first assessment of the behavior of the *Great Deluge GA* (GDGA), experiments were run on a NP–hard multiple knapsack problem with varying neighborhood and window size. The problem can be formalized as follows:

$$\begin{aligned} f(x) = c^T x \quad &\to \max \\ \text{s.t. } Ax \le b \end{aligned}$$

with $x \in \mathbb{B}^l$, $c \in \mathbb{R}_+^l$, $b \in \mathbb{R}_+^m$ and $A \in \mathbb{R}_+^{m,l}$. The constraints were included into the objective function by a penalty technique in the same manner as in [8]:

$$f(x) = c^T x - v \cdot c_{max} \quad \to \max \,,$$

where v denotes the number of violated constraints and c_{max} the largest entry in the cost vector c.

Here, the problem had dimension $l = 50$ and $m = 5$ constraints. Figure 1 summarizes the

[2] Since the threshold adjustment rule (3) depends on the time parameter for the first δ steps, the Markov chain is not homogeneous for the first steps. But we are interested in asymptotic results, so we regard the first steps as a mechanism to generate the initial distribution and let the chain start at step δ.

success frequency of the GDGA with 500 individuals after at most 500 generations for neighborhood sizes varying from 1 to 200 and window sizes between 1 and 500. The mutation probability was $p_m = 1/l$ and the crossover probability for one point crossover was $p_c = 1$ for all runs. For each setting 200 runs were made resulting in more than 50,000 runs in total.

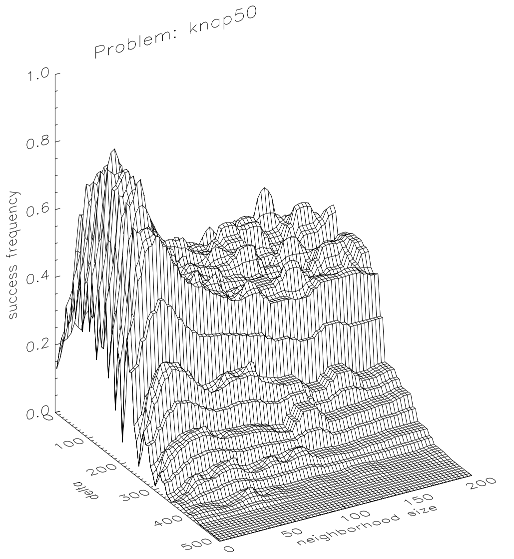

Fig. 1: Success frequency of GDGA for a multiple knapsack problem of dimension 50 with 5 constraints for varying neighborhood size and window size δ.

The above figure indicates that this problem was solved significantly more often with a neighborhood size at about 20 and and a window size at about 100 than with other parameter settings. The GDGA with neighborhood size 200 and window size $\delta = 500$ should behave similar to a traditional GA, because reproduction then considers almost the entire population and the threshold rule is switched off effectively.

For safety experiments were run with a traditional GA. The results were obtained by using the software package *Genesis 5.0* that uses two–point–crossover and stochastic universal selection by default.

The tests were conducted with mutation probabilities $p_m \in \{.001, .005, .010, .015, .200\}$ and crossover probabilities $p_c \in \{.6, 1.0\}$, using 200 runs per parameter setting. None of the runs succeeded in finding the optimum within 500 generations – as expected.

5 Conclusions

The GA with spatially local interactions and self-adapting acceptance threshold shows a significant better convergence behavior than its traditional counterpart with global (panmictic) interactions and without acceptance threshold, if the parameters are set appropriate. Although the tests were run for one problem instance only, we expect similar results for other problem instances. The relatively large neighborhood radius necessary to produce good results is a little bit counter–intuitive and disappointing with respect to the suitability of the approach for SIMD parallel computers. A possible route to reduce the communication effort might rely on other spatial topologies.

Finally, different selection operators like ranking etc. could produce a different convergence behavior with smaller optimal neighborhood radii. These investigations, the test of other problem instances and the understanding of the search dynamics remain for future work.

References

[1] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[2] Y.S. Chow and H. Teicher. *Probability Theory*. Springer, New York, 1978.

[3] G. Dueck, T. Scheuer, and H.-M. Wallmeier. Toleranzschwelle und Sintflut: neue Ideen zur Optimierung. *Spektrum der Wissenschaft*, pages 165–171, March 1993.

[4] E. Goles and S. Martinez. *Neural and automata networks*. Kluwer, Dordrecht, 1990.

[5] M. Gorges-Schleuter. ASPARAGOS: an asynchronous parallel genetic optimization strategy. In J.D. Schaffer, editor, *Genetic Algorithms, Proceedings of the 3rd*

International Conference on Genetic Algorithms, pages 422–427. Morgan Kaufman, San Mateo, 1989.

[6] J.H. Holland. *Adaptation in natural and artificial systems*. The University of Michigan Press, Ann Arbor, 1975.

[7] M. Iosifescu. *Finite Markov Processes and Their Applications*. Wiley, Chichester, 1980.

[8] S. Khuri, Th. Bäck, and J. Heitkötter. The zero/one multiple knapsack problem and genetic algorithms. In E. Deaton, D. Oppenheim, J. Urban, and H. Berghel, editors, *Proceedings of the 1994 ACM Symposium on Applied Computing*, pages 188–193. ACM Press, New York, 1994.

[9] E. Lukacs. *Stochastic Convergence*. Academic Press, New York, 2nd edition, 1975.

[10] S.W. Mahfoud and D.E. Goldberg. A genetic algorithm for parallel simulated annealing. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 301–310. North Holland, Amsterdam, 1992.

[11] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer. Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7:65–88, 1988.

[12] M.E. Palmer and S.J. Smith. Improved evolutionary optimization of difficult landscapes: Control of premature convergence through scheduled sharing. *Complex Systems*, 5:443–458, 1991.

[13] G. Rudolph. Parallel approaches to stochastic global optimization. In W. Joosen and E. Milgrom, editors, *Parallel Computing: From Theory to Sound Practice, Proceedings of the European Workshop on Parallel Computing (EWPC 92)*, pages 256–267. IOS Press, Amsterdam, 1992.

[14] G. Rudolph. Massively parallel simulated annealing and its relation to evolutionary algorithms. *Evolutionary Computation*, 1(4):361–382, 1993.

[15] G. Rudolph. Convergence properties of canonical genetic algorithms. *IEEE Transaction on Neural Networks*, 5(1):96–101, 1994.

[16] P. Spiessens and B. Manderick. A massively parallel genetic algorithm: Implementation and first analysis. In R.K. Belew and L.B. Booker, editors, *Proceedings of the fourth Conference on Genetic Algorithms*, pages 279–286. Morgan Kaufmann, San Mateo, 1991.

[17] J. Sprave. Parallelisierung Genetischer Algorithmen zur Suche und Optimierung. Diploma thesis, University of Dortmund, Department of Computer Science, December 1990.

[18] J. Sprave. Linear neighborhood evolution strategies. In A.V. Sebald and L.J. Fogel, editors, *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 42–51. World Scientific, River Edge (NJ), 1994.

[19] M. Tomassini. The parallel genetic cellular automata: Application to global function optimization. In R.F. Albrecht, C.R. Reeves, and N.C. Steele, editors, *Artificial Neural Nets and Genetic Algorithms, Proceedings of the International Conference in Innsbruck, Austria*, pages 385–391. Springer, Wien, 1993.

[20] A.L. Toom et al. Discrete local Markov systems. In R.L. Dobrushin, V.I. Kryukov, and A.L. Toom, editors, *Stochastic cellular systems: ergodicity, memory, morphogenesis*, pages 1–182. Manchester University Press, Manchester, 1990.

[21] D. Whitley. Cellular genetic algorithms. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, page 658. Morgan Kaufman, San Manteo (CA), 1993.

[22] S. Wolfram, editor. *Theory and Applications of Cellular Automata*. World Scientific, Singapore, 1986.