Evolution of Fault-Tolerant Self-replicating Structures

Ludovic Righetti[1], Solaiman Shokur[1] and Mathieu S. Capcarrere[2,3]

[1] Logic Systems Laboratory
Lausanne Swiss Federal Institute of Technology, CH-1015 Lausanne
Name.Surname@epfl.ch, http://lslwww.epfl.ch
[2] Computer Science Institute
University of Lausanne, CH-1015 Lausanne
[3] Computing Laboratory
University of Kent, Canterbury CT2 7NF
mathieu@capcarrere.org, http://www.cs.ukc.ac.uk

Abstract. Designed and evolved self-replicating structures in cellular automata have been extensively studied in the past as models of Artificial Life. However, CAs, unlike their biological counterpart, are very brittle: any faulty cell usually leads to the complete destruction of any emerging structures, let alone self-replicating structures. A way to design fault-tolerant structures based on error-correcting-code has been presented recently[1], but it required a cumbersome work to be put into practice. In this paper, we get back to the original inspiration for these works, nature, and propose a way to *evolve* self-replicating structures, faults here being only an idiosyncracy of the environment.

1 Introduction

Self-replication is at the very core of the cellular automata inception. The original study of von Neumann that lead to the development of CA was it not centered around the question of self-reproduction[4] ? This birthmark lead to many studies of self-replication by means of CA, some famous landmarks being Langton's loop [6], Tempesti's loop [14] or the simplest self-replicating loop to date, Chou and Reggia's [12].

Self-replicating structures in cellular automata have thus generated a large quantity of papers [13]. However, most of these have been devoted to the study of the self-replicating process in itself, as a model of one of life primary properties. The underlying motivation of such researches could be expressed as a question: What minimal information processing is required for self-replication to happen?

[4] Barry McMullin in [10] argued very interestingly that in fact self-reproduction is a paraphernalia rather than the center of von Neumann's work. We would not quite disagree with him but think that this "means" of getting complexity has turned out to be an interesting problem in itself, as demonstrated by the large subsequent literature.

Simplification was thus a prime constraint in all these studies, unfortunately, not only as a guide to extract the quintessential ideas behind self-replication, but also for obvious practical reasons. We believe that for the latter reasons rather than the former principle, an always present quality in natural systems was omitted: *robustness*.

The usual model of cellular automata is discrete in time and space and all cells update synchronously according to a deterministic rule. No fault ever occurs in the transition rule nor on the cell current state. Nevertheless if we are to look at real biology, it appears at all levels that either insensitiveness to faults or self-repair is a prime condition of survival, the environment being extremely noisy and uncertain. Most fault resistant cellular automata designed in the past[3–5] are based on a hierarchical constructions, i.e., an alteration of the inner architecture of the CA. In this paper, we rather propose to develop fault-tolerant transition rules, thereby keeping the classical CA struture unchanged. We evolve CA that are fault-tolerant in their functioning itself, at a minimal cost in terms of space, time or memory.

In section 2 of this paper, we present the main previous work on the evolution of self-replicating structures which served as the base for this work. In section 3, we then accurately define what we mean by a faulty environment. Section 4, presents our evolutionary model, notably detailing the algorithm and parameters we used to successfully evolve fault-tolerant self-replicating objects. Finally in section 5, we show the results we obtained with a straightforward approach and then present refinements that lead to the evolution of (relatively) very robust self-replicating structure. We conclude in section 6.

2 Previous Work on the Evolution of Self-replicating Structures

If *hand-designed* self-replication in CA has attracted its chunk of interest, very few have tackled *the evolution* of self-replication, a more arduous but also a more "natural" approach[5]. The main works on the evolution of self-replication were lead by Jason Lohn and James Reggia [8]. Let us now briefly present the framework of this previous study thereby introducing ours as it was used as the general grounds for our work.

2.1 Framework of Previous Evolution of Self-replication

The three important aspects in an evolutionary algorithm are: the encoding, the fitness function, and finally the structure of the algorithm itself.

The encoding of the transition rules of the CA is rather straightforward. As von Neumann neighborhood was used, the transition rule is a function $t : q^5 \rightarrow q$, where q is the set of all possible states of each cell. If an order on q^5 is defined, then a string of symbols of q of length q^5 is enough to characterize a transition

[5] Beyond the scope of CAs, there are some very interesting studies of evolution of self-replication, to cite but one, Pargellis [11]. Talking of this work, we should underline here that we evolve worlds where robust self-replication emerges, rather than setting-up a world where self-replication evolves.

rule. In other words, as the set of all possible neighborhoods is ordered, the first symbol of this string is the future state matching the first neighborhood, the second symbol is the future state matching the second neighborhood, and so on and so forth. Then the chromosome is just that characterizing string.

The fitness function is the core of the evolutionary algorithm. In the original population no candidate solution is even close to self-replication, the quality of a fitness function will thus lie in its ability to detect elements with great potentials. Lohn's fitness function is decomposed into three parts F_g, F_p, and F_r: $-$ F_g measures the tendency to grow for each element (each state). It is bounded between 0 and 1 and is equal to 0 if all elements disappear and to 1 if all elements present in the seed configuration increases in numbers.– F_p measures the quality of the relative position of each element. This is the most important fitness bit as it will make the difference between chaotic growth and self-replication. It is bounded between 0 and 1, 0 meaning that the neighborhood of an element has no connection to its neighborhood at the previous time step and 1 meaning the exact opposite. Replication would imply high values for both F_g and F_p but not necessarily vice-versa. – F_r, hence, measures directly the number of replicants. More precisely it is a function of the number of replicants. It is also bounded between 0 and 1. For each evaluation the CA is initialized with a seeding configuration and is evaluated on T time steps. Finally the three fitness components are weighted out against each other. Lohn used $F = 0.05F_g + 0.75F_p + 0.2F_r$. As we will see these weights are of prime importance.

The algorithm itself is a simple straightforward genetic algorithm using a fitness proportionate selection, a cross-over rate of 0.8 and a mutation rate of 0.1. The only specific operator is the crossover operator which is multi-point: the order of the characterizing string is such that the states of the center cell of the first q^4 neighborhoods matching the first q^4 characters are identical, the same for the following q^4 characters, and so on and so forth; simply a crossover point is chosen for each q^4 long chunk, for each "gene"; the crossover happens in parallel on all genes.

2.2 Previous Results

The complexity of the task depends on the size of the structure that should replicate. This is determined by the configuration used as the seed on which the evolution takes place. The results published by Lohn and Reggia [7, 8, 12] show a success rate of 93% for structures of size 2 (linear), 22% for structures of size 3 (right angle) and only 2% for structures of size 4. A run is succesful if the CA replicates at least once. We can note that the latter result is statistically not significant and we will thus concentrate on 3 element structures.

3 A Non-deterministic Cellular Automata Space

The idea of fault-tolerance is rather vague. We propose here to delimit the scope of validity of our work.

Formally cellular automata are d-dimensional discrete space, in which each point, called a cell, can take a finite number of state q. Eevery cells update

synchronously its own state according to the states of a fixed neighborhood of size n following a deterministic rule. In CAs, faults may basically occur at two levels: the synchronization and the cell transition rule. We do not cater specifically for the former problem in this paper as the latter encompasses it. Though as demonstrated in [2], limiting ourself to the former would have been simpler as it is possible to simulate any q-state synchronous CA with a $3.q^2$ state asynchronous CA. As for the latter problem, which could be further divided into reading and writing errors, we model it as a non-deterministic rule. More precisely, there is a probability of faults p_f, such that any given cell will follow the transition rule with probability $(1 - p_f)$, and take any state with probability p_f. This model, though apparently catering only for writing errors, does simulate reading errors[6]. One may object that we did not mention the major fault problems when a cell simply stops functioning at all. This is not our purpose to treat that kind of 'hardware' problem here, but we may say that our model could be implemented over an embryonics tissue [9] which deals with this kind of malfunction. Besides, one may note that such a permanent failure is not, in itself, fatal to our system, but only weakens all intersecting neighborhoods. As we have shown in [1] this noisy environment is more than enough to irrevocably disturb and destroy any of the most famous self-replicating structures.

In the scope of the experiments presented in this paper, we evolve a perfectly standard CA which works in a faulty environment. However, unlike in our preceding paper [1], cells which are not updated, i.e. cells which are quiescent surrounded only by quiescent cells, are *not* subject to faults. Matter is not created out of nothing.

4 The Evolutionary Framework

We will now briefly overview the parameters and the algorithm we used, thereby allowing for future replication of our results.

The algorithm used is a straightforward classical Genetic Algorithm. Our initial population size for all experiments was 200 and the mutation and crossover rate were respectively 0.1% and 80%.

The selection method chosen is a peculiar version of rank selection. Every individuals are sorted, the worst ranked first and the best ranked last. Then the probability of choosing an individual with rank i for reproduction in a population of size N is $\frac{i}{\sum_{k=1}^{N} k}$.

The encoding is exactly the same as for Lohn's work. We use the set of states q to be $\{0, 1, 2, 3\}$, where 0 is the quiescent state. However there is a subtlety in the sense that each state, except the quiescent state, has a direction component which may be one of the four directions $\leftarrow, \uparrow, \rightarrow, \downarrow$. This direction component is ignored by the transition rule. More explicitely, $3 \leftarrow, 3 \uparrow, 3 \rightarrow$ or $3 \downarrow$ will be taken simply as 3. The future state, on the other hand, may be any of the $4 * 3 + 1$ possible states, thereby leading to a redundant encoding. Thus the length of our encoding string is 4^5. This choice was made as it yielded better results.

[6] Strictly speaking, it is not exactly equivalent as a writing error induces a reading error for *all* surrounding cells.

The crossover used is the multi-point crossover described earlier.

The fitness function was at first exactly the one described by Lohn which we talked about earlier. This choice was motivated by the fact that fault-tolerance should naturally evolve from the environmental pressure: if it is not robust, it won't replicate. It turned out, however, that Lohn's fitness lead to very poor result, *even in non-faulty environment*. We thus adapted the fitness as outlined now.

The fitness F is decomposed in three subfitnesses F_r, F_g and F_p. F_r, the fitness that evaluates the number of replicants is defined as in Lohn's work, that is: $F_r = (1 + exp^{-(max(r_t)-\theta)})^{-1}$, where r_t is the number of replicants at time t and θ is the number of replicants from which the fitness importance of this measure decreases, here θ is 4. F_p the fitness that measures the quality of the position of the elements is also left unchanged. Let S_v be the average number of elements adjacent to element v. Let M_v^t be the number of elements v at time t. Finally, let $m_v(t)$ be the number of elements adjacent to v at time t that were of the same type and position relative to v at time 0. Then we define $o_v(t)$ to be 0 if $M_v^t \leq 1$ and $\frac{m_v(t)}{M_v^t . S_v}$ otherwise. Then $Fp = \frac{1}{T \Sigma_v S_v} \Sigma_v \Sigma_{t=1}^{T} S_v o_v(t)$. If F_r and F_p were left unchanged, we found necessary to modify F_g. The first runs highlighted a high tendency of the structure to grow as a compact pack, which in conjunction with a good relative position of elements, lead to very high fitness and premature convergence. Following the observation that interesting patterns of growth exhibited a growth rate of 1.7, we modified Lohn's definition of F_g as follows. Let $p_v(t)$ be 1 if $M_v^t > M_v^{t-1}$ and $M_v^t < 2 * M_v^{t-1}$, be 0.5 if $M_v^t = M_v^{t-1}$ and be 0 otherwise.

Finally we weighted quite differently from Lohn's work the fitnesses, here $F = 7.5F_g + 72.5F_p + 20F_r$.

The seeding configuration chosen contained a right angle three-element structure as shown in Figure 1. As said earlier two-element structures were devised as too easy and four-element results were deemed as too incidental. The grid is toroidal and of global size 50x50.

0	0	0	0
0	1	0	0
0	2	3	0
0	0	0	0

Fig. 1. The seeding configuration for all runs.

5 Results

In this last section we present the fault-tolerant self-replicating CAs obtained through evolution. As we will see these results, though structurally simple, are as complex as their non-robust counterpart found by evolution. In section 5.1, we present a decisive improvement in the encoding. In section 5.2, we detail the good

results obtained through a "straightforward" evolution. Finally, in section 5.3, we show how a more refined evolutionary process can lead to great improvements of the results.

5.1 Preliminary Experiments: on the Importance of Destruction

As said earlier, the fitness was modified to reduce the "attractiveness" of the compact growth strategy. This improved things but compact growth remained a strong basin of attraction and further fitness manipulation did not seem to do the trick. However it appears, if one thinks of what the main problem is with this behavior of the CA, that in fact growth is equivalent to a lack of cell destruction. In other words, not enough rules lead to the quiescent state. The main reason for this is quite obvious given our orientation insensitiveness. That property implies that there are 4 possible encodings for all states except for the quiescent state which has only one encoding. Thereby in the system as defined earlier there is only 1 chance in 13 (7.7%) of "choosing" cell destruction as its transition rule.

The main idea here is thus to introduce a redundant encoding for the quiescent state. As we found out from a batch of 50 experiments, in a safe environment, the optimal number of way of coding the quiescent state was 5. This is slightly more than 4, which is the number of redundant encoding for all the other states. Thereby the quiescent state must be over-represented to create better condition of success. To sum up the results, while the success rate was only 8% with no redundant encoding, it jumped to 60% with the optimal number of 5 quiescent states. We can note in passing that this is way better than Lohn's results [12].

5.2 First Experiments

The main idea behind that paper was that robustness in living organisms evolved as a consequence of their environmental pressure. Environment here is considered in its largest sense, that is the external environment pressure such as hardship, starvation, attack by predators etc. but also the system internal pressure such as defects in the cellular replication process, in the accuracy of the sensor, etc. and obviously anything in-between such as virus or parasites. Thereby, after the fine tuning of the fitness exposed before, the experiments run were intended as "spontaneous" evolution of robust self-replicating structures.

To do this we just changed the normal CA environment with the faulty one described in section 3, introducing a failure rate p_f. This failure rate means that, statistically, p_f faults occur every 100 transitions. This implies that the probability of one fault occurring on a structure of size n is $1-(1-p_f)^n$, where the size n of the structure encompasses all the surrounding quiescent cells of the non-quiescent structure. Thereby even the seeding three-element structure involves 15 cells, and thus a p_f of 1% implies a global failure rate of 14%; with a size of 100, which is quickly reached, the global failure rate is 63%. Using this environment, we are not facing a deterministic automata anymore. Running the same CA on the same seeding configuration may not lead to the same results. Therefore we introduce a new parameter, n_i, which denotes the number of repeated runs of

the same CA on the seed configuration used for evaluation. The n_i fitnesses then obtained are averaged to minimize as much as possible the stochastic bias.

The error rate tested out, p_f, were 0.5, 1, 1.5 and 2. As could have been expected from the simple global failure equation above, high error rates of 1.5 and 2% lead very quickly to complete chaos. On the other hand, evolving with $p_f = 0.5$ leads to too little selection pressure and, while a high success rate[7] is attained, the solutions found turn out to be more brittle than those found with a rate of 1%, which thus seems to be the most appropriate.

As for the number of iterations, on a theoretical point of view the higher the better. However, practically, evaluating the CA is the most time consuming part of the algorithm. Thereby one should aim at diminishing as much as possible that value.

n_i	p_f	runs	successful runs	interesting CAs	robust CAs
1	0.5	50	34	8	1
1	1	50	29	7	0
5	0.5	50	27	11	5
5	1	50	34	12	4
10	0.5	50	25	10	6
10	1	50	33	13	7

Table 1. Different success rates for different values of n_i and p_f. 'runs' is simply the number of runs on which the statistics were established, 'successful runs' are runs for which at least one replication occurred, 'interesting CAs' are those truly replicating, and finally 'robust CAs' are those at least moderately fault-tolerant.

From table 1 it is hard to draw definitive conclusions but one may say: – First and foremost, there is a saturation to the improvement that may be brought by an increase in n_i. Quite clearly a few repeats are enough to discriminate between different CAs. It may be concluded that a n_i of 5 is a good trade-off between the time required and the number of robust CAs found. Nevertheless, if we refine the analysis further by visual testings of the solution found, it turns out that they are less resistant than the ones found with a n_i of 10. The same remark is true for the lower p_f. While the number of robust CAs found is somewhat equivalent to those found with a p_f of 1, they appear visually to be brittler. Therefore the most favorable conditions seems to be a n_i of 10 and a p_f of 1, while the more reasonable ones would be $n_i = 5$ and $p_f = 1$.

The batch of experiments with the optimal settings for n_i and p_f (10 and 1) is studied here in more details. Figure 2 illustrates the fitness evolution through time of a typical successful run. Three kinds of successful strategies were found by evolution. The first strategy, which may be designated as opportunist, replicates along a diagonal in a safe environment, but uses error to create other lines of replication. Figure 3 illustrates a typical run of this strategy. The limit of this approach is that it does not correct error as such and given the toroidal nature of the grid error accumulates to the point where the CA collapses into chaos.

[7] An evolution is rated as successful if the best CA replicates at least once.

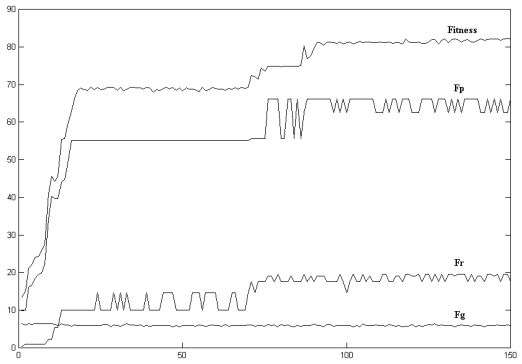

Fig. 2. Typical run fitness evolution through time. Here the population size is 200, the error rate, p_f, is 1%, the redundancy of quiescent state is implemented and n_i is 10. As one may see F_r and F_p are exhibiting exactly the same behavior after generation 8. However before F_r is on a plateau while F_p is continuously increasing. Quite obviously the relative position of the element increases before any real replication occurs. F_g remains almost constant as from generation 2 onwards. The selected individuals always tend to overgrow, thereby encountering the cut-off introduced in the fitness.

The second strategy, which one could call the gnat strategy, is simply based on fast exponential replications to compensate for the losses due to errors. Though the replication process in a safe environment of such automata is usually truly impressive, this strategy fails here and it eventually ends up in total chaos in a closed environment. Finally the third strategy, the no-error-zone strategy, simply consists at maintaining some sort of gap between the chaos generated by errors and the single diagonal line of replication. This last strategy, though not visually very interesting, is quite efficient at surviving. However like the other two strategies, it fails eventually.

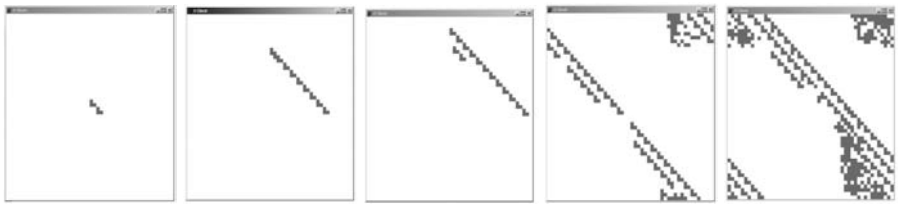

Fig. 3. Example of a "successful" strategy. An error appears at t=17 and is used to create a new front of replication at t=22, but then error accumulates and while the replication process still goes on, one may see at t=60 the seeds of the chaos to come. Though evolved with a p_f of 1%, it is demonstrated here with a p_f of 0.5%.

As we see, evolving robust self-replication by adding pressure from the environment is possible. However, even if faults are handled with, it does not satisfy

our original aim of correcting errors. In consequence, errors always submerge the CAs found through evolution. These mitigated results lead us to develop a more complex evolutionary process which is described in the next section.

5.3 Improving the Evolutionary Process

The results found until here were not completely satisfying. What was lacking in most of the solutions found was the ability to repair. To concentrate on this, we fixed the genes that were involved in the replication process under a deterministic environment and only evolved the genes left unused. This choice is not totally unreasonable if one thinks about short timescale evolution. In that context the more important genes of a population are fixed in the sense that every individuals have the same alleles at the same loci. To do this, a promising individual that has been found in the preceeding runs, is selected. It is then evaluated for a dozen time steps in a safe environment. The rules used during that evaluation are marked. It is then easy to generate a population of 200 individuals, random on all the unmarked rules, and identical on the marked ones. Except for the first population, the rest of the algorithm is identical as before.

As one can see in figure 4, this two-steps evolutionary process improves dramatically the results of the evolution. It gives rise to an almost perfectly fault-tolerant, self-replicating CA, correcting errors as expected (see figures 4 & 5).

Fig. 4. A selected individual from the previous runs is shown on the left at t=50. On the right, the result from the evolution of the population generated from that individual. As one may see the garbage has disappeared and with a p_f of 0.5 as shown here the CA on the left does never collapse into chaos.

6 Concluding Remarks

In this paper we have shown that it was possible to evolve fault-tolerant self-replicating structures using a straightforward evolutionary approach under faulty conditions pressure. Moreover, by implementing a two step evolutionary process, we have shown that error-correcting self-replicating CAs could be evolved.

Obviously the structures found were of small size, but not smaller than the one found before in deterministic space and the results are even better in terms of success rate. This indicates that research efforts should now concentrate first on refining the evolutionary process under deterministic condition. Indeed, the improvement we brought to the original algorithm may lead in the future to the evolution of more complex structures.

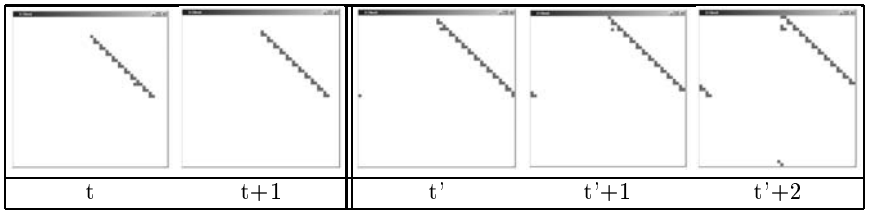

Fig. 5. The improved rule of figure 4 is shown on the left as correcting an error, while on the right it expels the error at time t'+1 and uses this to create a new self-replication from at t'+2.

References

1. Daniel C. Bünzli and Mathieu S. Capcarrere. Fault-tolerant structures: Towards robust self-replication in a probabilistic environment. In J. Kelemen *et al* (eds), *European Conference on Artificial Life 2001*, pages 90–99. Springer Verlag, 2001.
2. Mathieu S. Capcarrere. *Cellular Automata and Other Cellular Systems: Design & Evolution*. Phd No 2541, Swiss Federal Institute of Technology, Lausanne, 2002.
3. Peter Gács. Self-correcting two-dimensionnal arrays. In Silvio Micali (ed), *Randomness in computation*, vol. 5 of *Advances in Computing Research*, pages 223–326, Greenwich, Conn, 1989. JAI Press.
4. Peter Gács. Reliable cellular automata with self-organization. In *Proceedings of the 38th IEEE Symp. on the Foundation of Computer Science*, pages 90–99, 1997.
5. Masaretu Harao and Shoici Noguchi. Fault tolerant cellular automata. *Journal of computer and system sciences*, 11:171–185, 1975.
6. Christopher G. Langton. Self-reproduction in cellular automata. *Physica D*, 10:135–144, 1984.
7. Jason D. Lohn and James A. Reggia. Discovery of self-replicating structures using a genetic algorithm. In *Proceedings of the 1995 IEEE International Conference on Evolutionary Computing (Perth)*, pages 678–683. Springer-Verlag, 1995.
8. Jason D. Lohn and James A. Reggia. Automatic discovery of self-replicating structures in cellular automata. *IEEE Trans. on Evol. Computation*, 1:165–178, 1997.
9. Daniel Mange, Moshe Sipper, Andre Stauffer, and Gianluca Tempesti. Towards robust integrated circuits: The embryonics approach. *Proc. of the IEEE*, 88(4):516–541, April 2000.
10. Barry McMullin. John von neumann and the evolutionary growth of complexity: Looking backwards, looking forwards... In Mark A. Bedau *et al*, (eds), *Artificial Life VII: Proceedings of the seventh international conference*, pages 467–476, Cambridge, MA., 2000. MIT Press.
11. A. N. Pargellis. Digital life behavior in the amoeba world. *Artificial Life*, 7(1-2):63–75, Winter 2001.
12. James A. Reggia, Hui-Sien Chou, and Jason D. Lohn. Self-replicating structures : Evolution, emergence and computation. *Artificial Life*, 4:283–302, 1998.
13. Moshe Sipper. Fifty years of research on self-replication: An overview. *Artificial Life*, 4:237–257, 1998.
14. Gianluca Tempesti. A new self-reproducing cellular automaton capable of construction and computation. In F. Morán *et al* (eds), *Proc. 3rd European Conf. on Artificial Life 1995*, vol. 929 of *LNAI*, pages 555–563. Springer-Verlag, 1995.