Correlation Analysis of the Synchronizing-CA Landscape

Wim Hordijk

Santa Fe Institute, 1399 Hyde Park Road, Santa Fe, NM 87501, phone: (505) 984-8800, fax: (505) 982-0565, email: wim@santafe.edu

A correlation analysis will be applied to subspaces of the fitness landscape generated by the synchronization task for one-dimensional cellular automata. This results in a stochastic model that can be used to characterize the correlation structure of those subspaces. The results show that both subspaces can be characterized by an AR(2) model, both for point mutation and crossover.

Key words: Fitness landscapes. Correlation analysis. Cellular automata. AR models. Landscape structure.

1 Introduction

In [6] an extended correlation analysis for fitness landscapes is introduced. This analysis is based on a time series analysis known as the Box-Jenkins approach, which tries to identify and estimate a model that adequately represents a given time series. In [6] the analysis is demonstrated using the NK-model [8,9], which generates a family of fitness landscapes of which the "ruggedness" can be tuned.

However, fitness landscapes generated by the NK-model are rather "artificial" and have "well-behaved" properties that are relatively easy to measure. To show that the proposed correlation analysis still works well on more realistic, less well-behaved landscapes, it is applied here to the fitness landscape that results from the synchronization task for cellular automata [2]. To be able to apply the correlation analysis on this landscape in a useful way, subspaces in the landscape have to be characterized first, to which the correlation analysis is restricted, as was already suggested in [6]. The analysis results in a stochastic model that characterizes the correlation structure of those subspaces.

The next section explains the synchronization task for cellular automata. Section 3 gives a brief overview of the proposed correlation analysis for fitness

Preprint submitted to Elsevier Preprint *14 January 1997*

landscapes. Section 4 then shows how subspaces in the "synchronizing-CA landscape" are identified. Section 5 then shows the results of applying the correlation analysis to these subspaces in the landscape. Finally, section 6 will summarize the main conclusions.

2 The Synchronization task for Cellular Automata

In [2] a space of cellular automaton rules is searched, trying to find a rule that is capable of performing some non-trivial computation. A (one-dimensional) cellular automaton (CA) [3,12,15] is a lattice of cells, each of which can be in one of a number k of states (say zero and one for a two-state CA). At each time step, all cells simultaneously update their state, looking at their current state and those of their r direct neighbors to the left and to the right, and consulting an update rule which is the same for each cell. Iterating this over a number of time steps, and plotting the lattice over time (called a *space-time diagram*), can give rise to complicated patterns, and for some rules even to some sort of computation.

A CA update rule can be represented as a so called *lookup table*, that has an entry for each possible neighborhood configuration a cell can be in (i.e., each possible configuration of states a cell and its $2r$ neighbors can be in; so there are k^{2r+1} such configurations). For each possible configuration, a new state is given, which will be the state of the cell at the next time step given that it is in that particular neighborhood configuration at the current time step. For two-state CAs, a lookup table can thus be represented by a bit string of length 2^{2r+1}, and there are $2^{2^{2r+1}}$ different lookup tables possible.

The computational task stated in [2] is *global synchronization*: the space-time behavior of the CA has to settle down to a synchronized oscillating behavior between all 0s and all 1s. Figure 1 shows a space-time diagram of a particular rule that was found for this task. Global synchronization is a non-trivial task, requiring global coordination between all the cells in the CA lattice, while each cell is only capable of interacting locally (i.e., only with neighboring cells). This means that local information has to be transferred to other regions in the lattice, to achieve global coordination.

It turns out that the CAs capable of performing the synchronization task make use of "traveling" local structures in the space-time diagram, called "particles", to accomplish this information transfer (see [4,5] for a formal definition of particles). In Figure 1, the particles can be seen at the "boundaries" between the more or less stable regions where a synchronized oscillating pattern or a zigzag pattern can be observed. The particles contain information about the structure of a local region in the lattice, and transfer this information to other regions by moving through the lattice. When two particles "collide" they exchange their information, for example by creating other particles, or annihilating each other, as can be observed in Figure 1. These particles form part of the "strategy" that the CA employs to perform the given task (see [2,7] for a more detailed description of computational strategies in CAs).

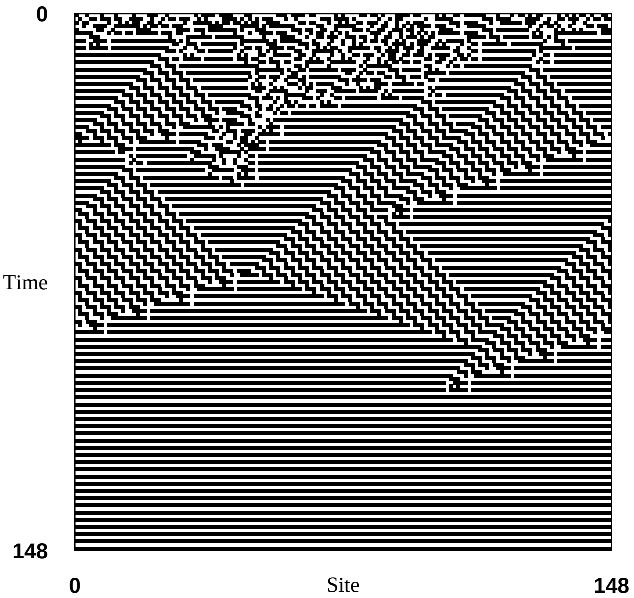

Fig. 1. A space-time diagram of a particular CA that solves the synchronization task. 0s are represented by white, 1s by black. After [2].

In [2], two-state CAs ($k = 2$) with $r = 3$ were used, so the length of the lookup tables is $2^7 = 128$, and there are 2^{128} such lookup tables in the search space. The fitness landscape resulting from this task, the "synchronizing-CA landscape", is made up of all possible 2^{128} CA rules, where the fitness of each CA rule is its ability to perform the synchronization task. This is measured by giving a CA rule a set of random initial configurations, and then simply counting the fraction of initial configurations in this set on which the CA synchronizes within some maximum number of iterations.

3 The correlation analysis

In this section a brief overview of the correlation analysis, as introduced in [6] is given. The analysis starts by performing a random walk on the given landscape, using some move operator of interest (for example mutation or crossover). At every step the fitness value of the current point is recorded, thus generating a time series of fitness values. Then the *Box-Jenkins approach* [1] is applied to this time series of fitness values.

The Box-Jenkins approach is a time series analysis that tries to find an autoregressive moving-average (ARMA) model of an appropriate order, that adequately represents the data generating process from which the given time series resulted. An ARMA(p,q) model is of the form:

$$y_t = c + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{j=0}^{q} \beta_j \varepsilon_{t-j} \ , \qquad \beta_0 = 1$$

i.e., the value at the current time step depends on p previous values (i.e., the amount of memory in the process), and on a weighted average of a *white noise* series (a stochastic variable with zero mean and fixed finite variance, and no correlation between values at different time steps).

The approach consists of three stages (see [1,6] for a more detailed description of the stages):

(i) *Identification* in which the order of the model is identified (i.e., appropriate values for the parameters p and q are identified).
(ii) *Estimation* in which the actual parameters of the model (α_i and β_j) are estimated, given the identified values of p and q.
(iii) *Diagnostic checking* in which the model is tested for adequateness.

In the process, the approach estimates the autocorrelations of the given time series for different time lags. From these estimates a correlation length for the landscape can be derived. To decide whether an estimate is still (statistically) significant, an approximate two-standard-error bound of $\pm 2/\sqrt{T}$ is used, where T is the length of the time series.

The stochastic model that results from applying the Box-Jenkins approach to a time series of fitness values generated by a random walk on the landscape, together with statistics about the explanatory and predictive value of the model, can serve as a measure of the correlation structure of that fitness landscape [6]. This correlation analysis is more complete and informative than standard correlation analyses [10,11,13], which basically only calculate a value for the average correlation between points in the landscape.

One requirement for the above analysis to work, is that the landscape is *isotropic*. Informally, this means that the landscape is "statistically the same everywhere". Regardless of where the random walk is started, the resulting statistics will always be the same (within some statistically acceptable range). As is already suggested in [6], if this requirement is not fulfilled, then it might be possible to characterize subspaces in the landscape that are "locally" isotropic. The correlation analysis can then be applied to each of these subspaces, by restricting the random walk to one particular subspace.

4 Characterizing subspaces in the synchronizing-CA landscape

It turns out that the synchronizing-CA landscape is highly non-isotropic. Performing a random walk on the entire landscape and applying the Box-Jenkins approach results in meaningless statistics, that are different every time the process is repeated. Therefore, two subspaces in the synchronizing-CA landscape are characterized, that somehow "contain" a particular strategy that a CA employs to solve the synchronization task. This characterization of subspaces is done by identifying bits in the lookup table that are necessary for "supporting" a particular strategy, and keeping those bits fixed. The remaining bits can then be used to perform a random walk on, and to apply the correlation analysis.

As was mentioned in section 2, part of the strategy that a CA uses to perform the synchronization task, is made up by so called particles that can be identified in the space-time diagram. Figure 2 shows space-time diagrams of two particular CA rules that occurred (early on) during a search through the CA rule space. In figure 2 (left), three different particles can be identified (numbered 1,2 and 3), and in figure 2 (right) two different ones (numbered 1 and 4, because particle number 1 here turns out to be the same as particle number 1 in figure 2 (left)). We will refer to the CA rule shown in figure 2 (left) as rule "A", and to the CA rule of figure 2 (right) as rule "B". The lookup tables for both rules are given below, where the first bit is for the all 0s neighborhood configuration, and the last bit for the all 1s neighborhood configuration. Note that both lookup tables differ in only one bit position (marked by an underline), but give rise to different sets of particles used in the respective strategies.

Rule A:
1100111010110010111011110010100011000110100011010010101000000100
111000110100000111111010111000101110011100011000011110101$\underline{1}$101000
Rule B:
1100111010110010111011110010100011000110100011010010101000000100
111000110100000111111010111000101110011100011000011110101$\underline{0}$101000

To characterize a subspace in the fitness landscape that somehow represents, or "contains", a particular strategy employed by a CA, a first requirement is that the particles that are used in that strategy are being fixed. To do this, all the bits in the lookup table that are "used" by the particles have to be identified and remain fixed. Since particles are local, repeating structures, only a fraction of all the bits in the lookup table is necessary to "support" one particular particle. The entries in the lookup table that represent neighborhood configurations that do not occur within a particle, will obviously never be used by that particle.

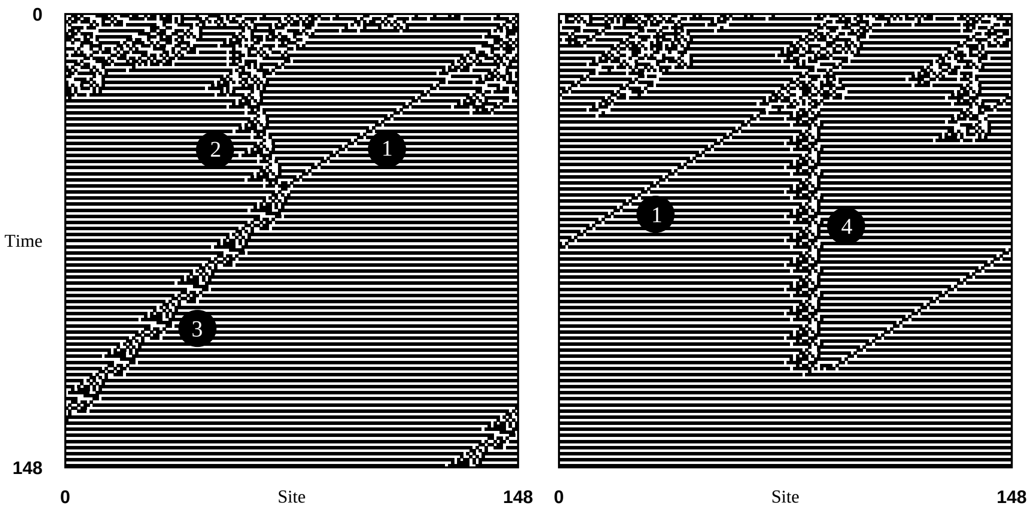

Fig. 2. Space-time diagram of CA rule A (left) and CA rule B (right), showing the two different computational strategies used to solve the synchronization task. The different particles are labeled 1–4.

Identifying the bits in the lookup table that are used by the particles in a certain strategy is a straightforward procedure. So the first approach to characterize a subspace in the synchronizing-CA landscape, will be to fix all the bits in the lookup table that are used by at least one of the occurring particles in a strategy. The result of this is shown below for both CA rules, where a dot (.) means that that bit is not used by any of the particles in the strategy; it is a "free" bit that can be used to do the random walk on.

Rule A:
```
110011101011.0.01.10.111..10.0.01.0.01.01000.1.10...1010.0.00100
11..0...0100..0111111.101.100.101...0..1000110000.1110.011101000
```
Rule B:
```
1100111.10.1.0..1..0..11..101..01.0....01....1.1....10.00...0.00
1...0..........0111.11.1...1...101.....10001...00..110.0.0101000
```

A second characteristic of a strategy is that it has a certain fitness associated with it (how well it performs on the given task). For example, rule A has a fitness of 0.2679 (meaning, in this case, that the CA synchronizes within a fixed maximum number of time steps on 2679 out of 10,000 random initial configurations). However, the fitness of rule B is 0.5180 (it synchronized on 5180 out of 10,000 random initial configurations).

So a second requirement for the subspace is that the fitness values of the rules within the subspace do not differ too much from that of the rule (strategy) that is used in the first place to characterize the subspace. As an approximation to this requirement, first the fitnesses of all the one-mutant neighbors (i.e., all CA rules that differ only in the value of one bit in the lookup table) of both rules are calculated (in a similar way as for the original rules themselves). Next, the fitnesses of the one-mutant neighbors that correspond to bits that are still free (i.e., those bits that were not already fixed because they are used by the particles) are compared to the fitness of the original rule. If the fitness of a one-mutant neighbor corresponding to a particular free bit differs too much from that of the original rule, than that particular bit is fixed as well. If the fitness does not differ too much, then that bit remains free. As a bound on how much the fitness of a one-mutant neighbor is allowed to differ from that of the original rule, an approximate two-standard-error bound is used (in this case $\pm 2/\sqrt{10,000} = 0.02$, because the fitness is calculated over 10,000 initial configurations).

So eventually, all the remaining free bits are those bits in the lookup table that are not used in any of the particles that make up the strategy, *and* for which the fitness does not change too much when their value is changed. The resulting lookup tables for both rules are shown below. Again, dots (.) denote the free bits. For rule A there is a total of 14 free bits, for rule B there are 11 free bits. The random walks for the correlation analysis will be restricted to these free bits only, while the rest of the bits remain fixed, thus restricting the analysis to a particular subspace of the landscape that represents a particular strategy.

Rule A:
`110011101011.0101.1011110.101000110001101000110100..101000.00100`
`11100.1101000.0111111.10111000101...0.11000110000.11101011101000`
Rule B:
`11001110101100.01.1011110.10100011000110100011010.1010.000000100`
`111000110..0000111111010111000101.1.0.1100011.00011110101010100 0`

The subspace derived from rule A will be denoted by subspace A, and the subspace derived from rule B by subspace B. The correlation analysis can now be applied to both these subspaces.

5 Results

Random walks on the two subspaces characterized in the previous section are performed with two different move operators: point mutation and random-mate-random-child crossover (rmrc crossover). The point mutation operator just randomly flips one bit at each time step. The rmrc crossover operator starts with a random bit string, picks a random string as mate, performs one-point crossover, and chooses one of the children at random to proceed with, picking a new random mate at the next step, etc. (see [6]). Of course, in this case, only the free bits are allowed to be flipped in point mutation, or set at random in rmrc crossover.

Random walks of 1,000 steps are performed, where the fitness at each step is calculated over a set of 10,000 random initial configurations (a different set at each step). The approximate two-standard-error bound used in the Box-Jenkins approach is $\pm 2/\sqrt{1,000} = \pm 0.0632$.

5.1 Point mutation

Identification
Figure 3 (left) shows the estimated autocorrelations for subspace A (the result for subspace B looks similar). Also, the approximate two-standard-error bound of ± 0.0632 is plotted. From this plot, it can be derived that the correlation length in both subspaces for point mutation is around 10 steps (i.e., the first time lag at which the autocorrelation falls within the two-standard-error bound).

Figure 3 (right) shows the estimated partial autocorrelations for subspace A (again, the result for subspace B looks similar). It shows that the first two partial autocorrelations are outside the approximate two-standard-error bound, while the rest are within this bound. This, together with the gradual decline of the autocorrelations, suggests an AR(2) model to be estimated (see [6] for the motivation for this).

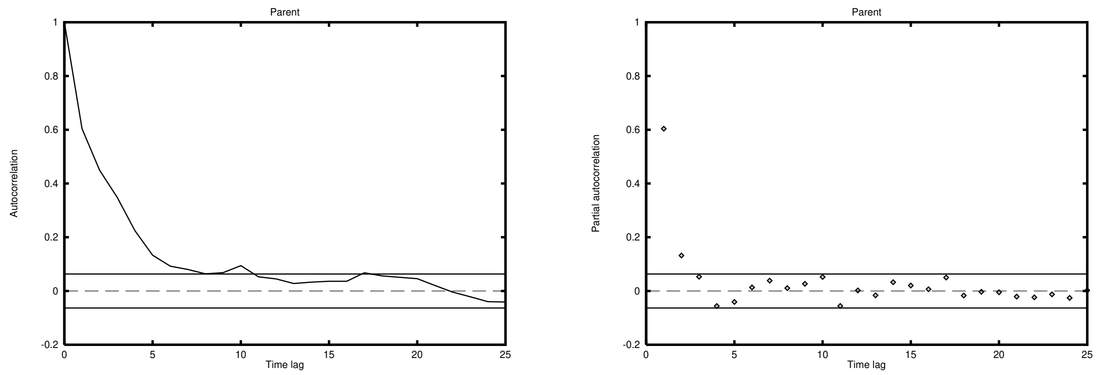

Fig. 3. The first 25 autocorrelations (left) and partial autocorrelations (right) for point mutation in subspace A.

Estimation
For both subspaces, an AR(2) model is estimated. Table 1 shows the results of this estimation. As a measure of significance of the estimated parameters, their t-statistics are given in parentheses. Also, the variance of the stochastic white-noise variable, and the adjusted R-squared (a measur of the goodness of fit of the estimated model) are provided.

Diagnostic checking
To check the adequateness of the estimated model, figure 4 shows the residual

	Estimated model	$\mathrm{Var}(\varepsilon_t)$	$\overline{R}^2$
subspace A	$\underset{(12.0)}{0.07}+\underset{(16.7)}{0.52}y_{t-1}+\underset{(4.2)}{0.13}y_{t-2}+\varepsilon_t$	0.0019	0.38
subspace B	$\underset{(12.7)}{0.16}+\underset{(17.0)}{0.54}y_{t-1}+\underset{(3.0)}{0.09}y_{t-2}+\varepsilon_t$	0.0071	0.35

Table 1
The results of estimating an AR(2) model for point mutation.

autocorrelations for subspace A (results for subspace B look similar). Besides two minor exceptions, they are all well within the two standard-error-bound (as they should be, because the residuals should be uncorrelated).

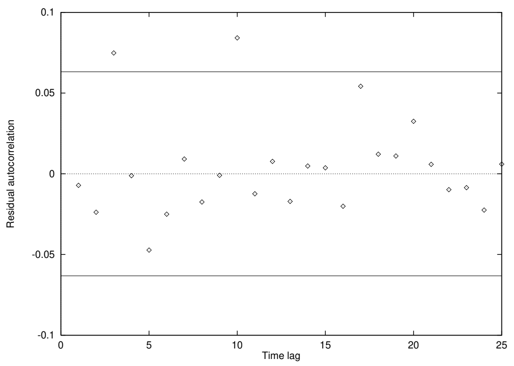

Fig. 4. The first 25 residual autocorrelations for point mutation in subspace A.

As an extra check, for both subspaces an AR(3) model was estimated. The t-statistics of the extra parameter for respectively subspace A and B are 1.7 and 1.2. Both are less than 2, so in both cases the extra parameter is not significant.

So it seems that the estimated AR(2) models for both subspaces for point mutation are adequate enough.

5.2 Rmrc crossover

Identification
Figure 5 (left) shows the estimated autocorrelations for subspace B (the result for subspace A looks similar). Also, the approximate two-standard-error bound of ± 0.0632 is plotted. From this plot, it can be derived that the correlation length in both subspaces for crossover is around 3 steps (i.e., the first time lag at which the autocorrelation falls within the two-standard-error bound).

Figure 5 (right) shows the estimated partial autocorrelations for subspace B (again, the result for subspace A looks similar). It shows that the first two partial autocorrelations are outside the approximate two-standard-error bound, while the rest are within this bound. This again suggests an AR(2) model to be estimated here.

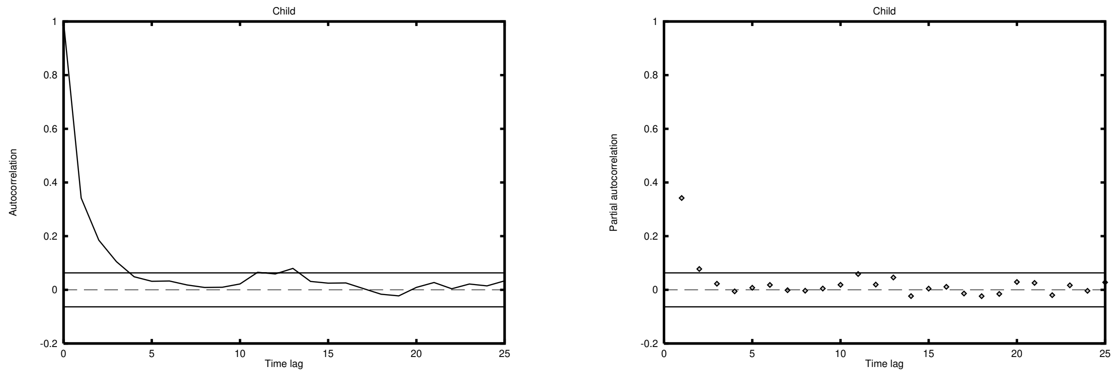

Fig. 5. The first 25 autocorrelations (left) and partial autocorrelations (right) for crossover in subspace B.

Estimation

Table 2 shows the results of the estimation. The t-statistics are given in parentheses. Also, the variance of the stochastic white-noise variable, and the adjusted R-squared (a measure of the goodness of fit of the estimated model) are provided.

	Estimated model	$\mathrm{Var}(\varepsilon_t)$	$\overline{R}^2$
subspace A	$0.13{+}0.32y_{t-1}{+}0.07y_{t-2}+\varepsilon_t$	0.0036	0.12
	(16.4) (10.1) (2.3)		
subspace B	$0.25{+}0.32y_{t-1}{+}0.08y_{t-2}+\varepsilon_t$	0.0108	0.12
	(16.3) (10.0) (2.5)		

Table 2
The results of estimating an AR(2) model for crossover.

Diagnostic checking

To check the adequateness of the estimated model, figure 6 shows the residual autocorrelations for subspace B (results for subspace A look similar). It appears that they are all well within the two standard-error-bound.

As an extra check, for both subspaces an AR(3) model was estimated. The t-statistics of the extra parameter for respectively subspace A and B are -0.7 and 0.7. Both are less than 2 (in absolute value), so in both cases the extra parameter is not significant.

So again it seems that the estimated AR(2) models for both subspaces for rmrc crossover are adequate enough.

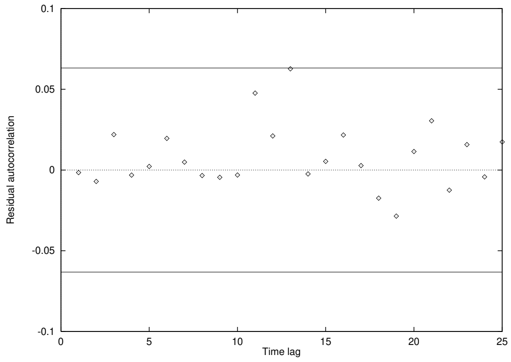

Fig. 6. The first 25 residual autocorrelations for crossover in subspace B.

6 Conclusions

Both subspaces that were identified in the synchronizing-CA landscape can be characterized by an AR(2) model, for both point mutation and rmrc crossover. As can be seen from the similarity in the estimated models for both subspaces for one particular move operator, both subspaces have very similar correlation structures.

In an AR(2) model, a "memory" of two time steps is used. This means that in predicting the fitness at the current time step, the fitness values of the two previous time steps provide relevant and useful information. This brings up an interesting question: Would it be possible to construct a more efficient search algorithm that makes use of the knowledge of the fitness at the two previous time steps, instead of just one, as is done in general search algorithms like hill climbing and genetic algorithms (which are based on mutation and/or crossover)? If so, then identifying a landscape as AR(2) also gives information about what kind of search algorithm should be used.

Weinberger [14] suggest that AR(2) landscapes are a "combination" of two (independent) AR(1) sub-landscapes. Performing a random walk on the AR(2) landscape, with probability c a step is taken on the first sub-landscape, and with probability $1 - c$ a step is taken on the second sub-landscape. If this is true, then it would be interesting to try to find out what the two underlying sub-landscapes are here, and what the value of the parameter c is.

One of the goals of this paper was to show that the in [6] proposed landscape analysis will still work nicely on less well behaved, non-isotropic landscapes. This turns out to be true, as long as subspaces in the landscape can be identified that are locally isotropic enough to produce good results. Identifying these subspaces will generally be the most difficult part, but once they are found, the landscape analysis can be applied in a straightforward way.

Acknowledgement

This research was supported by the Santa Fe Institute under grants from the National Science Foundation (grant NSF IRI-9320200) and the Department of Energy (grant DE-FG03-94ER25231). Many thanks to Peter Stadler for reading an earlier version of this paper, and to Jim Crutchfield and Melanie Mitchell for support and for providing the opportunity to do this work.

References

[1] G. E. P. Box and G. M. Jenkins. *Time Series Analysis, Forecasting and Control.* Holden Day, 1970.

[2] R. Das, J. P. Crutchfield, M. Mitchell, and J. E. Hanson. Evolving Globally Synchronized Cellular Automata. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 336–343. Morgan Kaufmann, 1995.

[3] H. A. Gutowitz. *Cellular Automata.* MIT Press, 1990.

[4] J. Hanson and J. Crutchfield. The Attractor-Basin Portrait of a Cellular Automaton. *Journal of Statistical Physics*, (66):1415–1462, 1992.

[5] J. E. Hanson. *Computational Mechanics of Cellular Automata.* PhD thesis, University of California, Berkeley, CA, 1993.

[6] W. Hordijk. A Measure of Landscapes. *Evolutionary Computation*, 4(4), 1996.

[7] W. Hordijk, J. P. Crutchfield, and M. Mitchell. Embedded-Particle Computation in Evolved Cellular Automata. In T. Toffoli, M. Biafore, and J. Leão, editors, *Proceedings of the Fourth Workshop on Physics and Computation*, pages 153–158. New England Complex Systems Institute, 1996.

[8] S. A. Kauffman. Adaptation on Rugged Fitness Landscapes. In D. Stein, editor, *Lectures in the Sciences of Complexity*, pages 527–618. Addison-Wesley, 1989.

[9] S. A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[10] M. Lipsitch. Adaptation on Rugged Landscapes Generated by Iterated Local Interactions of Neighboring Genes. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 128–135. Morgan Kaufmann, 1991.

[11] B. Manderick, M. d. Weger, and P. Spiessens. The Genetic Algorithm and the Structure of the Fitness Landscape. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 143–150. Morgan Kaufmann, 1991.

[12] T. Toffoli and N. Margolus. *Cellular Automata Machines.* MIT Press, 1987.

[13] E. D. Weinberger. Correlated and Uncorrelated Fitness Landscapes and How to Tell the Difference. *Biological Cybernetics*, (63):325–336, 1990.

[14] E. D. Weinberger. Fourier and Taylor Series on Fitness Landscapes. *Biological Cybernetics*, (65):321–330, 1991.

[15] S. Wolfram. *Cellular Automata and Complexity.* Addison-Wesley, 1994.