

The Structure of the Synchronizing-CA Landscape

Wim Hordijk

SFI WORKING PAPER: 1996-09-077

SFI Working Papers contain accounts of scientific work of the author(s) and do not necessarily represent the views of the Santa Fe Institute. We accept papers intended for publication in peer-reviewed journals or proceedings volumes, but not papers that have already appeared in print. Except for papers by our external faculty, papers must be based on work done at SFI, inspired by an invited visit to or collaboration at SFI, or funded by an SFI grant.

©NOTICE: This working paper is included by permission of the contributing author(s) as a means to ensure timely distribution of the scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the author(s). It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may be reposted only with the explicit permission of the copyright holder.

www.santafe.edu



SANTA FE INSTITUTE

The Structure of the Synchronizing-CA Landscape

Wim Hordijk
Santa Fe Institute
1399 Hyde Park Road
Santa Fe, NM 87501
wim@santafe.edu

Abstract

In many complex systems control situations, searching for solutions or alternatives is involved. Searching for solutions can be modeled by a search on a fitness landscape. Knowing the structure of the underlying landscape can help in explaining or predicting aspects of an actual search on it, and thus in controlling the system that is living on the landscape.

This paper presents results on characterizing the structure of the fitness landscape that results from searching for a cellular automaton, a simple mathematical model of a complex system, that can perform a certain non-trivial computational task (global synchronization). The structure of this landscape turns out to be quite different from the more standard fitness landscapes that have been looked at so far. It is furthermore shown that the characterization of this structure can explain certain phenomena that are observed in an actual search on this landscape.

1 Introduction

Controlling a complex nonlinear system requires a great deal of knowledge about how the system works and, more importantly, how it responds to adjustments. Control often involves a search among several alternative adjustments to a system, looking for the one that results in the desired (change in) behavior of the system. But for most complex nonlinear systems it is impossible or too costly to approach this search analytically, and much of the search relies on trial and error. It is therefore desired to have some knowledge about the structure of the underlying search space, in order to “guide” the search.

Using a metaphor from biology, looking for alternatives in a search space can be modeled by a search on a fitness landscape. A *fitness landscape* is a usually high dimensional space where each point represents, for example, a possible solution

to a problem, or a possible internal state of a system. Furthermore, points in the landscape are connected to each other, and the connections, usually called the *neighborhood relation*, are determined by the move operator that is used to search the space of alternatives. Two points in the landscape are connected if one can be reached from the other by applying the move operator once. Lastly, each point is assigned a *fitness*, which is a measure of how good a solution, represented by that point, for the given problem it is. This representation space, together with the neighborhood relation and the assigned fitness values, can now be envisioned as a more or less mountainous landscape (the height of a point is given by its fitness value), where the highest peaks denote the best solutions. A search on a fitness landscape looks for the highest regions in the landscape, or in other words tries to “walk uphill”.

To understand a complex nonlinear system that is “living” on a fitness landscape, it is very useful to have some insight in the structure of that landscape, because it can reflect how easy or difficult it is for a search algorithm to find good solutions [MWS91, MFH92]. Usually the landscape is too large, i.e., contains too many points, to exhaustively search it. However, there are some general statistics that can be derived from the landscape which can help in building a global characterization of the structure of the landscape. One could, for example, look at the height and location distributions of peaks in the landscape, the average number of steps necessary to reach a peak, or the correlation structure of the landscape, by sampling points in the landscape.

So one way of characterizing a complex nonlinear system for purposes of control is to gather statistics that give insight in the structure of the fitness landscape on which the system is living. Knowledge about this structure can help in searching for good solutions or alternatives, or in explaining the behavior of the system or the search process.

This paper presents some results of characterizing the structure of a fitness landscape that results from searching for a cellular automaton (a simple model of complex nonlinear systems) that can perform a certain non-trivial computational task, called synchronization (see [DCMH95]). The next section explains the concept and usefulness of fitness landscapes in more detail. Section 3 explains what cellular automata (CAs) are, and explains the specific task that the CA is required to perform. The section following that presents the results of characterizing the structure of the resulting fitness landscape. Section 5 then shows how this characterization can be used to explain or predict certain aspects of an actual search on the landscape.

2 Fitness Landscapes

The notion of a fitness landscape comes from biology (first introduced in [Wri32]), where it is used as a framework for thinking about evolution. Biological organisms can be represented by their *genotype*, which is the genetic encoding of the organism, or their *phenotype*, which is the actual form and behavior of the organism. The abstract notion of *fitness* can be assigned to each phenotype, which reflects its ability to survive and reproduce. Evolution can now be viewed as a process that searches a fitness landscape of possible genotypes, looking for genotypes that represent (encode) highly fit phenotypes.

This fitness landscape paradigm can be used for search in general. Given a problem and a set of possible solutions to that problem, the first step is to find an encoding, or representation, for the possible solutions. In the traveling salesperson problem, for example, the goal is to find the shortest possible tour along a number of cities, visiting each city exactly once. So the set of possible solutions is the set of all possible tours along those cities. An encoding for these possible solutions, given a numbering of the cities, could be a list of integers, denoting the order in which to visit the cities. So if there are N cities, numbered 1 to N , the encoding consists of all the possible permutations of $\{1, \dots, N\}$, thus creating a *representation space*.

The next thing that is needed is a *neighborhood relation*. This relation indicates which points in the representation space are connected to each other (effectively imposing a graph structure on the set of points). This neighborhood relation is usually defined by the *move operator* (or combination of move operators) that is used to search the representation space. In the traveling salesperson example, a move operator could for example be swapping two cities in the current order (permutation). By applying this move operator, a move is made from one point in the representation space to another. Each pair of points in the representation space that can be reached from one another by applying the move operator exactly once, is thus connected. The representation space is now a graph that can be traversed by applying the move operator repeatedly.

Finally, a *fitness function* is needed. This function takes as input an encoding for a possible solution to the given problem, converts the encoding to the actual solution it represents, and returns a value that denotes how good this solution is for the given problem. The better a solution, the higher the fitness value. For the traveling salesperson problem, the fitness function will take as input an ordering, or permutation, of the cities and calculate the length of the tour when the cities are visited in that particular order. Since a better solution, in this case, is a *shorter* tour length, the fitness function could for example return the inverse of the tour length, so that better solutions have higher fitness values. Using this fitness function, a fitness can be assigned to each point in the representation space.

Taken together — the representation space, the neighborhood relation, and the fitness function — they define a fitness landscape. Graphically, this can be imagined as a space of connected points, where each point has a height according to its fitness value, giving rise to the image of a more or less mountainous landscape where the

peaks denote good solutions to the given problem. When searching on a fitness landscape, the goal is usually to find the highest peak(s).

Summarizing, a fitness landscape is defined by three things:

1. A representation space (i.e., encodings for possible solutions).
2. A neighborhood relation that defines which points are connected in the representation space.
3. A fitness function that assigns a fitness value to each point in the representation space.

There are several characteristics that make up the structure of a fitness landscape. For example, one might be interested in the number of local optima in the landscape. A *local optimum* is defined as a point in the landscape that has equal or higher fitness than all of its neighbors (i.e., the points it is directly connected to). On a landscape with a lot of local optima, there is a danger that a search will get stuck in a local optimum that is (much) less fit than the actual global optimum (the point with the highest fitness in the landscape). Other characteristics are the distribution of the (relative) heights and locations of local optima, or the number of steps (applications of the move operator) it takes to reach a local optimum starting at a random point in the landscape.

Another important characteristic of a fitness landscape is its “ruggedness”. This means the (average) fitness differences between neighboring points. If the fitness difference between neighboring points is very small, the landscape is very smooth and there is a high *correlation* between the fitness values of neighboring points. If the fitness difference is very large, the landscape is very rugged and there is very little correlation. So the more correlated the landscape is, the “further away one can look”, still having some information about the fitness values of distant points. Roughly, the correlation structure of a landscape indicates how easy it is to keep walking uphill.

The landscape metaphor is useful in several ways. First of all, it gives rise to intuitively simple images like hills, valleys and peaks, that can help in understanding search processes in general. Secondly, the definition of a fitness landscape (which is basically a labeled graph) lends itself to rigorous mathematical analysis [Jon95, Sta95]. Finally, as was mentioned in the introduction, the structure of a fitness landscape can reflect how easy or difficult it is for a search algorithm to find good solutions [MWS91, MFH92].

The next section introduces a fitness landscape that results from searching through a space of cellular automaton rules for a rule that is capable of performing a certain computational task, called synchronization.

3 Cellular Automata and the Synchronization Task

Cellular automata (CAs) [TM87, Gut90, Wol94] are a class of simple mathematical models that can give rise to complicated behavior. Therefore, they are often used as simple models of complex (nonlinear) systems. In the simplest case, a CA is a one-dimensional lattice of length L of identical cells, each of which can be in one of two possible states (for example black or white). The collection of all these cells and their particular values is called a *configuration*. The lattice is updated in discrete time steps, where at each time step all cells simultaneously update their state. There is a deterministic update rule that each cell consults, which takes as input the current state of a cell and that of its direct neighbors, and returns the new state of the cell. The number of neighbors, r , to the left and to the right of a cell that are used in the update rule is called the *radius*. Usually, the lattice has periodic boundary conditions, which means that the lattice is “wrapped around” at the ends, i.e., cell 1 and cell L are each others neighbors.

The update rule is simply a lookup table which gives the new state for each of the 2^{2r+1} possible neighborhood configurations a cell can be in. Since there are 2^{2r+1} entries in the lookup table, and each entry can have two possible values for the new state, there is a total of $2^{2^{2r+1}}$ possible update rules. Starting with a random initial configuration — randomly assigning black or white to each cell in the lattice —, iterating the lattice using one of the possible update rules, and plotting the configurations over time, gives rise to a *space-time diagram* which can display (depending on the specific update rule) complicated patterns.

It is even possible to perform certain computations with CAs. One way of having a CA perform a computation is to encode an input in an initial configuration (IC), then let the CA iterate for a certain number of time steps, transforming the input to some output, and read back, or decode, the output from the final configuration. In [DCMH95] a space of CA update rules is searched to find a rule that is capable of doing a non-trivial computation called *synchronization*: the space-time behavior of the CA has to settle down to a synchronized periodic oscillation between an all white configuration and an all black configuration. The input is just a random initial configuration, and the CA is given a certain number of time steps to reach the globally synchronized state, which can be considered as the desired output. An example is given in Figure 1 (taken from [DCMH95]), where global synchronization is reached roughly at time step 100. This task is non-trivial for a CA, since it requires global coordination between all the cells, while each cell can only interact locally with its direct neighbors. For a detailed description of how the CA actually does the synchronization task, see [DCMH95, HCM96].

Looking for such a CA rule can, of course, also be modeled by searching a fitness landscape. In [DCMH95] CAs with a radius $r = 3$ were used, so there is a total of $2^{2^{3+1}} = 2^{128}$ possible update rules — too many to do an exhaustive search. These CA update rules can be represented as bit strings of length 128 (one bit for each lookup table entry), so the representation space consists of 2^{128} bit strings each of

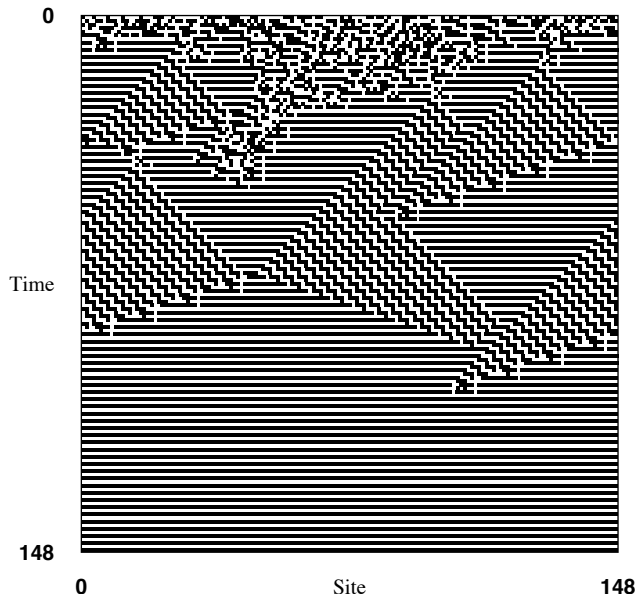


Figure 1: A space-time diagram, started from a random initial configuration of black and white cells, of a particular ($r = 3$) CA that solves the synchronization task. Global synchronization is reached roughly at time step 100. After [DCMH95].

length 128. In this representation a value of 0 denotes white and a 1 denotes black. The first bit in the string gives the new state for the neighborhood 0000000, the second bit gives the new state for the neighborhood 0000001, the next for 0000010, and so on until the last bit for the neighborhood 1111111.

The neighborhood relation of the landscape depends of course on the search method used. In [DCMH95] a genetic algorithm was used, which is a search algorithm modeled after natural evolution, using genetic operators called crossover and mutation. Here, we will look only at point-mutation, i.e. bit strings that differ only in the value of one bit are neighbors. So the fitness landscape is simply the boolean hypercube of dimension 128. The fitness of a particular CA rule is defined as the fraction of initial configurations out of a set of 100 randomly generated ICs on which it displays the desired synchronized behavior within a maximum number of time steps M , where M is a function of the lattice size L . So the fitness is normalized between 0 and 1. Here, as in [DCMH95], $L = 149$ and $M = 320$, and the ICs are randomly generated with a uniform distribution over the density of black cells.

So the problem of finding a CA that can solve the synchronization task can be modeled by a search on a particular fitness landscape. This landscape will be called the “synchronizing-CA landscape”. The next section gives the results of characterizing the structure of this landscape.

4 The Structure of the Synchronizing-CA Landscape

To gather statistics for characterizing the structure of the synchronizing-CA landscape 1,000 uphill walks on this landscape are performed. The algorithm for an uphill walk is as follows:

1. Start at a random point in the landscape and calculate its fitness (by iterating the CA that this point represents on 100 random ICs).
2. Calculate the fitness of all neighbors, or one-point mutants, of the current point, and record the ones that are fitter than the current point.
3. If there are no fitter neighbors then save the current point, which is a local optimum, and stop. Otherwise, go to 4.
4. Select one of the fitter neighbors at random and make that one the current point.
5. Go to step 2.

Whenever the fitness of a point is calculated, a new set of 100 random ICs is created, with a uniform distribution over the density of 1's in the IC. The same point can thus have a slightly different fitness value each time its fitness is calculated. Besides saving the local optima, during the walks several statistics like the number of fitter neighbors of the current point, the average fitness of those fitter neighbors, and the number of steps it took to reach the local optimum are saved. Together, these statistics give a nice characterization of the structure of the fitness landscape [Kau93].

Figure 2 shows a histogram of the fitness values of local optima. It shows that there are very many low-fitness (lower than 0.1) local optima, almost no local optima with a fitness between 0.1 and 0.6, and a small number of higher-fitness (higher than 0.6) local optima. Figure 3 shows the same plot with a much smaller y-scale to show the distribution of higher-fitness local optima more clearly.

Figure 4 shows the walk lengths, i.e., number of uphill steps, to the local optima against the fitness of the local optima. Clearly, low-fitness local optima are reached in a small number of steps, as shown by the cluster of points in the lower-left corner. Furthermore, there seems to be a clear correlation between the height of a low-fitness local optimum and the number of steps it takes to reach it. However, higher-fitness local optima can be reached in any number of steps ranging (roughly) from 5 to 30, as shown by the cloud of points at the right. There seems to be no correlation between the height of a higher-fitness local optimum and the number of steps it takes to reach it.

Figure 5 shows the number of fitter neighbors of a point along an uphill walk, against the fitness of that point. The plot shows all the points that were encountered

during all the 1,000 uphill walks. Points in the landscape with low fitness can have a number of fitter neighbors ranging (roughly) from 0 to 100 (remember that there are 128 neighbors in total for each point). This range becomes gradually smaller for increasing fitness — the higher the fitness of a point is, the smaller the number of fitter neighbors, on average, it will have. Note also that the density of points for smaller numbers of fitter neighbors is higher: having just a few fitter neighbors is more likely than having many fitter neighbors.

Figure 6 shows the average fitness of the fitter neighbors of a point along an uphill walk, against the fitness of that point. Again, all the points encountered during all the uphill walks are plotted. Similarly to the number of fitter neighbors, the range of the average fitness of fitter neighbors is wider for low-fitness points and becomes narrower for increasing fitness.

However, the result of Figure 5 does not necessarily mean that during a particular uphill walk the number of fitter neighbors decreases gradually. Figure 7 shows the number of fitter neighbors of the points along the longest uphill walk that was found (which reached a final fitness of 0.74 in 28 steps). As can be seen, at any step the number of fitter neighbors can be arbitrary, but of course within the range shown in Figure 5. There is no gradual decrease in the number of fitter neighbors, and no apparent correlation between the number of steps and the number of fitter neighbors.

Figure 8 shows the distribution of the locations of the local optima in the landscape. The fitness of a local optimum is plotted against its Hamming distance from the best CA rule, ϕ_{sync} , that was found for the synchronization task (see [DCMH95]). The Hamming distance between two bit strings is the number of bits in which the two strings differ in value. In other words, if the Hamming distance between two bit strings is d , it will take at least d bit flips, or point-mutations, to go from one point to the other. As the plot shows, the distances of the local optima from ϕ_{sync} are uniformly distributed within a certain range, and there is no correlation between the location of a local optimum and its height. The plot where the fitness of the local optima is plotted against their Hamming distance from the best local optimum found among all the uphill walks, looks similar.

As mentioned in section 2 another important part of the characterization of the structure of a fitness landscape is its correlation structure. Results on characterizing the correlation structure of the synchronizing-CA landscape are presented in another paper [Hor96]. The next section shows how the structure of a fitness landscape can explain certain aspects of an actual search on the landscape.

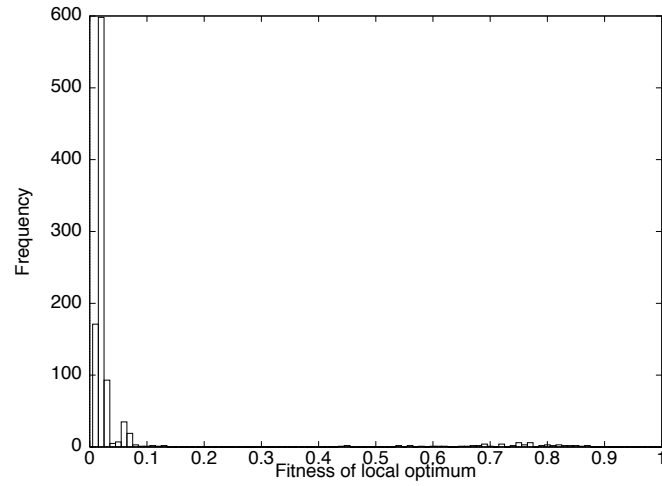


Figure 2: The number of local optima with a particular fitness.

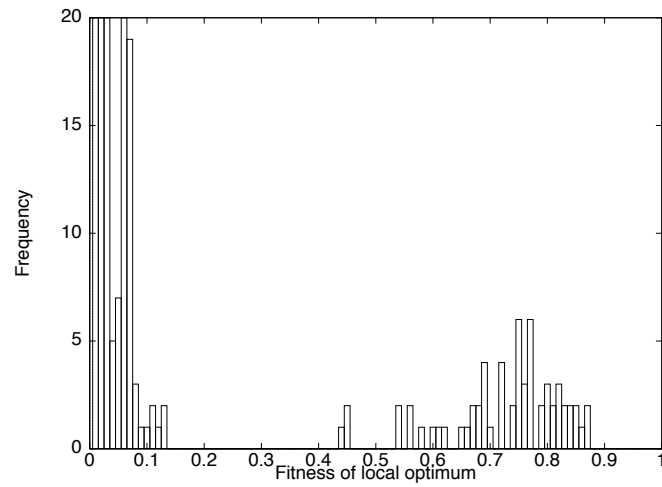


Figure 3: The number of local optima with a particular fitness with a smaller y-scale to show the distribution of higher-fitness optima more clearly.

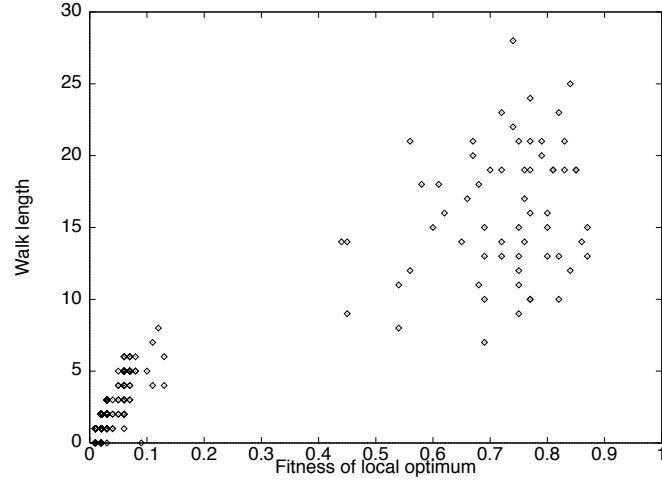


Figure 4: The walk length (number of steps) to reach a local optimum against the fitness of the local optimum.

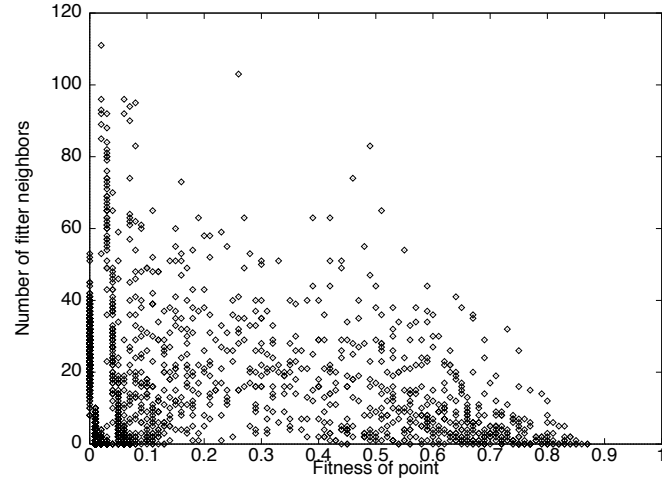


Figure 5: The number of fitter neighbors of a point in the landscape against the fitness of that point.

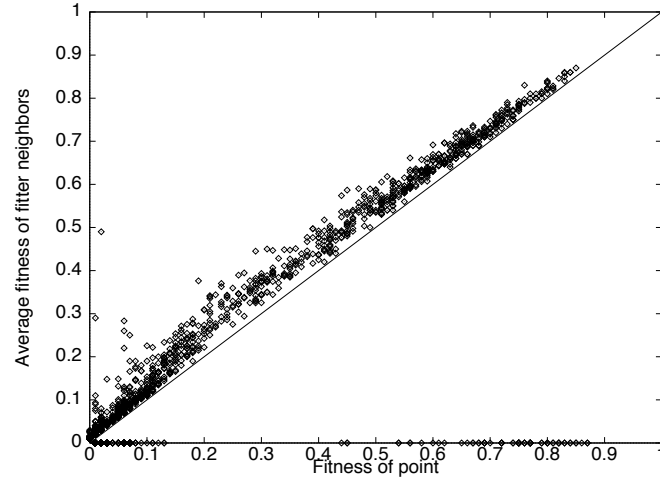


Figure 6: The average fitness of the fitter neighbors of a point in the landscape against the fitness of that point.

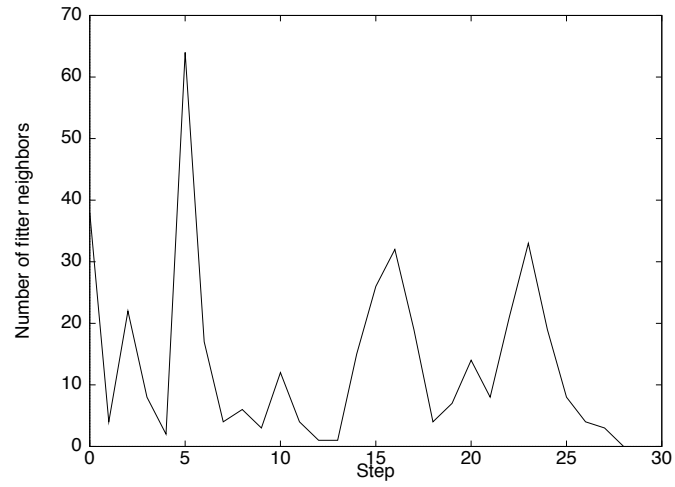


Figure 7: The number of fitter neighbors along one particular uphill walk.

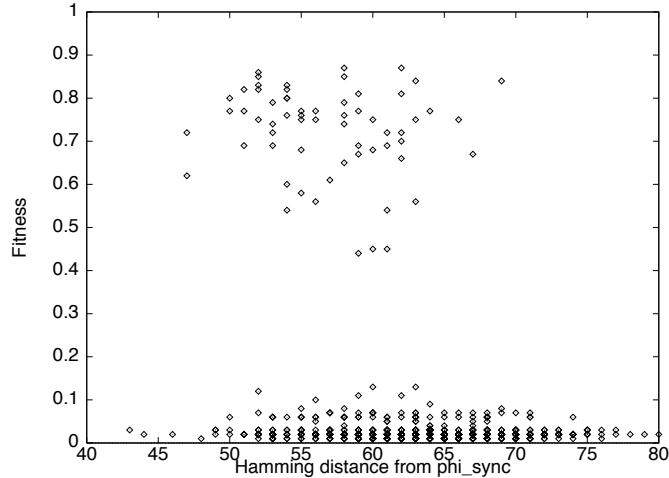


Figure 8: The fitness of the local optima against their Hamming distance from ϕ_{sync} , the best rule found for the synchronization task.

5 Landscape Structure and Search

What can the structure of a fitness landscape tell us about search on that landscape? The results presented in the previous section show that for the synchronizing-CA landscape there are many low-fitness local optima, a few higher-fitness local optima, and basically nothing in between (Figures 2 and 3). This means that once a search algorithm is able to escape from the low-fitness local optima, it can not get stuck on intermediate-fitness local optima. Figure 9 shows five different runs of applying a genetic algorithm (GA) to the problem of finding a CA rule for the synchronization task. The best fitness value in each generation is plotted over time. The particular GA that is used here uses only point-mutation (at each time step exactly one random bit is flipped in each string), and no crossover. This way, the search results can be related to the landscape structure, which is also derived by using point-mutation only.

As the plot shows, in all five runs the GA indeed jumps from low-fitness points (less than 0.1) to higher-fitness points (larger than 0.6) in just a few generations, and then gradually climbs toward a perfect-fitness of 1.0. The time to wait before this jump occurs is different for the different runs, but once the GA is able to escape the region of low-fitness peaks, it can not get stuck on intermediate-fitness local optima, and almost immediately jumps to regions of higher-fitness.

Looking closer at some of the local optima, it turns out that the low-fitness local optima are CA rules that do not show any interesting behavior (at least not in terms of the synchronization task), but happen to have the first and last bit in the bit string representing the CA lookup table, set to the correct value of 1 and

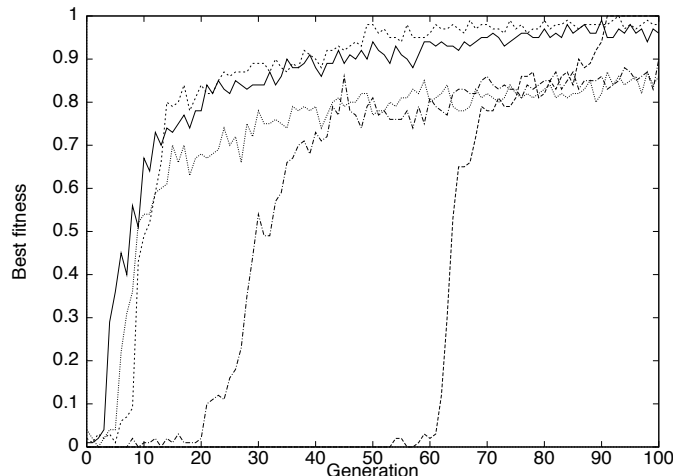


Figure 9: Five different runs of a genetic algorithm, using point-mutation and no crossover, applied to the synchronizing-CA problem. The best fitness in each generation is plotted over time.

0 respectively. Note that this is necessary for global synchronization, because an all 0's neighborhood (represented by the first bit in the string) has to go to a 1, and an all 1's neighborhood (represented by the last bit) has to go to a 0. Since the initial configurations are randomly distributed with a uniform distribution over the density of 1's, once in a while there is an IC with all 0's (white cells) or an IC with all 1's (black cells), on which those particular CA rules will synchronize immediately. If this happens twice for example (in the set of 100 ICs), then the fitness of such a CA rule will be 0.02.

The higher-fitness local optima, however, are actually applying a certain complicated strategy to solve the synchronization task (see [DCMH95, HCM96] for more details). They are not just relying on a particular advantageous initial configuration, but are more robust, and solve the task roughly 70 to 90 percent of the time. Figure 1 gives an example of a complicated strategy that solves the synchronization task on random initial configurations.

As it turns out, the GA first has to find the correct values for the first and the last bit in the bit string, and once it has found those values, it can start to try to improve the fitness values by looking for CA rules that actually apply some sort of strategy. This is reflected in Figure 9, where during the first generations the GA is more or less looking around in the region of the low-fitness local optima to get the values of the first and last bit set correctly in most of the members in the population. Once a small improvement on this is found, the GA quickly jumps up to higher fitness local optima, because it can not get stuck on intermediate-fitness local optima.

Some other conclusions can be drawn from Figures 5 and 6: points in the landscape with higher fitness values will, on average, have a smaller number of fitter neighbors, with a smaller average increase in fitness, than points in the landscape with lower fitness values. In other words, when the search is still in a region of low fitness, there are more opportunities to find fitter neighbors, and the increase in fitness will on average be larger, then when the search is already in regions of higher fitness. This, again, is reflected in Figure 9. As soon as the GA is able to escape the low-fitness local optima, large increases in fitness are made very rapidly, but later on the increases in fitness become smaller and smaller, and it takes longer before a new fitness increase is found.

However, as Figure 7 shows, the number of fitter neighbors along a particular walk can fluctuate a lot, and does not always decrease gradually. This could explain why in different runs of the GA there is a difference in the waiting time before the jump towards higher-fitness points occurs, and why the duration of this jump is also different.

Finally, from Figure 8 it can be concluded that local optima are randomly distributed throughout the landscape. They are not clustered together, and the location of one local optimum does not necessarily give information about the location of other local optima. However, this means that it does not matter very much where a search is started, because there will always be local optima relatively nearby, so only part of the landscape needs to be searched to find high peaks. As Figure 9 shows, in all five runs the GA was able to find high-fitness local optima.

6 Discussion

The goal of this paper is twofold. First, it presents results on characterizing the structure of a fitness landscape that results from searching for a complex system (a CA) that can perform a certain computational task called synchronization. As it turns out, the structure of this synchronizing-CA landscape is quite different from more “standard” fitness landscapes that have been looked at so far, like NK-landscapes [Kau93] or some combinatorial optimization problems landscapes [Kra95, Pre]. The synchronizing-CA landscape seems to be less regular, or less “well-behaved”. This makes the landscape harder to understand, but also more interesting, at least from a complex systems point of view.

Second, it is shown that this characterization of the structure of the landscape can help in explaining certain aspects of an actual search on that landscape. Characterizing the structure of the fitness landscape a complex system is living on can thus help in understanding that system, or make predictions about a search on the landscape. Therefore, it can also help in the control of complex non-linear systems.

Acknowledgments

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 Jim Crutchfield and Melanie Mitchell for support and for providing the opportunity to do this work. Also thanks to Catie Grasso for reading an earlier version of this paper.

References

- [DCMH95] 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. Morgan Kaufmann, 1995.
- [Gut90] H. A. Gutowitz. *Cellular Automata*. MIT Press, 1990.
- [HCM96] W. Hordijk, J. P. Crutchfield, and M. Mitchell. Embedded-Particle Computation in Evolved Cellular Automata, 1996. To appear in the pre-*Proceedings of Physics and Computation '96*.
- [Hor96] W. Hordijk. Correlation Analysis of the Synchronizing-CA Landscape, 1996. Submitted to *Proceedings of the CNLS Conference on Landscape Paradigms in Physics and Biology*.
- [Jon95] T. C. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, NM, March 1995.
- [Kau93] S. A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.
- [Kra95] B. Krakhofer. Local Optima in Landscapes of Combinatorial Optimization Problems. Master's thesis, University of Vienna, Vienna, Austria, January 1995.
- [MFH92] M. Mitchell, S. Forrest, and J. H. Holland. The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance. In F.J. Varela and P. Bourguin, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*. MIT Press, 1992.
- [MWS91] B. Manderick, M. de 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.

- [Pre] P. Preux. Personal communication.
- [Sta95] P. F. Stadler. Towards a theory of landscapes. In R. Lopés-Peña, R. Capovilla, R. García-Pelayo, H. Waelbroeck, and F. Zertuche, editors, *Complex Systems and Binary Networks*. Springer Verlag, 1995.
- [TM87] T. Toffoli and N. Margolus. *Cellular Automata Machines*. MIT Press, 1987.
- [Wol94] S. Wolfram. *Cellular Automata and Complexity*. Addison-Wesley, 1994.
- [Wri32] S. Wright. The roles of mutation, inbreeding, crossbreeding and selection in evolution. In D.F. Jones, editor, *Proceedings of the Sixth International Congress on Genetics*, volume 1, pages 356–366, 1932.