

University of Montana

ScholarWorks at University of Montana

Graduate Student Theses, Dissertations, &
Professional Papers

Graduate School

1995

Fitness landscapes investigated

Richard Kenneth Thompson
The University of Montana

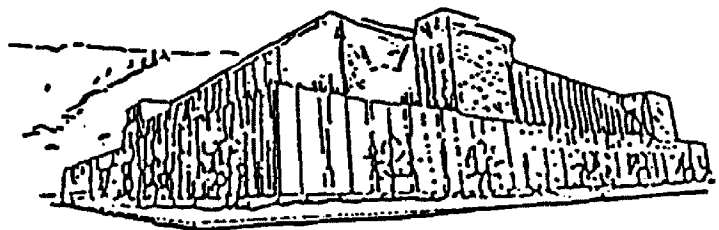
Follow this and additional works at: <https://scholarworks.umt.edu/etd>

Let us know how access to this document benefits you.

Recommended Citation

Thompson, Richard Kenneth, "Fitness landscapes investigated" (1995). *Graduate Student Theses, Dissertations, & Professional Papers*. 8352.
<https://scholarworks.umt.edu/etd/8352>

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.



Maureen and Mike
MANSFIELD LIBRARY

The University of **MONTANA**

Permission is granted by the author to reproduce this material in its entirety, provided that this material is used for scholarly purposes and is properly cited in published works and reports.

*** Please check "Yes" or "No" and provide signature ***

Yes, I grant permission

 X

No, I do not grant permission

Author's Signature Richard K. Thompson

Date July 25, 1995

Any copying for commercial purposes or financial gain may be undertaken only with the author's explicit consent.

Fitness Landscapes Investigated

by

Richard Kenneth Thompson

B.A. The University of Montana, 1991

presented in partial fulfillment of the requirements

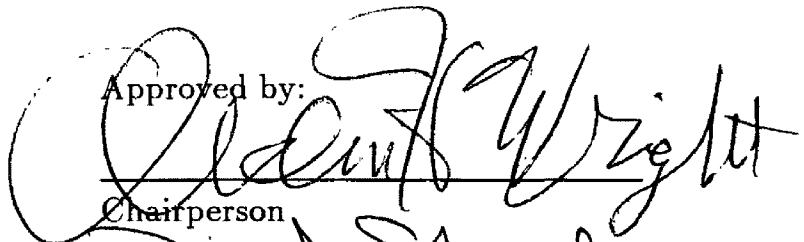
for the degree of

Master of Science

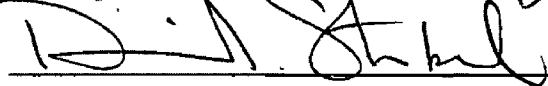
The University of Montana

July 1995

Approved by:

A large, stylized handwritten signature in black ink, appearing to read "Robert Wright", written over a horizontal line.

Chairperson

A handwritten signature in black ink, appearing to read "D. A. Stahl", written over a horizontal line.

Dean, Graduate School

July 26, 1995

Date

UMI Number: EP39153

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI EP39153

Published by ProQuest LLC (2013). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Fitness Landscapes Investigated (54 pp.)

Director: Alden H. Wright



Fitness landscapes and Boolean networks presented by Kauffman are studied in terms of their computational complexity. It is shown that if the epistatic influences for fitness contributions of the landscape are "localized," the optimal fitness can be found in polynomial time. If, however, the epistatic influences are from arbitrarily chosen positions of the string, the problem becomes NP-complete. These results are extended to show that the problem of finding pre-images in Boolean networks is NP-complete if the positions that affect the future of any given position of the string are arbitrarily chosen.

Contents

1	Dynamical Systems	1
1.1	Introduction	1
2	Fitness Landscapes	7
2.1	Definitions	7
2.1.1	The Organisms	8
2.1.2	Organism Fitness	8
3	Structured Behavior	13
3.1	Polynomial Time	13
4	Crossing the Border?	33
4.1	NP-completeness	33
5	Extensions	39
5.1	Boolean Networks Revisited	39
5.2	Satisfiability	41
6	Discussion	44
7	Conclusions	50
7.1	Conclusions	50

List of Figures

1.1	A circular N-tuple	4
3.1	Graph with two components.	25

Chapter 1

Dynamical Systems

1.1 Introduction

We consider dynamical systems over a finite state space. We assume that updates are synchronous and, hence, these systems are deterministic. Since their behavior at any given time is fully determined, once a previously attained state is reached the system will cycle through the same states indefinitely.

We can make formal the definition of those systems that we wish to consider. We let A be a set of discrete elements. For much of the discussion, $A = \{0, 1\}$, i.e. A represents the Boolean set. Then, we define the state of the system as an N -tuple $\mathbf{a} \in A^N$, i.e.

$$\mathbf{a} = (a_0, a_1, \dots, a_{N-1}),$$

$a_i \in A$ for $0 \leq i < N$. When we refer to the i^{th} position of \mathbf{a} , we are referring to a_i in the N -tuple \mathbf{a} . The dynamical behavior of the system is determined by a state transition function

$$F: A^N \longrightarrow A^N.$$

Given an initial state, $\mathbf{a}_0 \in A^N$, the state of the system at any time t is simply

$F^t(\mathbf{a}_0)$, where

$$F^t(\mathbf{a}) = F(F^{t-1}(\mathbf{a}))$$

and we define

$$F^0(\mathbf{a}) = \mathbf{a}, \quad \mathbf{a} \in A^N.$$

Due to the finiteness of A^N , there must exist a t_1 and t_2 with $t_1 \neq t_2$ such that

$$F^{t_1}(\mathbf{a}_0) = F^{t_2}(\mathbf{a}_0)$$

and

$$F^{t_1+m}(\mathbf{a}_0) = F^{t_2+m}(\mathbf{a}_0)$$

for all $m \in \mathcal{N}$.

We will add to this definition so that we may classify these systems. To enhance our definition, we will consider each position of the N -tuple \mathbf{a} independently. Usually, the number of positions of the N -tuple at time t that affect any given position of the N -tuple at time $t + 1$ is some constant K . Given that, we can define projection functions

$$p_i: A^N \longrightarrow A^K$$

for $0 \leq i < N$, which project an element of N space into K space. Each p_i is defined by a subset of $\{0, 1, \dots, N - 1\}$ of cardinality K . For example, if $N = 12$, $K = 3$ and p_0 is defined by $\{3, 5, 11\}$, then

$$p_0(\mathbf{a}) = (a_3, a_5, a_{11})$$

is a K -tuple in A^K . We also define for each position of the N -tuple the transition functions

$$f_i: A^K \longrightarrow A,$$

$0 \leq i < N$, which determine the setting of the i^{th} position of the N -tuple as we move through a state transition. These position transition functions are defined by a table;

for example, a position transition function would be defined by a table over A^K with values from A .

We can use the position projection functions and the position transition functions to redefine

$$F: A^N \longrightarrow A^N$$

as

$$F(\mathbf{a}) = (f_0(p_0(\mathbf{a})), f_1(p_1(\mathbf{a})), \dots, f_{N-1}(p_{N-1}(\mathbf{a}))), \quad \mathbf{a} \in A^N.$$

We will discuss four main classes of discrete dynamical systems,¹ and the behaviors with which each is associated. We begin with a discussion of random Boolean networks. As the name suggests, the random Boolean network is a discrete dynamical system based on $A = \{0, 1\}$. As mentioned previously, much of the work that has been done with random Boolean networks [2, 10] assumes that the number of positions of the N -tuple at time t that affect each position of the N -tuple at time $t + 1$ is some constant, K . Then, an N -bit Boolean string with K inputs to each Boolean function defined as above is called a random NK Boolean network [2, 10]. Essentially, this is the definition above, that is, there are no stipulations for the choice of any of the p_i 's or f_i 's other than those previously stated. The K positions that affect the future of any given position are randomly chosen, and the position transition function for that position is also randomly chosen from the 2^{2^K} Boolean functions over K bits.

When we think of this as a network, we can think of the p_i 's as the wiring of the network and the f_i 's as the functional aspect of the network. These are the essential elements of the networks that will differ in the following classes of discrete dynamical systems.

There are two main subclasses of networks under the random Boolean networks.

¹These four classes of discrete dynamical systems are those discussed in [10].

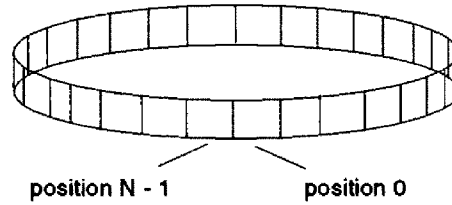


Figure 1.1: A circular N-tuple

Each of these will have structural requirements that are not present in the random networks. The first of these will have a homogeneous rule, but random wiring, that is, each of the positions will have the same Boolean function operating on randomly chosen inputs. Note that this network necessarily has the number of positions affecting the future of each position as a constant. In our chosen notation,

$$f_i = f_C: A^K \longrightarrow A, \quad 0 \leq i < N.$$

The second subclass has randomly chosen Boolean functions for each variable, but a homogeneous wiring template for each variable. Note, for homogeneous wiring to be applicable, we not only need to consider the set of variables as ordered, i.e. as an N -tuple, but also need to consider this N -tuple as circular as indicated in Figure 1.1. Furthermore, position indexing is to be considered $i \bmod N$. For example, if $i = N - 2$, then the position $i + 3$ is actually position 1, i.e.

$$\begin{aligned} (i + 3) \bmod N &= (N - 2 + 3) \bmod N \\ &= (N + 1) \bmod N \\ &= 1. \end{aligned}$$

In our chosen notation, this means that the positions chosen by p_i are dependent upon the value of i . For example, if $K = 3$, the wiring template for a network of this

subclass, p_i , could be defined from the subset $\{i - 1, i, i + 1\}$ and

$$p_i(\mathbf{a}) = (a_{i-1}, a_i, a_{i+1}) \in A^3.$$

Finally, we consider the last of our four classes, which has homogeneous Boolean functions and homogeneous wiring templates. One-dimensional cellular automata are discrete dynamical systems of this type. Note, that this class of systems is a subclass of both of the previous classes.

Much work has been done examining the complex or “edge of chaos” behavior of many of these systems [7, 10, 2, 3, 5]. In his book *The Origins of Order – Self-Organization and Selection in Evolution*, Stuart Kauffman discusses the effects of minimal perturbations on random NK Boolean networks. A minimal perturbation consists of arbitrarily changing the setting of one of the bits of the state vector. He finds that random Boolean networks with $K = 2$ are relatively stable with respect to minimal perturbations, that is, they almost always stay in the same basin of attraction in which they were before the perturbation. Furthermore, they are relatively insensitive to initial conditions, i.e. similar initial states tend to the same basin of attraction. Networks with $K \geq 3$, however, exhibit “chaotic” behavior with respect to minimal perturbations. They are also sensitive to initial conditions.

Kauffman also presents tunably rugged fitness landscapes. These landscapes are representations of fitness functions over a string of N bits. The fitness contribution of each position may be “influenced” by K other positions in the string. The ruggedness of the landscape can be increased by increasing the parameter K [2], much like the behavior of a Boolean network can be altered by changing the parameter K . The fitness of the entire string is just the average of the fitness contributions of each of the bits in the string, i.e.

$$F = \frac{1}{N} \sum_{i=1}^N f_i.$$

The *NK* Boolean networks and fitness landscapes presented by Kauffman appear to be a good model of many real biological processes. The Boolean networks have features much like the genetic regulatory networks which are apparently responsible for cell differentiation. Kauffman shows that the ratio of the number of cell types in an organism to the number of genes in that organism's genetic code is of the same order of magnitude as the ratio of the number of basins of attraction to the number of bits in a Boolean network. Kauffman uses the properties of populations adapting on rugged fitness landscapes to explain the radiation in early phylogenies in the evolutionary record. These properties also explain the stasis occurring later in the record. Kauffman also uses the structure of these landscapes to investigate how populations can best adapt toward the optimal fitness peaks.

The goal of this work is to determine the complexity involved in finding an optimally fit representative of the population with this type of fitness determination. We will see that as the size of the influencing string increases, so does the time complexity. The linear proximity of the influencing bits is also considered when determining the complexity of finding the optimally fit string. We will also investigate how linear proximity can affect the expected behavior of *NK* Boolean networks and how this can be exploited to reveal the underlying structure of the basins of attraction for these networks.

Chapter 2

Fitness Landscapes

2.1 Definitions

We will now discuss the computational complexity of finding the optimally fit member of some discrete, finite set of elements. The fitness of each member of the set will be determined by a fitness function resembling those presented in [2]. The model assumes that each position or gene in the string contributes to the overall fitness based on its setting and the settings of other genes with which it epistatically interacts. The major difference with the functions that we consider is that the fitness contributions are integer-valued. This is done to avoid the problems with complexity analysis for real number problems. We also remove the $1/N$ factor from the total resultant fitness.

Another consideration is the difference between “arbitrarily” versus “randomly” chosen fitness functions. Most of the literature on Boolean networks and fitness landscapes suggests that these networks and fitness functions are randomly generated. We will analyze arbitrarily chosen networks and fitness functions. It is assumed that an arbitrarily chosen fitness function is, in fact, the possible outcome of random generation. This is another way of saying that we are doing “worst case” analysis of the computational complexity.

When discussing fitness functions, it is natural to think of the members of the set as members of a population. We therefore call these members organisms.

2.1.1 The Organisms

First, we need a refined representation of the organisms to be investigated. Generally, the genotype of an organism is considered to be a string of genetic characters of some given length N . The possible values that each character of the string or N -tuple can assume are generally elements of some finite set A . We will call any string that has $|A|$ possible values per position an A -ary string. Let A^N denote the set of all A -ary strings of length N . We will denote an element of A^N by \mathbf{a} and note the

$$\mathbf{a} = (a_0, a_1, \dots, a_{N-1})$$

with each $a_i \in A$.

2.1.2 Organism Fitness

We must also have a clear definition of the fitness function that will be used to determine the fitness of the organism. Since the organism can be considered an element of A^N , we will consider the fitness function:

$$F: A^N \longrightarrow Z$$

where Z is the set of integers. In our analysis of the fitness function, we will consider the maximum fitness to be the best fitness and the minimum fitness to be the worst fitness, although this decision is arbitrary.

The definition of F will come from the NK model of epistatic interactions [2]. The idea is that each of the positions of the N -tuple makes a fitness contribution which depends on the value of that position as well as the $K - 1$ values of the other

positions of the string.¹ The parameter K then determines the amount of epistasis in the fitness function. Therefore, as in the NK Boolean networks, each position i has an associated projection function,

$$p_i: A^N \longrightarrow A^K,$$

that defines the positions of the string that affect the fitness contribution for position i . As with the NK Boolean networks, the p_i 's are defined by a subset of $\{0, 1, \dots, N-1\}$ of cardinality K . Since the p_i 's are linear transformations from A^N to A^K , the p_i 's are represented by $N \times K$ matrices, with each row vector being a unique basis vector of A^N , or simply as K -tuples of single-position projection functions over N . We will denote e_i as the basis vector with 1 in the i^{th} position and zeros everywhere else, or as the projection function over an element of A^N which chooses the i^{th} position of the N -tuple. For example, if $K = 3$, $N = 7$, and the fitness contribution of position 6 of the string depends upon positions 2 and 4, as well as position 6, then

$$\begin{aligned} p_6 &= \begin{bmatrix} e_2 \\ e_4 \\ e_6 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

and note that $p_6(\mathbf{a}) = (a_2, a_4, a_6) \in A^3$. These projection functions correspond to the wiring of the fitness network. We can also think of these projections in terms of a vector of projection functions

$$\mathbf{p}: A^N \longrightarrow (A^K)^N,$$

¹We use K as the total number of positions that affect the fitness contribution at any position, where Kauffman uses K as the number of additional positions that affect the fitness contribution at any given position.

where

$$\mathbf{p} = (p_0, p_1, \dots, p_{N-1}).$$

Also associated with each position is a fitness contribution function,

$$f_i: A^K \longrightarrow Z.$$

These fitness contribution functions correspond to the functional aspect of the fitness function. As suggested in [2], since we have no predetermined notion of what an optimally fit string should be, we choose these fitness contribution functions arbitrarily. As with the discrete dynamical systems of Chapter 1, these fitness contribution functions are simply defined by an $|A|^K$ look-up table of integers. We can, again, think of these functions together as a vector of fitness contribution functions

$$\mathbf{f}: (A^K)^N \longrightarrow Z^N,$$

where

$$\mathbf{f} = (f_0, f_1, \dots, f_{N-1}).$$

Then we can define

$$\mathbf{f} \circ \mathbf{p}(\mathbf{a}) = (f_0 \circ p_0(\mathbf{a}), f_1 \circ p_1(\mathbf{a}), \dots, f_{N-1} \circ p_{N-1}(\mathbf{a}))$$

for all $\mathbf{a} \in A^N$.

The total fitness of the string is then the sum of the fitness contributions of its parts, i.e.

$$F(\mathbf{a}) = \sum_{i=1}^N f_i(p_i(\mathbf{a})).$$

If we have the property that

$$f_i(\mathbf{b}) \geq 0, \quad \forall \mathbf{b} \in A^K, \quad 0 \leq i < N,$$

then we can define

$$F(\mathbf{a}) = \|\mathbf{f} \circ \mathbf{p}(\mathbf{a})\|_1$$

for all $\mathbf{a} \in A^N$, where $\|\cdot\|_1$ is the l_1 -norm.

We will call any system like that discussed above an NK fitness function. Notice that the choice of the fitness values for each of the f_i 's is arbitrary, but that the choice of the p_i 's can have a significant impact on the complexity of fitness computations, as we will see later.

Definition 2.1.1 *Let F be an NK fitness function. We say that F is a **random fitness function** if the $K - 1$ other positions that affect the fitness contribution of a position i are randomly chosen from the $N - 1$ other positions of the network.*

A random fitness function corresponds to the structure of an NK random Boolean network. For the following definitions, we will consider the string upon which the function is calculated to be circular, that indices are mod N , and let

$$r(i) = i \bmod N.$$

Definition 2.1.2 *Let F be an NK fitness network. We say that F is a **nearest-neighbor fitness function** if for each i ,*

$$p_i = \begin{bmatrix} e_{r(i)} \\ e_{r(i+1)} \\ \vdots \\ e_{r(i+K-1)} \end{bmatrix}$$

that is, the fitness contribution of position i depends on the $K - 1$ positions immediately to the right of position i .

Note, that this definition is slightly removed from the traditional nearest-neighbor definitions in that we use the K positions to the right of position i instead of $\frac{K}{2}$

positions on either side of position i . Since we are dealing with fitness functions, the organism fitness is invariant to the positioning of the projection function/fitness contribution pairs, so long as we keep the property that position i affects its own fitness contribution. This is easily seen by noting that a re-indexing of the components of \mathbf{p} and \mathbf{f} by mapping p_i to $p_{i+\frac{K}{2}}$ and mapping f_i to $f_{i+\frac{K}{2}}$ leaves us with fitness contributions that depend on the traditional nearest neighbors.

Nearest-neighbor fitness functions correspond to Boolean networks with homogeneous wiring but arbitrary functions for each position. The following fitness network is a less structured form of the nearest-neighbor network.

Definition 2.1.3 *Let F be an NK fitness function. We say that F is an η -neighborhood fitness function, $K < \eta$, if for each i , the $K - 1$ other positions that affect the fitness of position i are arbitrarily chosen from the $\eta - 1$ positions immediately to the right of position i , i.e.*

$$p_i = \begin{bmatrix} e_{r(j_1)} \\ e_{r(j_2)} \\ \vdots \\ e_{r(j_K)} \end{bmatrix}$$

where $j_1 = i$ and $i < j_a < i + \eta$ for $2 \leq a \leq K$.

Notice that, in the above definitions, the value of position i always affects the fitness contribution for position i . This criterion was kept for compatibility with the work presented in [2]. In the next chapter, we will attempt to analyze these different functions in terms of the difficulty of finding the optimal total fitness of the fitness landscape, and thus finding an optimally fit member of A^N .

Chapter 3

Structured Behavior

3.1 Polynomial Time

The analysis of fitness landscapes and the functions that yield them attempts to show how the time complexity of finding the optimally fit organism \mathbf{a} is closely related to K and the method of choosing \mathbf{p} . The goal is to determine whether the optimal fitness of any of the functions can be found in polynomial time. The following proofs show that as the random Boolean networks show a significant change in behavior between $K = 2$ and $K = 3$, i.e. the “edge of chaos,” so do their fitness function counterparts show a significant change in complexity between functions with influencing strings of length 2 versus those with influencing strings of length 3. As we cross the border between order and chaos, we also cross the border between polynomial time and NP-completeness.

In the following proofs of polynomial time performance, we will use the notion of the commutative diagram

$$\begin{array}{ccc} \textit{Problem}(N) & \longrightarrow & \textit{Solution}(N) \\ \downarrow & & \uparrow \\ \textit{Problem}(N - 1) & \longrightarrow & \textit{Solution}(N - 1); \end{array}$$

that is, given a problem of dimension N , we will show that we can produce in reasonable time a like problem of dimension $N - 1$. Likewise, we will show that the solution of this new problem can in reasonable time be used to provide the solution to the problem of dimension N . If each of these steps can be done in time of $O(p(N))$, where p is a polynomial, then we can find the solution to the original problem in $O(N * p(N)) = O(p'(N))$ polynomial time.

In discussing the time needed to produce a solution, we will make the following assumptions. These assumptions are based on the premise that any integer a can be efficiently represented by a binary string of length $\log_2 a$. First, we will assume that bitwise operations can be done in $O(1)$ time. This assumption yields that the addition of integers a and b can be done in $O(\log(\max\{a, b\}))$ time, since there are at most two times this many bitwise operations in one addition. This assumption also yields that comparisons between integers a and b can be done in the same time as addition. Finally, we will assume that the size of the result of addition of integers a and b will be $O(1 + \log(\max\{a, b\}))$, that is, the result of addition can be represented efficiently in a string that is at most one larger than the representation of the larger addend. This stems from the fact that at most one carry is generated from the last (leftmost) bitwise operation of an addition.

When discussing the solvability of a problem, we must consider the size of the input to the program that will solve the problem. We will refer to the maximum value of all of the fitness contributions as f_{max} , i.e.

$$f_{max} = \max_i \left\{ \max_{\mathbf{b} \in A^K} \{f_i(\mathbf{b})\} \right\}.$$

In the case of the NK fitness function, we see that we can represent the fitness contribution functions efficiently in $O(|A|^K N \log f_{max})$ space. The projection functions for each position can be efficiently represented in $O(KN \log N)$ space. When considering

nearest-neighbor fitness functions this space will not be considered, as we can represent all of the projection functions as a function of the position i . For η -neighborhood fitness functions, we can represent the projection functions in $O(KN \log \eta)$ space.

We will begin with a discussion of nearest-neighbor fitness functions with $K = 2$. In the following discussion, we will freely move elements between the sets A^N and A^{N-1} by ignoring one of the positions. This can be accomplished with a projection function

$$P_i^N: S^N \longrightarrow S^{N-1}$$

where S is any type of set. Then P_i is defined by

$$\begin{aligned} P_i^N(\mathbf{s}) &= P_i^N(s_0, \dots, s_{i-1}, s_i, s_{i+1}, \dots, s_{N-1}) \\ &= (s_0, \dots, s_{i-1}, s_{i+1}, \dots, s_{N-1}) \end{aligned}$$

for $\mathbf{s} \in S^N$. To move back to N space, we will use an injection function

$$I_i^N: S \times S^{N-1} \longrightarrow S^N$$

with I_i^N defined by

$$\begin{aligned} I_i^N(t, \mathbf{s}') &= I_i^N(t, (s'_0, \dots, s'_{i-1}, s'_i, \dots, s'_{N-1})) \\ &= (s'_0, \dots, s'_{i-1}, t, s'_i, \dots, s'_{N-1}) \end{aligned}$$

for all $t \in S$ and $\mathbf{s}' \in S^{N-1}$. Note that by definition

$$I_i^N(s_i, P_i^N(\mathbf{s})) = \mathbf{s}$$

and

$$P_i^N(I_i^N(t, \mathbf{s}')) = \mathbf{s}'$$

for all $t \in S$, $\mathbf{s} \in S^N$, and $\mathbf{s}' \in S^{N-1}$.

We will also use a replacement function

$$R_i^N: S \times S^N \longrightarrow S^N$$

with R_i defined by

$$\begin{aligned} R_i^N(t, \mathbf{s}) &= R_i^N(t, (s_0, \dots, s_{i-1}, s_i, s_{i+1}, \dots, s_{N-1})) \\ &= (s_0, \dots, s_{i-1}, t, s_{i+1}, \dots, s_{N-1}) \end{aligned}$$

for all $t \in S, \mathbf{s} \in S^N$.

Lemma 3.1.1 *Let F be an NK nearest-neighbor fitness function over the alphabet A with $K = 2$. Then we can construct F' , an $(N-1)K$ nearest-neighbor fitness function over the alphabet A with $K = 2$ in $O(|A|^3(1 + \log f_{max}))$ time with the properties:*

$$\max_{\mathbf{a} \in A^N} \{F(\mathbf{a})\} = \max_{\mathbf{a}' \in A^{N-1}} \{F'(\mathbf{a}')\}$$

and given \mathbf{a}' is an optimally fit member of A^{N-1} with respect to F' , we can find $v \in A$ such that

$$\mathbf{a} = I_i^N(v, \mathbf{a}')$$

is an optimally fit member of A^N with respect to F in $O(|A|(1 + \log f_{max}))$ time, for some $i, 0 \leq i < N - 1$.

Proof: Given F , defined by \mathbf{p} and \mathbf{f} , each position i of $\mathbf{a} \in A^N$ is chosen by only $K = 2$ projection functions, p_{i-1} and p_i , and therefore affects only $K = 2$ fitness contribution functions of F , namely f_{i-1} and f_i . Thus we choose position i and eliminate it and f_i from consideration.

Consider the nearest-neighbor vector of projection functions

$$\mathbf{p}': A^{N-1} \longrightarrow (A^2)^{N-1}$$

where

$$p'_j(\mathbf{a}') = (a'_j, a'_{j+1})$$

for $0 \leq j < N - 1$, and $\mathbf{a}' \in A^{N-1}$.

It is important to notice that given $\mathbf{a}' = P_i^N(\mathbf{a})$,

$$p'_j(\mathbf{a}') = \begin{cases} p_j(\mathbf{a}) & \text{for } 0 \leq j < i - 1, \\ (a'_{i-1}, a'_i) = (a_{i-1}, a_{i+1}) & \text{for } j = i - 1, \\ p_{j+1}(\mathbf{a}) & \text{for } i \leq j < N - 1. \end{cases}$$

Also consider the function

$$g: A^2 \longrightarrow Z$$

where

$$g(b, c) = \max_{d \in A} \{f_{i-1}(b, d) + f_i(d, c)\}$$

for all $b, c \in A$. Then we define

$$\mathbf{f}': (A^2)^{N-1} \longrightarrow Z^N$$

as

$$\mathbf{f}' = P_i^N(R_{i-1}^N(g, \mathbf{f})) \tag{3.1}$$

$$= P_i^N(f_0, \dots, f_{i-2}, g, f_i, \dots, f_{N-1}) \tag{3.2}$$

$$= (f_0, \dots, f_{i-2}, g, f_{i+1}, \dots, f_{N-1}) \tag{3.3}$$

that is, given i ,

$$f'_j = \begin{cases} f_j & \text{for } 0 \leq j < i - 1, \\ g & \text{for } j = i - 1, \\ f_{j+1} & \text{for } i \leq j < N - 1. \end{cases}$$

Note that

$$f'_j \circ p'_j(\mathbf{a}') = \begin{cases} f_j \circ p_j(\mathbf{a}) & \text{for } 0 \leq j < i - 1, \\ g(a'_{i-1}, a'_i) = g(a_{i-1}, a_{i+1}) & \text{for } j = i - 1, \\ f_{j+1} \circ p_{j+1}(\mathbf{a}) & \text{for } i \leq j < N - 1 \end{cases}$$

for all $\mathbf{a} \in A^N$, $\mathbf{a}' \in A^{N-1}$ with $\mathbf{a}' = P_i^N(\mathbf{a})$.

Notice that each of the $|A|^2$ elements that define the table for $f'_{i-1} = g$ takes $O(|A|(1 + \log f_{max}))$ time to compute. Therefore, the development of \mathbf{f}' can be done in $O(|A|^3(1 + \log f_{max}))$ time. Then the fitness function,

$$F': A^{N-1} \longrightarrow Z$$

is defined as

$$F'(\mathbf{a}') = \|\mathbf{f}' \circ \mathbf{p}'(\mathbf{a}')\|_1$$

for all $\mathbf{a}' \in A^{N-1}$.

Now we must show that the optimal fitness value from this reduced fitness function is the same as that of the original fitness function, and that we can, in $O(|A|(1 + \log f_{max}))$ time, use the solution to the problem of size $N - 1$ to provide a solution to the problem of size N .

Note that for any $\mathbf{a} \in A^N$, let $\mathbf{a}' = P_i^N(\mathbf{a})$. Then

$$\begin{aligned} F(\mathbf{a}) &= \|\mathbf{f} \circ \mathbf{p}(\mathbf{a})\|_1 \\ &= \sum_{j=0}^{N-1} f_j(p_j(\mathbf{a})) \\ &= f_{i-1}(p_{i-1}(\mathbf{a})) + f_i(p_i(\mathbf{a})) + \sum_{j=0}^{i-2} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-1} f_j(p_j(\mathbf{a})) \\ &= f_{i-1}((a_{i-1}, a_i)) + f_i((a_i, a_{i+1})) + \sum_{j=0}^{i-2} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-1} f_j(p_j(\mathbf{a})) \\ &\leq \max_{v \in A} \{f_{i-1}((a_{i-1}, v)) + f_i((v, a_{i+1}))\} + \sum_{j=0}^{i-2} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-1} f_j(p_j(\mathbf{a})) \\ &= g(a_{i-1}, a_{i+1}) + \sum_{j=0}^{i-2} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-1} f_j(p_j(\mathbf{a})) \\ &= f'_{i-1}(p'_{i-1}(\mathbf{a}')) + \sum_{j=0}^{i-2} f'_j(p'_j(\mathbf{a}')) + \sum_{j=i}^{N-2} f'_j(p'_j(\mathbf{a}')) \\ &= \sum_{j=0}^{N-2} f'_j(p'_j(\mathbf{a}')) \end{aligned}$$

$$\begin{aligned}
&= \|\mathbf{f}' \circ \mathbf{p}'(\mathbf{a}')\|_1 \\
&= F'(\mathbf{a}').
\end{aligned}$$

Furthermore, given $\mathbf{a}' \in A^{N-1}$ and the position i , we can use \mathbf{a}' to find $\mathbf{a} \in A^N$ with

$$F(\mathbf{a}) = F'(\mathbf{a}'),$$

i.e.

$$\mathbf{a} = I_i(v, \mathbf{a}'),$$

where $v \in A$ is chosen such that

$$f_{i-1}(a_{i-1}, v) + f_i(v, a_{i+1}) = f'_{i-1}(a_{i-1}, a_{i+1}).$$

Then

$$\begin{aligned}
F'(\mathbf{a}') &= \|\mathbf{f}' \circ \mathbf{p}'(\mathbf{a}')\|_1 \\
&= f'_{i-1}(p'_{i-1}(\mathbf{a}')) + \sum_{j=0}^{i-2} f'_j(p'_j(\mathbf{a}')) + \sum_{j=i}^{N-2} f'_j(p'_j(\mathbf{a}')) \\
&= f_{i-1}(a_{i-1}, a_i) + f_i(a_i, a_{i+1}) + \sum_{j=0}^{i-2} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-1} f_j(p_j(\mathbf{a})) \\
&= \sum_{j=0}^{N-1} f_j(p_j(\mathbf{a})) \\
&= \|\mathbf{f} \circ \mathbf{p}(\mathbf{a})\|_1 \\
&= F(\mathbf{a}).
\end{aligned}$$

Now let \mathbf{x}' be an optimally fit member of A^{N-1} with respect to F' . Then let $\mathbf{x} \in A^N$ be such that

$$F'(\mathbf{x}') = F(\mathbf{x})$$

by the argument above. Then,

$$\begin{aligned}
F'(\mathbf{x}') &= F(\mathbf{x}) \\
&\geq F'(\mathbf{y}) \text{ for all } \mathbf{y} \in A^{N-1} \\
&\geq F(I_i^{N-1}(b, \mathbf{y})) \text{ for all } \mathbf{y} \in A^{N-1}, b \in A.
\end{aligned}$$

Therefore \mathbf{x} is an optimally fit member of A^N with

$$F'(\mathbf{x}') = F(\mathbf{x}).$$

■

In Lemma 3.1.1, it was assumed that given an optimal solution to the problem of size $N - 1$ and the position i , we could refer to the original fitness function to determine an optimally fit member of the problem of size N . In the following proof we will maintain fitness functions at every level of iteration, so that we can provide not only the optimal fitness value, but also determine from this value an optimally fit member of the population.

Theorem 3.1.2 *Let F be an NK nearest-neighbor fitness function over the alphabet A , with $K = 2$. Then the optimal fitness value can be found in space and time of $O(|A|^3 N(1 + \log f_{max}))$.*

Proof: Without loss of generality, we assume $N = 2^m$, for some integer m . We iterate as follows: We can apply the process of position elimination from Lemma 3.1.1 to every other position of the fitness function, i.e. we have $\frac{N}{2}$ applications of the lemma, each taking $O(|A|^3(1 + \log f_{max}))$ space and time. Therefore, the time for iteration j is $O\left(\frac{N}{2^j}|A|^3(j + \log f_{max})\right)$, where $\frac{N}{2^j}$ represents the number of applications of the lemma and $j + \log f_{max}$ represents the increasing size of the integers involved in the computation. After m steps, we reach the point where we have only two positions left in the string; there are only $|A|^2$ members of the original set of strings left to consider, thus we can find an optimally fit string in $O(|A|^2(m + \log f_{max}))$ time. Using the facts

$$\sum_{i=1}^m \left(\frac{1}{2}\right)^i \leq 1$$

and

$$\sum_{i=1}^m \left(\frac{i}{2}\right)^i \leq 2,$$

and that multiplicative constants can be ignored when determining the order of the computation, the entire process takes

$$O\left(\sum_{i=1}^m \frac{N}{2^i} |A|^3 (i + \log f_{max})\right) = O(|A|^3 N (1 + \log f_{max}))$$

space and time. ■

We will next look at $K > 2$ fitness functions in which the epistatic interactions of the function are contained in a localized neighborhood. We will find that, from these too, we can determine the optimal fitness in polynomial time.

We will see that the difference between the $K = 2$ and $K > 2$ nearest-neighbor fitness functions is simply a change in the alphabet A . When discussing changing alphabets, we are really only talking about a base change in the underlying number system. For example, if the original alphabet is the Boolean or binary set $\{0, 1\}$ and $K = 4$, then the alphabet $A^{K-1} = A^3$ is just the octal alphabet, $\{0, 1, \dots, 7\}$ with the normal mapping

<i>binary</i> ³	<i>octal</i>
000	0
001	1
⋮	⋮
111	7.

Likewise, we can think of any $3n$ length binary string as an n length octal string.

Theorem 3.1.3 *Let F be an NK nearest-neighbor fitness function over the alphabet A . Then the optimal fitness value can be found in $O(|A|^{3(K-1)}Q(1 + \log(Kf_{max})))$ space and time, where $Q = \lceil N/(K-1) \rceil$.*

Proof: We will show that, with slight modifications, we can view the nearest-neighbor fitness function, F , over the alphabet A as a nearest-neighbor fitness function, F' ,

over an alphabet $A' = A^{(K-1)}$ with $K' = 2$. Then we can apply Theorem 3.1.2 to show that the optimal fitness can be found in $O(|A'|^3 N'(1 + \log f'_{max}))$ time.

For simplicity, we'll assume that $N = Q(K - 1)$, i.e. that N is evenly divisible by $K - 1$. Then A^N is equivalent to $(A')^Q$ and we can re-index positions and choice and fitness contribution functions as follows:

$$x_{j,k} = x_i \in A,$$

where $j = \lfloor i/(K - 1) \rfloor$ and $k = i \bmod (K - 1)$. Note that $0 \leq j < Q$, and $0 \leq k < K - 1$. We will use \mathbf{x}' to denote an element of $(A')^Q$. Then we have

$$\mathbf{x}'_j = (x_{j,0}, x_{j,1}, \dots, x_{j,K-2}) \in A'$$

for $0 \leq j < Q$. Note, that since F is a nearest-neighbor function, using $K - 1$ as the divisor of N gives us the property that projection functions reach exactly two major positions, i.e. $p_{j,k}$ can only choose positions from \mathbf{x}'_j and \mathbf{x}'_{j+1} for all $0 \leq k < K - 1$. Then we define

$$p'_j(\mathbf{x}') = (x'_{j,k}, x'_{j+1,k}),$$

where indices are taken mod Q , and note that there must exist

$$p''_{j,k}: (A')^2 \longrightarrow A^K$$

such that

$$p''_{j,k}(p'_j(\mathbf{x}')) = p_{j,k}(\mathbf{x}), \quad \forall 0 \leq k < K - 1, 0 \leq j < Q.$$

Then we define

$$f'_j(\mathbf{b}) = \sum_{k=0}^{K-2} f_{j,k}(p''_{j,k}(\mathbf{b}))$$

for all $\mathbf{b} \in (A^{(K-1)})^2$. Note that $f'_{max} \leq K f_{max}$. Then define

$$F'(\mathbf{x}') = \sum_{j=0}^{Q-1} f'_j(p'_j(\mathbf{x}')),$$

where

$$\begin{aligned}
F'(\mathbf{x}') &= \sum_{j=0}^{Q-1} f'_j(p'_j(\mathbf{x}')) \\
&= \sum_{j=0}^{Q-1} \sum_{k=0}^{K-2} f_{j,k}(p_{j,k}(\mathbf{x})) \\
&= \sum_{i=0}^{N-1} f_i(p_i(\mathbf{x})) \\
&= F(\mathbf{x})
\end{aligned}$$

for all $\mathbf{x} \in A^N = (A')^Q$, and

$$F' : (A')^Q \longrightarrow Z$$

is a nearest-neighbor fitness function over the alphabet $A' = A^{K-1}$ with $K' = 2$. F' was developed in $O(|A'|^2 Q(\log(K f_{max})))$ space and time. Then, by Theorem 3.1.2, the optimal fitness can be found in

$$O(|A'|^3 Q(1 + \log f'_{max})) = O(|A|^{3(K-1)} Q(1 + \log(K f_{max})))$$

time. ■

The next theorem deals with a slightly less structured form of the nearest-neighbor NK fitness function.

Theorem 3.1.4 *Let F be an NK η -neighborhood fitness function over the alphabet A . Then the optimal fitness value can be found in $O(|A|^{3(\eta-1)} Q(1 + \log(\eta f_{max})))$ space and time, where $Q = \lceil N/(\eta - 1) \rceil$.*

Proof: We will show that, with slight modifications, we can view the η -neighborhood fitness function over the alphabet A with $K < \eta$ as a nearest-neighbor fitness function over the alphabet A with $K' = \eta$. Then we can apply Theorem 3.1.3 to show that the optimal fitness can be found in $O(|A|^{3(\eta-1)} Q(1 + \log(\eta f_{max})))$ time.

Note that even though each contribution function, f_i , only explicitly depends on K elements of \mathbf{x} , we know that all of the positions of \mathbf{x} that affect f_i are contained in $(x_i, x_{i+1}, \dots, x_{i+\eta-1})$. We can therefore redefine p_i and f_i to make explicit the dependence on η positions by defining

$$p'_i(\mathbf{x}) = (x_i, x_{i+1}, \dots, x_{i+\eta-1})$$

and note that there must exist

$$p''_i: A^\eta \longrightarrow A^K$$

with

$$p''_i(p'_i(\mathbf{x})) = p_i(\mathbf{x})$$

for all $\mathbf{x} \in A^N$. Then define

$$f'_i(\mathbf{b}) = f_i(p''_i(\mathbf{b}))$$

for all $\mathbf{b} \in A^\eta$, and note that now each fitness contribution function, f'_i , is dependent on η positions of \mathbf{x} . Then

$$F'(\mathbf{x}) = \sum_{i=0}^{N-1} f'_i(p'_i(\mathbf{x})) = \sum_{i=0}^{N-1} f_i(p_i(\mathbf{x}))$$

is an A -ary nearest-neighbor fitness function with $K = \eta$. Then, by Theorem 3.1.3, the optimal fitness can be found in $O(|A|^{3(\eta-1)}Q(1 + \log(\eta f_{max})))$ space and time. ■

We now discuss a less structured form of the $K = 2$ fitness function. We maintain the restriction that each position affects its own fitness contribution, but put no criteria on the other position that affects this contribution.

We begin by looking at the graph defined by the system. Each position of the system defines a node of the graph, thus there are N nodes; each of the projection functions, p_i , defines an edge of the graph, thus there are N edges in the graph. Since we kept the criterion that each position affects its own fitness contribution function,

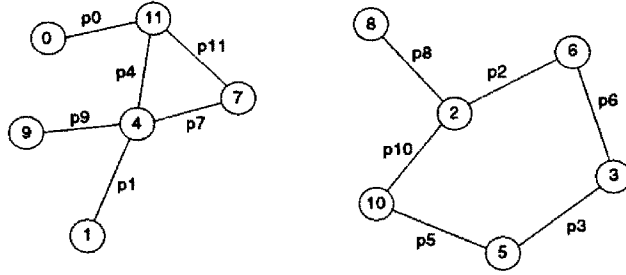


Figure 3.1: Graph with two components.

p_i chooses position i and one other position, j , of the system. Due to keeping this criterion, we can think of each node as having a trivial edge which starts and ends at that node, although this is unnecessary. We use p_i to define the edge connecting nodes j and i , and note that every node of the graph is contained in some non-trivial edge.

A quick observation of the graph yields the components of the graph, as seen in Figure 3.1. Given a node i of the graph, every node that is reachable from i , including i , belongs to one component of the graph. All nodes that are not reachable from i (and thus are not reachable from any node reachable from i) are independent in the sense that no fitness contribution function can be dependent upon nodes from two different components of the graph. Let n be the number of disjoint components of the graph, and label these sets of nodes S_i , $0 \leq i \leq n - 1$. Let $N_i = |S_i|$. Each component i of the original graph represents a connected graph with N_i nodes and N_i edges. Since this is true for all i , each of the components will contain exactly one cycle. Note that since $N = \sum N_i$, we have

$$A^N = \prod_{i=0}^{n-1} A^{N_i},$$

$$(A^K)^N = \prod_{i=0}^{n-1} (A^K)^{N_i},$$

and

$$R^N = \prod_{i=0}^{n-1} R^{N_i}.$$

We will abuse notation by letting \mathcal{P}_i denote all of the projections

$$\mathcal{P}_i: A^N \longrightarrow A^{N_i},$$

$$\mathcal{P}_i: (A^K)^N \longrightarrow (A^K)^{N_i},$$

and

$$\mathcal{P}_i: Z^N \longrightarrow Z^{N_i}$$

for each $0 \leq i < n$. Due to the fact that \mathbf{p} is used to determine the sets S_i , there must exist \mathbf{p}_i and \mathbf{f}_i for each $0 \leq i < n$ with the property

$$\begin{array}{ccc} & \mathbf{p} & \\ & \longrightarrow & \\ A^N & \longrightarrow & (A^K)^N \\ \mathcal{P}_i \downarrow & & \downarrow \mathcal{P}_i \\ A^{N_i} & \longrightarrow & (A^K)^{N_i} \\ & \mathbf{p}_i & \end{array}$$

and

$$\begin{array}{ccc} & \mathbf{f} & \\ & \longrightarrow & \\ (A^K)^N & \longrightarrow & Z^N \\ \mathcal{P}_i \downarrow & & \downarrow \mathcal{P}_i \\ (A^K)^{N_i} & \longrightarrow & Z^{N_i} \\ & \mathbf{f}_i & \end{array}$$

Then for $0 \leq i < n$, we define

$$F_i: A^{N_i} \longrightarrow Z$$

by

$$F_i(\mathcal{P}_i(\mathbf{a})) = |\mathbf{f}_i \circ \mathbf{p}_i(\mathcal{P}_i(\mathbf{a}))|.$$

Note that

$$F(\mathbf{a}) = \sum_{i=1}^n F_i(\mathcal{P}_i(\mathbf{a})).$$

Furthermore, since the F_i 's are independent,

$$\begin{aligned} \max_{\mathbf{a} \in A^N} \{F(\mathbf{a})\} &= \max_{\mathbf{a} \in A^N} \left\{ \sum_{i=0}^{n-1} F_i(\mathcal{P}_i(\mathbf{a})) \right\} \\ &= \sum_{i=0}^{n-1} \left(\max_{\mathbf{a} \in A^{N_i}} \{F_i(\mathcal{P}_i(\mathbf{a}))\} \right). \end{aligned}$$

Then if we can find the maximum fitness value for each of the independent fitness functions in $O(|A|^3 N_i (\log N_i + \log f_{max}))$ time, we can find the maximum fitness value for the entire system in

$$\begin{aligned} O\left(\sum_{i=0}^{n-1} |A|^3 N_i (1 + \log f_{max})\right) &= O\left(|A|^3 \sum_{i=0}^{n-1} N_i (1 + \log f_{max})\right) \\ &= O(|A|^3 N (1 + \log f_{max})) \end{aligned}$$

time.

Theorem 3.1.5 *Let F be a random NK fitness function over the alphabet A with $K = 2$ and the graph of F is connected. Then the optimal fitness value can be found in $O(|A|^3 N (1 + \log f_{max}))$ time.*

Proof: Assuming each position affects its own fitness contribution function, then the graph of F has N nodes and N edges. Since this graph is connected, it must have exactly one cycle. Then each position in a $K = 2$ fitness function must fall into one of three disjoint categories:

- It affects only one fitness contribution function;

- It affects exactly two fitness contribution functions;
- It affects more than two fitness contribution functions.

We begin by discussing the case where each position of the A -ary string affects exactly two fitness contribution functions of F . Since the graph of F is connected, each node acts as a link between two other nodes of the graph, and the graph is simply a circuit with every node of the graph on this circuit. Then the fitness function F is just a permutation of a nearest-neighbor fitness function with $K = 2$ and can be solved in $O(|A|^3 N(1 + \log f_{max}))$ time using the methods of Theorem 3.1.2.

We now discuss the case where at least one position of the A -ary string affects only one of the fitness contribution functions of F . We begin the reduction of such a fitness function by removing the dependence from those positions that only affect one of the fitness contributions. Without loss of generality, we assume that position $N - 1$ is a position that affects only one fitness contribution function, namely, f_{N-1} . Let the other position that affects f_{N-1} be position i , i.e.

$$p_{N-1}(\mathbf{a}) = (a_{N-1}, a_i).$$

We can maximize f_{N-1} with respect to position $N - 1$, thus making f_{N-1} dependent upon only one position, namely i , by defining g by

$$g: A \longrightarrow Z$$

with

$$g(b) = \max_{c \in A} \{f_{N-1}(c, b)\}$$

for all $b \in A$. This calculation takes $O(|A|(\log f_{max}))$ time for each element of the table defining g . Now g can be added appropriately to any fitness contribution function that depends upon position i , namely f_i . Let w be the other position that affects f_i ,

i.e.

$$p_i(\mathbf{a}) = (a_i, \mathbf{a}_w)$$

and define

$$g': A^2 \longrightarrow Z$$

as

$$g'(b, c) = g(b) + f_i(b, c)$$

for all $b, c \in A$. Note that the development of g' takes $O(|A|^3(1 + \log f_{max}))$ time, i.e. each of the $|A|^2$ elements of the table defining g' takes $O(|A|(1 + \log f_{max}))$ time to compute.

Then we define the vector of projection functions

$$\mathbf{p}' = P_{N-1}^N(\mathbf{p})$$

and note that

$$\mathbf{p}': A^{N-1} \longrightarrow (A^K)^{N-1}$$

and that

$$p'_j(P_{N-1}^N(\mathbf{a})) = p_j(\mathbf{a})$$

for $0 \leq j < N - 1$.

We then define

$$\mathbf{f}' = P_{N-1}^N(R_i^N(g', \mathbf{f})).$$

Again, notice that these definitions yield

$$f'_j \circ p'_j(\mathbf{a}') = \begin{cases} f_j \circ p_j(\mathbf{a}) & \text{for } 0 \leq j < N - 1; j \neq i, \\ g' \circ p_j(\mathbf{a}) & \text{for } j = i, \end{cases}$$

where $\mathbf{a}' = P_{N-1}^N(\mathbf{a})$. Then we define the fitness function F' as

$$F'(\mathbf{a}') = \sum_{j=0}^{N-2} f'_j(p'_j(\mathbf{a}'))$$

for all $\mathbf{a}' \in A^{N-1}$.

Note that for any $\mathbf{a} \in A^N$

$$\begin{aligned}
F(\mathbf{a}) &= \|\mathbf{f} \circ \mathbf{p}(\mathbf{a})\|_1 \\
&= \sum_{j=0}^{N-1} f_j(p_j(\mathbf{a})) \\
&= f_i(p_i(\mathbf{a})) + f_{N-1}(p_{N-1}(\mathbf{a})) + \sum_{j=0}^{i-1} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-2} f_j(p_j(\mathbf{a})) \\
&= f_i((a_i, a_n)) + f_{N-1}((a_{N-1}, a_i)) + \sum_{j=0}^{i-1} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-2} f_j(p_j(\mathbf{a})) \\
&\leq f_i((a_i, a_n)) + \max_{b \in A} \{f_{N-1}((b, a_i))\} + \sum_{j=0}^{i-1} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-2} f_j(p_j(\mathbf{a})) \\
&= f_i((a_i, a_n)) + g(a_i) + \sum_{j=0}^{i-1} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-2} f_j(p_j(\mathbf{a})) \\
&= g'((a_i, a_n)) + \sum_{j=0}^{i-1} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-2} f_j(p_j(\mathbf{a})) \\
&= f'_i(p'_i(\mathbf{a}')) + \sum_{j=0}^{i-1} f'_j(p'_j(\mathbf{a}')) + \sum_{j=i+1}^{N-2} f'_j(p'_j(\mathbf{a}')) \\
&= \sum_{j=0}^{N-2} f'_j(p'_j(\mathbf{a}')) \\
&= \|\mathbf{f}' \circ \mathbf{p}'(\mathbf{a}')\|_1 \\
&= F'(\mathbf{a}').
\end{aligned}$$

Furthermore, note given $\mathbf{a}' \in A^{N-1}$, there must exist $\mathbf{a} \in A^N$ with

$$F(\mathbf{a}) = F'(\mathbf{a}'),$$

i.e.

$$\mathbf{a} = I_{N-1}(v, \mathbf{a}'),$$

where $v \in A$ such that

$$f_{N-1}(v, a_i) + f_i(a_i, a_n) = f'_i(a_i, a_n).$$

Then

$$\begin{aligned}
F'(\mathbf{a}') &= \|\mathbf{f}' \circ \mathbf{p}'(\mathbf{a}')\|_1 \\
&= f'_i(p'_i(\mathbf{a}')) + \sum_{j=0}^{i-1} f'_j(p'_j(\mathbf{a}')) + \sum_{j=i+1}^{N-2} f'_j(p'_j(\mathbf{a}')) \\
&= f_i(a_i, a_n) + f_{N-1}(a_{N-1}, a_i) + \sum_{j=0}^{i-1} f_j(p_j(\mathbf{a})) + \sum_{j=i+1}^{N-2} f_j(p_j(\mathbf{a})) \\
&= \sum_{j=0}^{N-1} f_j(p_j(\mathbf{a})) \\
&= \|\mathbf{f} \circ \mathbf{p}(\mathbf{a})\|_1 \\
&= F(\mathbf{a}).
\end{aligned}$$

Again, as in Lemma 3.1.1, this implies that

$$\max_{\mathbf{a} \in A^N} \{F(\mathbf{a})\} = \max_{\mathbf{a}' \in A^{N-1}} \{F'(\mathbf{a}')\}.$$

Notice that the categorization of position j may have changed in F' since now j affects one less fitness contribution function, namely, f_{N-1} . We continue by eliminating those positions and their corresponding fitness contribution functions that only affect one contribution function, each step taking $O(|A|^3(1 + \log f_{max}))$ time. When we can find no more positions that affect only one contribution function, all of the remaining positions must affect exactly two fitness contribution functions. Then we are again left with a permutation of a $K = 2$ nearest-neighbor fitness function, which can be solved by the method of Theorem 3.1.2. ■

As we can see, when we keep the condition that each position of the string affects its own fitness contribution and the length of the influencing strings is two or less, the optimally fit organism can be found in polynomial time. We also have the situation that, if the epistatic interaction is contained within a localized neighborhood, the optimally fit organism can be found in polynomial time. In the next chapter, we investigate $K = 2$ fitness functions that do not have the property that each position

affects its own fitness contribution. We will see that that finding the optimal fitness of these functions, as well as any random fitness functions with $K \geq 3$, is NP-complete.

Chapter 4

Crossing the Border?

4.1 NP-completeness

In this chapter we will discuss two problems that we could not solve with means used in the previous chapter. The problems discussed here are those of the random NK fitness functions with $K = 2$, where positions don't necessarily affect their own fitness contribution, and those with $K = 3$. We will find that the problem of finding the optimal fitness for each of these problems, and thus an optimally fit member of the population, is NP-complete.¹

Before we begin, a discussion of the problem of SATISFIABILITY (SAT) is in order, as well as a general approach to proving NP-completeness. SATISFIABILITY is a problem from Boolean logic, which is described as follows:²

Let $U = \{u_1, u_2, \dots, u_m\}$ be a set of Boolean *variables*. A *truth assignment* for U is a function $t: U \rightarrow \{\mathcal{T}, \mathcal{F}\}$. If $t(u) = \mathcal{T}$ we say that u is “true” under t ; if $t(u) = \mathcal{F}$ we say that u is “false.” If u is a variable in U , then u and \bar{u} are *literals* over U . The literal u is true under t if and only if the variable u is true under t ; the literal \bar{u} is

¹See [1] for an indepth discussion of NP-completeness.

²The discussion of SATISFIABILITY is taken almost entirely from [1].

true if and only if the variable u is false.

A *clause* over U is a set of literals over U , such as $\{u_1, \bar{u}_3, u_8\}$. A clause is *satisfied* by a truth assignment if and only if at least one of its members is true under that assignment, thus a clause represents the disjunction of its literals. The clause above will be satisfied by t unless $t(u_1) = \mathcal{F}$, $t(u_3) = \mathcal{T}$, and $t(u_8) = \mathcal{F}$. A collection C of clauses over U is *satisfiable* if and only if there exists some truth assignment for U that simultaneously satisfies all the clauses in C . Such a truth assignment is called a *satisfying truth assignment* for C . The SATISFIABILITY problem is specified as follows:

SATISFIABILITY

INSTANCE: A set U of variables and a collection C of clauses over U .

QUESTION: Is there a satisfying truth assignment for C ?

The problem remains NP-complete even if each $c \in C$ satisfies $|c| = 3$ and, for each $u \in U$, there are at most 3 clauses in C that contain either u or \bar{u} [1].

The problem of SATISFIABILITY was proved to belong to the class NP of decision problems that can be solved in polynomial time by a nondeterministic computer by Stephen Cook in 1971 [1].

To prove NP-completeness, we must first phrase the NK fitness problem as a decision problem, show that this problem is an element of NP, and then find a polynomial transformation from a known NP-complete problem to the NK fitness problem. We state the NK decision problem as follows:

MAXIMUM ARBITRARY NK FITNESS with $K = 2$

INSTANCE: An NK fitness function F with $K = 2$ inputs for each of N fitness contributions, where both inputs are arbitrarily chosen, and positive integer T .

QUESTION: Is there an element $\mathbf{a} \in A^N$ with $F(\mathbf{a}) \geq T$?

If we can answer this question in polynomial $P(N)$ time and we have an upperbound B on the value of the fitness, we can then find the optimal fitness in $O(P(N) \log B)$ by testing only $\log B$ of the numbers less than or equal to B using methods similar to those of a binary search. In this case, the upperbound

$$B = N f_{max}$$

can be found in $O(A^K N)$ time and is $O(NS)$ size where S is the size of the input problem. We will see that the decision problem as stated is NP-complete.

The NP-complete problem that we will transform is a form of the SATISFIABILITY problem stated as follows:

MAXIMUM 2-SATISFIABILITY

INSTANCE: A set U of variables and a collection C of clauses over U , such that each $c \in C$ has $|c| = 2$, and positive integer $T \leq |C|$.

QUESTION: Is there a truth assignment for U that simultaneously satisfies at least T of the clauses of C ? [1]

From this instance of an NP-complete problem, we will construct a random NK fitness function over the character set $A = \{0, 1\}$ with $N = \max\{|U|, |C|\}$ and $K = 2$. A solution to the fitness function decision problem would provide a solution to the MAXIMUM 2-SATISFIABILITY problem.

Theorem 4.1.1 *Without loss of generality, assume that $A = \{0, 1\}$. The MAXIMUM ARBITRARY NK FITNESS with $K = 2$ problem is NP-complete.*

Proof: First, we must show that $F \in \text{NP}$. We need to show that given $\bar{a} \in A^N$ and $T \in Z$, we can verify whether $F(\bar{a}) \geq T$ in polynomial time. Given \bar{a} , the computation of $F(\bar{a})$ consists of the addition of N table lookups, which can clearly be done in polynomial time.

We must now map the instance of the previously mentioned SATISFIABILITY problem to a random NK fitness function with $K = 2$. Let (U, C, T) be an instance of MAXIMUM 2-SATISFIABILITY.

We define $N = \max\{|U|, |C|\}$, and let the first $|U|$ positions of the string represent the elements of U . We begin by arbitrarily assigning the clauses of C to the positions of the string. If $|U| > |C|$, we assign the false Boolean function of two inputs to the positions that are not assigned a clause of C and arbitrarily pick two inputs for these positions.

Next, we define p_i to select those positions of the string that affect the clause assigned to position i , and define

$$f_0(\mathbf{b}) = \begin{cases} 1 & \text{if } \mathbf{b} \text{ satisfies the clause assigned to position } i \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{b} \in 2^2$. Finally, we define

$$F(\mathbf{a}) = \sum_{i=0}^{N-1} f_i(p_i(\mathbf{a})).$$

Now we must show that $F(\mathbf{a}) \geq T$ if and only if at least T clauses of C are simultaneously satisfiable. First, let \mathbf{a} be a string such that the fitness $F(\mathbf{a}) \geq T$. Then, since

$$F(\mathbf{a}) = \sum_{i=1}^N f_i(p_i(\mathbf{a})) \geq T$$

and each f_i can add at most 1 to the total fitness, there must be at least T contribution functions that are of value 1. But the only f_i 's that can contribute to the total fitness are those that were matched to clauses of C . Since the total fitness is greater than or equal to T , then at least T of those f_i 's must have contributed, which implies that at least T of the clauses of C must be true. This implies that the first $|U|$ elements of the bit string \mathbf{a} represent a truth assignment of U that simultaneously satisfies at least T of the clauses of C .

Now let \mathbf{a} be a truth assignment of U such that at least T clauses of C are satisfied. Then let $\mathbf{b} \in 2^N$, such that the first $|U|$ elements of \mathbf{b} are equal to \mathbf{a} . Then, by the definition of F , each of the positions that is assigned a satisfied clause of C contributes 1 to the total fitness of \mathbf{b} . Then, if at least T clauses of C are satisfied, at least T of the fitness contributions are 1, and since none detract from the total fitness,

$$F(\mathbf{a}) \geq T.$$

All that remains is to show that the transformation was done in polynomial time, which is straightforward, since each p_i and f_i was determined locally with the information from the clause of C assigned to position i . ■

We can use the results of the proof of the NP-completeness of the MAXIMUM ARBITRARY NK FITNESS with $K = 2$ problem to show that the arbitrary NK fitness function optimization problem with $K \geq 3$ is NP-complete, even with the additional restriction that each position affects its own fitness contribution. We begin by stating the $K = 3$ problem as a decision problem.

MAXIMUM ARBITRARY NK FITNESS with $K = 3$

INSTANCE: An NK fitness function with $K = 3$ inputs for each on N fitness contributions, where each position affects its own fitness contribution and the other two inputs are arbitrarily chosen, and positive integer T .

QUESTION: Is there an element $\mathbf{a} \in A^N$ with $F(\mathbf{a}) \geq T$?

The result is presented in the following theorem.

Theorem 4.1.2 *The MAXIMUM ARBITRARY NK FITNESS with $K = 3$ problem is NP-complete.*

Proof: Again, we must first show that $F \in \text{NP}$. We need to show that given $\bar{\mathbf{a}} \in A^N$ and $T \in \mathbb{Z}$, we can verify whether $F(\bar{\mathbf{a}}) \geq T$ in polynomial time. As in the $K = 2$

case, given $\bar{\mathbf{a}}$, the computation of $F(\bar{\mathbf{a}})$ consists of the addition of N table look-ups, which clearly can be done in polynomial time.

Now we will map an instance of the MAXIMUM ARBITRARY NK FITNESS problem with $K = 2$ to the $K = 3$ problem. Let $(F, \mathbf{f}, \mathbf{p}, T)$ be an instance of the $K = 2$ problem. All we really need to do to turn this instance into the $K = 3$ problem is to add dependence on position i for each contribution function, f_i .

Let

$$p_i(\mathbf{a}) = (a_{i_1}, a_{i_2}),$$

that is, i_1 and i_2 represent the positions that affect fitness contribution f_i . We define

$$p'_i(\mathbf{a}) = (a_i, a_{i_1}, a_{i_2})$$

as the projection function that selects position i as well as those positions that affect f_i . Then we define

$$f'_i(p'_i(\mathbf{a})) = f_i(p_i(\mathbf{a}))$$

for all positions i and

$$F'(\mathbf{a}) = \sum_{i=0}^{N-1} f'_i(p'_i(\mathbf{a})).$$

Clearly $F'(\mathbf{a}) = F(\mathbf{a})$ for all $\mathbf{a} \in A^N$, therefore $F'(\mathbf{a}) \geq T$ if and only if $F(\mathbf{a}) \geq T$. Also, it is clear that this transformation is done in polynomial time, since the development of \mathbf{p}' and \mathbf{f}' were simple modifications of \mathbf{p} and \mathbf{f} . ■

In the next chapter, we will extend some of the findings discussed in Chapters 3 and 4 to include other systems of interest, namely, some of the complexity involved with the original Boolean networks, and extensions of our understanding of the SATISFIABILITY problem in general.

Chapter 5

Extensions

We now extend our results to include other systems of interest. In Chapter 3, we found that, if the NK fitness function had locality in the way that \mathbf{p} chose the elements of \mathbf{a} , we could find the optimal fitness in polynomial time. In Chapter 4, we saw this was not the case for fitness functions that were affected by randomly chosen elements of \mathbf{a} . We will extend both of these results.

5.1 Boolean Networks Revisited

We begin by re-examining the *NK* Boolean networks discussed in Chapter 1. One of the means for studying these networks is in terms of their basins of attraction. One way of finding the basins of attraction is to construct a map showing all of the state transitions of the system. To generate this map one must, for each state, know all of the pre-images. An algorithm has been presented [10] that has, on average, better performance than a total state space search. Here, we attempt to analyze this problem in more detail.

The question we propose to answer is whether a pre-image for a given state of the system exists. Following Wuensche, we call a state that has no pre-images a

garden-of-eden state[10]. We state the problem as follows:

GARDEN-OF-EDEN STATE with $K = 3$

INSTANCE: An NK arbitrary Boolean network with $K = 3$ and $\mathbf{s} \in \{0, 1\}^N$.

QUESTION: Is \mathbf{s} a garden-of-eden state?

We can easily see this problem is NP-complete if $K \geq 3$. All one need realize is the fact that 3-SAT problems are actually a subset of the random NK Boolean networks with $K = 3$. The 3-SAT problem is stated as follows:

3-SATISFIABILITY (3-SAT)

INSTANCE: A set U of variables, collection C of clauses over U such that each clause $c \in C$ has $|c| = 3$.

QUESTION: Is there a satisfying truth assignment for C ? [1]

We present the result in the following corollary.

Corollary 5.1.1 *The GARDEN-OF-EDEN STATE problem is NP-complete.*

Proof: First, it is obvious that the problem is in NP, for we could guess a pre-image, \mathbf{a} , and check in polynomial time whether \mathbf{a} is a pre-image of \mathbf{s} .

Now, we must transform a known NP-complete problem to the GARDEN-OF-EDEN STATE problem. We will transform an instance of the 3-SAT to this problem. Let (U, C) be an instance of 3-SAT. Let $N = \max\{|U|, |C|\}$. Since in random Boolean networks we make no stipulation that each position affect its own future, we arbitrarily assign the clauses of C to the positions of the network. Let the variables of U correspond to the first $|U|$ elements of the network. If we have positions of the network that have no clauses matched to them, we randomly assign three positions that will affect the future of each of these positions and assign them to the TRUE Boolean function of three variables.

Now we define the components of the NK Boolean network as follows: For each position, i , we define

$$p_i(\mathbf{s}) = (s_{i_1}, s_{i_2}, s_{i_3}),$$

where i_1 , i_2 , and i_3 are the positions that represent that variables of U that affect the clause of C that has been assigned to position i . Since all of the clauses are Boolean functions, the assignment of \mathbf{f} is simply

$$f_i(p_i(\mathbf{s})) = c_j,$$

where c_j is the clause that is assigned to position i .

Then we ask, is the state $\mathbf{1}$ (the state with all ones) a garden-of-eden state?

We must now show that the state $\mathbf{1}$ has a pre-image if and only if the clauses of C are satisfiable. Let \mathbf{a} be a pre-image of $\mathbf{1}$. Then clearly all of the clauses of C are satisfied, so the first $|U|$ elements of \mathbf{a} represent a satisfying truth assignment of U for C .

Now let \mathbf{b} be a satisfying truth assignment of U for the clauses of C . Then all of the positions of the network that are matched to clauses of C will be 1, and all of the rest of the positions always return 1, the state $\mathbf{1}$ has a pre-image and, thus, is not a garden-of-eden state. A pre-image is represented by the first $|U|$ positions being set to the values represented by \mathbf{b} and the rest of the positions arbitrarily set.

All that remains is to show that the transformation from 3-SAT was done in polynomial time which, again, is straightforward since \mathbf{p} and \mathbf{f} were developed with information obtained locally from the clause assigned to each position i . ■

5.2 Satisfiability

Another extension of the work of Chapters 3 and 4 is in the solution to the problem of SATISFIABILITY if some locality structure exists in the problem. The structure

necessary is stated in the following corollary.

Corollary 5.2.1 *Let S be an instance of the SATISFIABILITY problem, and assume that the set U of Boolean variables be indexed, i.e.*

$$U = \{u_0, u_1, \dots, u_{n-1}\}.$$

Let $C = \{c_0, c_1, \dots, c_{m-1}\}$ be the set of clauses over U and k be a constant integer. Suppose that for each clause c_i , there is an integer j_i so that the literals of c_i are taken from the set of variables $\{u_{j_i} \text{ to } u_{j_i+k}\}$, where the indices are taken mod n .

Then, if k is considered to be constant, the question of whether a satisfying truth assignment of U exists for the clauses of C can be answered in time which is polynomial in N .

Proof: To prove this corollary, we will develop an NK nearest-neighbor fitness function, F , with the property that

$$\max_{\mathbf{a} \in \{0,1\}^N} \{F(\mathbf{a})\} \geq N$$

if and only if the clauses of C are satisfiable. Since the maximum fitness can be found in $O(A^{3(K-1)}N)$ time, we can answer the satisfiability problem in polynomial time.

We begin by partitioning C into N sets

$$C_i = \{c_j : c_j \text{ is dependent on variables chosen from } u_i, \dots, u_{i+K}\}.$$

We can construct these C_i 's to be disjoint, i.e. even though some clauses could fit the description of more than one C_i ; once a clause is in one subset it is unavailable to other subsets.

Then, for each position we define the fitness contribution function

$$f_i(\mathbf{b}) = \begin{cases} 1 & \text{if } C_i = \emptyset, \text{ or if } \mathbf{b} \text{ satisfies the clauses of } C_i \\ 0 & \text{otherwise} \end{cases}$$

for all $\mathbf{b} \in \{0, 1\}^K$. Note that each of the 2^K elements of f_i can be determined in time linear in $|C_i|$. Therefore, the entire function set, \mathbf{f} , can be determined in $O(2^K N|C|)$ time.

All that is left is to show that a state \mathbf{s} of the variables of U is a satisfying truth assignment of the clauses of C if and only if

$$F(\mathbf{s}) \geq N,$$

which is straightforward from the definition of the fitness contribution functions, f_i .

Let \mathbf{s} represent a satisfying truth assignment of the variables of U for the clauses of C . Then for each i , either $C_i = \emptyset$, or the clauses of C_i are satisfied by \mathbf{s} . In either case, f_i will return 1 for all $0 \leq i < N$, which implies that

$$F(\mathbf{s}) = N.$$

Now let \mathbf{s} represent a state such that $F(\mathbf{s}) = N$. Then, since each f_i can contribute at most 1 to the total fitness, all of these contribution functions must contribute 1. This implies that either the clauses of C_i are satisfiable or $C_i = \emptyset$ for all i , which implies that \mathbf{s} represents a satisfying truth assignment of the variables of U for the clauses of C . ■

Again, we see that the local linear structure in the problem yields polynomial solvability, and lack of sufficient structure leads us to NP-completeness. In the next chapter, we discuss these and previous findings.

Chapter 6

Discussion

We now discuss our discoveries of Chapters 3 and 4. In Chapter 3, we showed that the optimal fitness for a function with local epistatic interaction can be found in polynomial time. In Chapter 4, it was demonstrated that the loss of the locality of these interactions causes the problem of finding the optimal fitness to become NP-complete. We might expect that more heuristic measures of fitness function complexity would also indicate an increase in complexity as locality is lost. Several methods for measuring the correlation of these landscapes are presented in [4]. One of the global measures of landscapes is the mean length of a hill-climbing walk from a random position on the landscape to a local fitness optimum.

When we look at results in [2], we see that the statistical difference between nearest-neighbor and random interactive networks is minimal in terms of the mean walk length to a local optima. This appears to be in contrast to our previous findings. The results of [2] suggest that, in the realm of fitness landscapes, the hill-climbing algorithm sees no difference in difficulty between an NP-complete problem and one which is polynomially solvable. This could indicate that the structure apparent in the fitness function is not translated into structure usable by the hill-climbing algorithm in the corresponding fitness landscape.

Another possibility is that the extent of the systems that are presented in [2] is not large enough to show a significant difference between these two distinct systems. In [2], we can see a slight difference between these two classes of systems. One could hypothesize that if the difference between N and K were greater, then the statistical differences would become apparent. Although the nearest-neighbor fitness functions can be optimized in polynomial time, they are dependent upon a factor of $A^{3(K-1)}$, which, although a constant, could be large enough to cause the statistical differences to be small for $N \leq 96$. Further work could be done to support or reject this notion.

Although the hill-climbing algorithm apparently sees no difference between a fitness landscape generated from a nearest-neighbor versus random epistatic fitness function, we might not expect the same result using a genetic algorithm with crossover. Assuming that we have two point crossover since we are dealing with a circular string, we would expect that two highly fit parents would produce highly fit offspring with high probability in the fitness landscapes generated from nearest-neighbor fitness functions. As discussed in [2], the reasoning behind this is that if $K \ll N$, distant regions are functionally independent. Crossover in the nearest-neighbor fitness function would affect only $2(K-1)$ fitness contributions ($K-1$ for each of the two breaks in the string) of the genome. If $K \ll N$, we would expect this affect to be negligible, thus yielding two half genomes with high fitness contributions. Furthermore, although it is noted that crossover performs better with parents close together (in terms of Hamming distance)[6], no assumption of the relative position of the parents is necessary in this hypothesis.

In the landscapes representing fitness functions with randomly chosen epistatic interactions, we expect the number of fitness contributions affected by crossover to be proportional to N , that is, each fitness contribution is affected by $K-1$ randomly chosen genes of the string; if any of these is contained in the portion of the string

to be crossed-over, the fitness contribution may change. If we began with highly fit parents, we would assume that more of the affected fitness contributions would be lower after crossover than would be higher. However, in [2] it is shown that recombination is useful in landscapes generated from random NK fitness functions if $K \ll N$. The reasoning behind this is that the highest optima are near one another in the landscape. Crossover is a means of searching the areas of the landscape between highly fit positions. It is clear that crossover would perform better with parents close together (in terms of Hamming distance) [6] on these landscapes. The reasoning is that the portion of the string that is crossed over is close (in terms of Hamming distance) to the portion of the string being replaced. This yields many fitness contributions that are unchanged during the crossover. Although crossover is apparently beneficial for overall fitness, it is unclear whether the amount of time for a population to reap the benefits of crossover on this type of landscape is of the same order of magnitude as a population using crossover on an NK landscape with local epistatic interactions. Again, further work could be done to investigate these properties of genetic algorithms on these types of landscapes.

Another goal of this work is to add meaning to the term “edge of chaos” as it applies to Boolean networks. We must begin first with some notion of “chaos” as it applies to the real number system.

Deterministic chaos in the real number setting is described by three essential properties.¹ First, chaotic systems show sensitivity to initial conditions. This property can also be discussed as the exponential growth of propagated errors. Second, these systems exhibit mixing behavior, i.e. small open subintervals of the domain which, when iterated, will eventually yield points in every small open subinterval of the domain. Finally, periodic points are dense in chaotic systems. Periodic points

¹This discussion of chaos is taken from [9].

are starting points which, through iteration, attain only a finite number of different values.

With Boolean networks, we have only a finite number of states. How then can we discuss these systems with the notion of chaos? We find in Boolean networks that a small proportion of gates that can be perturbed without affecting the state cycle that is entered corresponds to the sensitivity to initial conditions. The existence of large state cycles resembles the non-periodicity of chaotic systems[5].

Again, we found that with arbitrarily chosen wiring in a $K = 2$ fitness function with each position affecting its own contribution, the problem of finding the optimal fitness (and hence an optimally fit individual) is solvable in time polynomial in the size of the problem, i.e. N . However, when we go to a fitness function with arbitrarily chosen wiring with $K \geq 2$, the problem becomes NP-complete. The discussions of [2] suggest that we cross a similar border when we look at the differences in the behavioral aspects of random Boolean networks with 2 inputs per bit versus 3 inputs per bit. We might consider the time complexity of the optimization problem to be a good indication of the behavior of a similarly structured Boolean network, however this is not the case. Other work with random Boolean networks [2, 5, 7, 10], as well as cellular automata, shows that the complexity of the behavior can be controlled in networks by biasing the set of Boolean functions from which to choose. Most of the difference in behavior between $K = 2$ and $K = 3$ can be attributed to this phenomenon. Of the sixteen functions from which to build networks for $K = 2$, all but two are canalizing (controlled by only one input), yielding a network that is not sensitive to initial conditions. When we consider the 256 functions from which to build networks for $K = 3$, the percentage of canalizing functions is much reduced. There exists a strong link between the complexity of the behavior of the dynamical system and the types of functions that are considered available to build the network

[2, 5, 7]. Our assumptions of Chapters 3 and 4 only considered the structure of the wiring of the fitness functions with no stipulations on the integer-valued functions of the number of inputs per position.

We must look closer at the work of [10, 11, 12, 13] to find a more likely match between the time complexity analysis of Chapter 3 and the behavioral aspects of the corresponding dynamical systems. In [10], the dynamical systems discussed are separated into the aforementioned groups: random Boolean networks, homogeneous wiring with random functions, homogeneous functions with random wiring, and homogeneous wiring with homogeneous functions, i.e. cellular automata. The results of [10] indicate that the complex patterns that emerge in many cellular automata are lost when the wiring is randomized. We can relate such results to the difference in time complexity between the nearest-neighbor networks versus randomly wired fitness networks. Furthermore, in cellular automata with random wiring constrained to a neighborhood of cells, many complex features remain, which coincides with the η -neighborhood analysis of Chapter 3. There is no mention of $K = 2$ networks in [10], however, sample runs of NK Boolean networks with $K = 2$ show significantly ordered behavior. In essence, $K = 2$ networks are just permutations of one or more nearest-neighbor networks.

An interesting side-effect of the analysis of Chapters 3 and 4 is that the 3-SAT problem was used to show when we cross the border between polynomial time and NP-completeness. Much of the work done with random Boolean networks deals with the “canalizing” nature of the functions involved in the network. It is found that if the percentage of canalizing functions is high in the set of functions used for the network, the behavior of the network is more structured. Using a 3-SAT problem provides the most canalizing environment in that all of the functions chosen are OR clauses. The behavior of the network is very structured, yet finding pre-images must

be NP-complete (at least finding one of the pre-images is). Likewise, finding the optimal fitness in a like-structured fitness landscape was found to be NP-complete. This is another indicator that the proposed match between behavior of the dynamical systems and the complexity of their counterpart fitness landscapes is marginal at best.

Chapter 7

Conclusions

7.1 Conclusions

We see that the random epistatic interactions seem to be the cause for the NP-completeness of the problem of finding optimally fit beings in fitness landscapes. Although the statistical differences between random versus nearest-neighbor interactions in terms of mean walk length to local optima seems insignificant, it is hypothesized that this may be, in fact, due to the size of the test landscapes. The techniques used to show that fitness landscapes are NP-complete in terms of finding the optimal fitness can also be applied to show that finding the pre-images of a given state in a random NK Boolean network is also NP-complete.

Another property of NK Boolean networks that is not addressed by this work is the canalizing nature of the functions from which the network is built. In our analysis of the fitness landscapes, there were no restrictions on the integer-valued contribution functions for each position. Further work could be done to determine whether some sub-classes of integer valued functions would yield different analyses in terms of complexity.

Although we do not achieve a good match between the “edge of chaos” behavior

of Boolean networks and the complexity analysis of fitness landscapes, similarities do exist. We believe this work does expose some of the intricacies of fitness landscapes. In trying to add meaning to the phrase “chaotic behavior in finite systems,” any clues that help us better understand the causes or effects of this behavior are welcome.

Another benefit of this work is a better understanding of when combinatorial optimization problems actually do become NP-complete. The locality of the epistatic interactions in the NK fitness landscapes causes these problems to be solvable in polynomial time. A known NP-complete problem (SATISFIABILITY) also becomes solvable in polynomial time if similar locality is present in the problem. Understanding the importance of locality in these problems could lead to better search strategies by taking advantage of any locality that exists in instances of these types of problems.

Understanding the boundary at which these problems become NP-complete also gives us a means of testing algorithms that attempt to solve these problems. It is hoped that various “genetic” algorithms will provide good solutions to NP-complete problems in a reasonable amount of time. Comparing the performance of those algorithms on either side of the NP-completeness boundary may give us an indication of how well suited a particular algorithm is to solve a given problem.

Finally, the NK Boolean networks and fitness landscapes presented by Kauffman appear to be a good model of many real biological processes. The Boolean networks have features much like the genetic regulatory networks which are apparently responsible for cell differentiation. Kauffman uses the properties of populations adapting on rugged fitness landscapes to explain the radiation in early phylogenies in the evolutionary record and stasis occurring later in the record. Kauffman also uses the structure of these landscapes to investigate how populations can best adapt toward the optimal fitness peaks. Understanding the computational complexity of these landscapes should help us understand the subtleties of how well and how quickly

populations can achieve a reasonable fitness value.

In conclusion, we see many applications for the fitness landscapes representing the NK model of epistatic interactions presented by Kauffman. Understanding the computational complexity of finding the optima of these landscapes can help us better understand those applications that are modeled by these landscapes.

Bibliography

- [1] Garey, Michael R. and David S. Johnson, *Computers and Intractability – A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.
- [2] Kauffman, Stuart A. *Origins of Order – Self-Organization and Selection in Evolution*, Oxford University Press, New York, 1993.
- [3] Kauffman, Stuart A. *Antichaos and Adaptation*, Scientific American, August 1991: 78-84.
- [4] Lipsitch, Marc. *Adaptation on Rugged Landscapes Generated the Iterated Local Interactions of Neighboring Genes*, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, California, 1991.
- [5] Lynch, James F. *On the Threshold of Chaos in Random Boolean Cellular Automata*, Random Structures and Algorithms, Vol. 6, 1995: 239-260.
- [6] Manderick, Bernard, et al. *The Genetic Algorithm and the Structure of the Fitness Landscape*, Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, California, 1991.
- [7] Mitchell, Melanie, et al. *Dynamics, Computation and the “Edge of Chaos”*: A Re-Examination, Working Paper 93-06-040, Santa Fe Institute, 1993.
- [8] Papadimitriou, Christos H. and Kenneth Steiglitz, *Combinatorial Optimization – Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
- [9] Peitgen, Heinz-Otto, Hartmut Jürgens, and Dietmar Saupe, *Chaos and Fractals – New Frontiers of Science*, Springer-Verlag, New York, 1992.
- [10] Wuensche, Andrew. *The Ghost in the Machine – Basins of Attraction of Random Boolean Networks*, Artificial Life III, A proceedings volume in the Santa Fe Institute Studies in the Sciences of Complexity, Addison-Wesley Publishing Company, Reading, Massachusetts, 1994.
- [11] Wuensche, Andrew. *Complexity in One-D Cellular Automata: Gliders, Basins of Attraction and the Z Parameter*, Cognitive Science Research Papers, University of Sussex at Brighton, 1994.

- [12] Wuensche, Andrew. *The Emergence of Memory Categorisation Far from Equilibrium*, Cognitive Science Research Papers, University of Sussex at Brighton, 1994.
- [13] Wuensche, Andrew and Mike Lesser. *The Global Dynamics of Cellular Automata – An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.