Additively Decomposable Fitness Functions

Richard K. Thompson   Alden H. Wright
rkt@charlo.math.umt.edu   wright@cs.umt.edu
Computer Science Dept.
The University of Montana
Missoula, MT 59812-1008

ABSTRACT

We define a class of optimization objective functions over discrete domains called *additively decomposable* functions. An additively decomposable function is a function over several discrete variables that can be written as a sum of component functions, each of which depends on only a small number of those variables. The *NK* fitness landscapes defined by Kauffman are a special case. If the domain variables have a circular ordering, then under the *adjacent model*, we impose the further restriction that each component function depends on a sequence of adjacent variables in the ordering. Bandwidth-constrained problems and adjacent *NK* landscapes are special cases. We give a polynomial-complexity dynamic programming algorithm to optimize adjacent-model additively decomposable functions, and show that without the adjacency assumption, the optimization problem is NP-complete.

1 Introduction

One of the most important outstanding problems for heuristic optimization techniques (including evolutionary computation applied to optimization) is characterizing which optimization methods will succeed when applied to various categories of objective functions. The "No Free Lunch" results of Wolpert and Macready (Wolpert, 1995) help to clarify this problem. They consider the set of all search algorithms that look at a fixed number of points in a finite search space, and that do not look at the same point in the search space more than once. Then, considered over the space of all possible objective functions into a finite range, all such search algorithms are equivalent. Thus, if one does not somehow restrict the space of objective functions to some subspace of "reasonable" functions, one has a hard time claiming that any search method is better than an enumeration of the search space. Objective functions of the type considered in this paper are one candidate for a class of "reasonable" objective functions. We will use evolutionary computation terminology and call objective functions "fitness functions" or "fitness landscapes". We also suppose that our objective is to maximize the given fitness function.

Let f be a nonnegative real-valued function over fixed-length strings. We will say that f is *additively decomposable* if f can be written as the sum of functions each of which depends on a subset of the symbols of the string. To be more precise, let Σ be a fixed finite alphabet, and let f :

$\Sigma^N \longrightarrow R^{\geq 0}$. Then f is additively decomposable if $f = \sum f_i$, where f_i depends on $k_i < N$ symbols. To give an example, suppose that $n = 3$. Let $f(a_0, a_1, a_2) = f_0(a_0, a_1) + f_1(a_1, a_2) + f_2(a_2, a_0)$, where each f_i is an arbitrary nonnegative real-valued function over Σ^2.

If each $k_i = 1$ so that each component function depends on only a single symbol of the string, then we will call f *linear*, or *non-epistatic*.

The *NK* fitness landscapes introduced by Kauffman (Kaufman, 1993) are examples of additively decomposable fitness landscapes. For these landscapes, there are exactly N of the component functions f_i, and every component function f_i depends on $K+1$ symbols of the string. The values for the component functions are chosen from a uniform random distribution over the interval $[0,1]$. Thus, an *NK* fitness landscape is specified by N tables, where each table is indexed over Σ^{K+1} and contains random real values uniformly selected from $[0,1]$. The i^{th} table specifies the values of the function f_i. There are two variations of the *NK* model. In the *adjacent* model, f_i depends on symbol a_i and K adjacent symbols. In the *random* model, f_i depends on symbol a_i and K other randomly chosen symbols. In most applications and analysis of the *NK* model, the alphabet has been $\{0,1\}$.

In the *adjacent* additively decomposable model, we assume there are N component functions f_i, and that each f_i depends on the $K+1$ adjacent symbols $a_i, a_{i+1}, \ldots a_{i+K}$ of the string. (The usual assumption for the adjacent *NK* model is that f_i depends on string symbols $a_{i-\lfloor K/2 \rfloor} \ldots a_{i+\lceil K/2 \rceil}$. A reindexing of the component functions f_i gives our assumption, which gives simpler notation.)

In the *arbitrary* additively decomposable model, we make no assumptions on which symbols of string each f_i depends. Sometimes, we will want to assume that each f_i depends on at most $K+1$ symbols of the string.

We give an example of an adjacent-model additively decomposable fitness function with $K = 2$ and where N can be any even integer greater than or equal to 4.

i is even

σ_i	σ_{i+1}	σ_{i+2}	f_i
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

i is odd

a_i	a_{i+1}	a_{i+2}	f_i
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

For $N = 4$ this function has four global maxima, namely at 0000, 0011, 1100, and 1111. For general N, every string consisting of a sequence of pairs of equal bits has a value of N. Thus, f has $2^{N/2}$ global maxima.

Most of the results of this paper are in the Master's thesis Thompson (1995). Weinberger (1995) independently discovered several of the results of this paper.

2 Applications

Kauffman (1993) describes many possible applications of the NK model to evolutionary biology including evolution of the catalytic function of proteins, evolution of population fitness, and adaptive maturation of the immune response.

To show that additively decomposable model is relevant to combinatorial optimization, we show how constraint satisfaction problems can be formulated under this model. It will be convenient to temporarily assume that we are minimizing f rather than maximizing f.

Consider the problem of coloring the vertices of a graph with m colors so that the colors of adjacent vertices are different. Let the alphabet Σ be the set of colors. Let each vertex correspond to a string position, so that a string over Σ defines a coloring of the graph. For each vertex i, let f_i be the number of vertices adjacent to i with the same color as i. Clearly, if $f = \sum f_i$ is zero, we have achieved a legal coloring of the graph. Then N is the number of vertices of the graph. An alternative approach would be to let the functions f_i correspond to edges of the graph, where f_i has value 1 if the vertices of the edge have the same color, and 0 otherwise. This leads to a model with more component functions than N.

For a more general constraint satisfaction problem over a set of discrete-valued variables, we assume that each constraint is associated with a penalty function. We let Σ be the possible values for the variables, and we let string positions correspond to variables. Then f_i can be the sum of the penalty functions of the constraints that involve the variable i, or f_i can be the penalty function associated constraint i.

The adjacent-model additively decomposable functions are closely related to the bandwidth-constrained NP-complete problems considered by Monien and Sudborough (1985). A SAT (satisfiability) problem has bandwidth less than or equal to L if there is a rearrangement of the order of the clauses so that whenever a variable or its negation appears in clauses C_i and C_j, then $|i - j| \leq L$. Then Monien and Sudborough define 3SAT($f(n)$) to be the set of 3SAT problems with bandwidth bounded by $O(f(n))$. They show that 3SAT($\log n$) is in NSPACE($\log n$), which implies that it is in class P (solvable by a deterministic polynomial-time algorithm). They define a notion of bandwidth preserving log space reduction, and show that a variety of combinatorial problems with bandwidth constrained by $f(n)$ are equivalent to 3SAT($f(n)$) under this notion of reduction.

The the adjacent model additively decomposable functions of this paper can be thought as subject to a circular bandwidth restriction, since the indices are taken $\bmod N$ in the definition of adjacency. (See the next section.)

3 Formalization

We represent the additively decomposable model in a mathematically precise way.

Let Σ denote a (finite) alphabet. To specify an additively decomposable fitness function $f = \sum f_i$, we must specify the positions that influence each term f_i, and we must specify the f_i functions.

We specify the positions through projection functions p_i, where each p_i is a mapping from Σ^N to Σ^{k_i}. Each p_i is defined by a cardinality k_i subset of $\{0, 1, \ldots, N-1\}$. For example, if $k_i = 3$ and if p_2 is defined by the subset $\{1, 3, 6\}$, then p_2 is defined by $p_2(a_0, a_1, \ldots, a_{N-1}) = (a_1, a_3, a_6)$.

For the adjacent model, we choose the subset of $\{0, 1, \ldots, N-1\}$ that defines p_i to be $\{i, i+1, \ldots, i+K\}$, where the indices are taken $\bmod N$. For the arbitrary model, the subset that defines p_i is arbitrarily chosen. Sometimes we may specify that i is in the subset that defines p_i. Then we assume that the component functions f_i map Σ^{K+1} into $R^{\geq 0}$. Then f is written more precisely as

$$f = \sum_{i=0}^{N-1} f_i \circ p_i$$

where the domain of each f_i is Σ^{K+1}. When it is clear from the context, we will omit the projection functions.

The input to the optimization problem consists of the subsets defining the p_i and a set of tables defining the f_i.

4 Polynomial Algorithms

4.1 The $K = 1$ adjacent case

An algorithm similar to the one of this section was independently discovered by Ed Weinberger (1995).

We demonstrate a polynomial dynamic programming algorithm to find the optimum fitness of the adjacent model. We start with an algorithm for the $K = 1$ case.

The idea of the algorithm is to reduce the problem of size N to a problem of size $N-1$ by defining an fitness function $f' : \Sigma^{N-1} \longrightarrow R^{\geq 0}$, where $R^{\geq 0}$ denotes the nonnegative reals. The function f' is defined as

$$f' = \sum_{i=0}^{N-2} f'_i$$

where $f'_i = f_i$ for $0 \leq i < N-1$, and

$$f'_{N-2}(a_{N-2}, a_0) = \max\{f_{N-2}(a_{N-2}, b) + f_{N-1}(b, a_0) : b \in \Sigma\}$$

Theorem 4.1 *Under the definition of f' given above,*

$$\max\{f'(\mathbf{a}') : \mathbf{a}' \in \Sigma^{N-1}\} = \max\{f(\mathbf{a}) : \mathbf{a} \in \Sigma\}$$

Proof. Define $p : \Sigma^N \longrightarrow \Sigma^{N-1}$ by $p(a_0, \ldots, a_{N-2}, a_{N-1}) = (a_0, \ldots, a_{N-2})$. Then for any $\mathbf{a} \in \Sigma^N$,

$$f(\mathbf{a}) - f'(p(\mathbf{a})) = f_{N-2}(a_{N-2}a_{N-1}) + f(a_{N-1}a_0) - f'(a_{N-2}a_0)$$

Since

$$\begin{aligned} f'_{N-2}(a_{N-2}, a_0) &= \max\{f_{N-2}(a_{N-2}, b) + f_{N-1}(b, a_0) : b \in \Sigma^N\} \\ &\geq f_{N-2}(a_{N-2}, a_{N-1}) + f(a_{N-1}, a_0) \end{aligned}$$

we have

$$f(\mathbf{a}) \quad \leq \quad f'(p(\mathbf{a})) \tag{1}$$

Let $\mathbf{c}' \in \Sigma^{N-1}$ be such that $f'(\mathbf{c}') = \max\{f'(\mathbf{a}') : \mathbf{a}' \in \Sigma^{N-1}\}$. Then

$$f'_{N-2}(c'_{N-2}, c'_0) \quad = \quad \max\{f_{N-2}(c'_{N-2}, b) + f_{N-1}(b, c'_0) : b \in \Sigma\}$$

Choose $c_{N-1} \in \Sigma$ to achieve this maximum, and let $c_i = c'_i$ for $i < N-1$ so that $\mathbf{c}' = p(\mathbf{c})$, and

$$f'_{N-2}(c'_{N-2}, c'_0) \quad = \quad f_{N-2}(c_{N-2}, c_{N-1}) + f_{N-1}(c_{N-1}, c_0)$$

which implies that $f(\mathbf{c}) = f'(p(\mathbf{c})) = f'(\mathbf{c}')$. This, along with equation (1), implies the statement of the theorem. $\square$

If we know the string $\mathbf{c}' \in \Sigma^{N-1}$ that maximizes f', and if we know the element $b \in \Sigma$ which realizes the maximum in the definition of f'_{N-2}, then we can construct the element $\mathbf{c} \in \Sigma^N$ which maximizes f. It is not hard to see that if $\mathbf{c}' = (c_0, c_1, \ldots, c_{N-2})$, then $\mathbf{c} = (c_0 c_1 \ldots c_{N-2} b)$.

We can now write the above in the form of an algorithm. We assume that the tables that define the components of the given fitness function f are given in the form of a 3-dimensional array $F[0..N-1, \Sigma, \Sigma]$. In other words, the function $f_i : \Sigma^2 \longrightarrow R^{\geq 0}$ is given by $f_i(a, b) = F[i, a, b]$. The algorithm repeatedly applies the process of Theorem 4.1 to reduce the problem to a problem of size $N = 2$. The solution of the $N = 2$ problem is found by direct search, and then a solution to the original problem is reconstructed. The F array is used as temporary storage by the algorithm, so the original contents of F are destroyed. At the end of the ith stage, $F[0..i-2, \Sigma, \Sigma]$ contains the definition of the current component functions, and $F[i-1..N, \Sigma, \Sigma]$ contains the elements of Σ that realize the maximum in the definition of f' above. For example, if $F[N-1, a, c] = d$, then $f_{N-2}(a, d) + f_{N-1}(d, c) = \max\{f_{N-2}(a, b) + f_{N-1}(b, c) : b \in \Sigma\}$.

Optimize(F)
/// Assumes that $K = 1$.
/// The given array $F[0..N-1, \Sigma, \Sigma]$ which defines the fitness function, is destroyed.
/// $v[\Sigma, \Sigma]$ and $u[\Sigma, \Sigma]$ are temporary arrays.
/// Reduce the problem size to $N = 2$.

for n **from** $N-1$ **downto** 2 **do**
  for $a \in \Sigma$ **do**
    for $c \in \Sigma$ **do**
      Choose b_{max} so that
        $F[n-1, i, b_{max}] + F[n, b_{max}, k] = \max\{F[n-1, a, b] + F[n, b, c] : b \in \Sigma\}$
      $v[a, c] \leftarrow b_{max}$
      $u[a, c] \leftarrow F[n-1, i, b_{max}] + F[n, b_{max}, k]$
  for $a \in \Sigma$ **do**
    for $c \in \Sigma$ **do**
      $F[n-1, a, c] \leftarrow u[a, c]$
      $F[n, a, c] \leftarrow v[a, c]$

/// Problem size is now $N = 2$. Find the max fitness string for $N = 2$
  Choose a_{max} and c_{max} so that
    $F[0, a_{max}, c_{max}] + F[1, c_{max}, a_{max}] = \max\{F[0, a, c] + F[1, c, a] : a \in \Sigma, c \in \Sigma\}$

/// Construct an optimal string S for the whole problem.
  $S[0] \leftarrow i_{max}$
  $S[1] \leftarrow k_{max}$
  for i **from** 2 **to** $N-1$ **do**
    $S[i] \leftarrow F[i, S[i-1], S[0]]$

  return S

Algorithm 1

Clearly, the time complexity of this algorithm is $\Theta(N|\Sigma|^3)$ if we assume that operations on real numbers (such as addition and maximizaton) require $O(1)$ time.

The space required is that required for the arrays F, u, and v. Thus the space used is $O(N|\Sigma|^2)$ on the assumption that every real number used in the algorithm can be stored in $O(1)$ space.

4.2 The $K > 1$ adjacent case

First, suppose that N is divisible by K. Then we can view a string of length N over Σ as a string of length N/K over alphabet Σ^K. Any component function f_i which depends on at most $K+1$ positions of the string over Σ will depend on at most 2 positions of the string over alphabet Σ^K. If we let $\widetilde{f}_i = \sum_{j=Ki}^{Ki+K-1} f_i$, then $\widetilde{f}_i$ depends on only symbols $\widetilde{a}_i$ and $\widetilde{a}_{i+1}$ of the string over the alphabet Σ^K. Then we can apply the Algorithm 1 of the previous section to achieve an algorithm of complexity $\Theta(N|\Sigma|^{3K})$.

If N is not divisible by K, let $Q = \lfloor N/K \rfloor$ and $r = N \bmod K$. We view a string of length N over Σ as a string of symbols $\widetilde{a}_0\widetilde{a}_1 \ldots \widetilde{a}_Q$, where $\widetilde{a}_0 \in \Sigma^K$, $\widetilde{a}_1 \in \Sigma^r$, and $\widetilde{a}_2, \ldots, \widetilde{a}_Q \in \Sigma^K$. We can write $f = \sum_{i=0}^{Q} \widetilde{f}_i$ where each $\widetilde{f}_i$ depends on $\widetilde{a}_i$ and $\widetilde{a}_{i+1}$, except for $\widetilde{f}_0$ which depends on $\widetilde{a}_0$, $\widetilde{a}_1$, and $\widetilde{a}_2$. We apply Algorithm 1 of the previous section to reduce the problem to a problem over a string of length $2K + r$ over Σ, or a string of length 3 over $\Sigma^K \times \Sigma^r \times \Sigma^K$. Note that Algorithm 1 does not work with component function $\tilde{f}_0$. Each major iteration of the first part of Algorithm 1 takes $\Theta(|\Sigma|^{3K}|)$ steps. We solve the reduced problem by enumerating all solutions, which takes $|\Sigma|^{2K+r} \in O(|\Sigma|^{3K})$ steps. This gives the following theorem:

Theorem 4.2 *Let f be an adjacent-model additively decomposable fitness function over the alphabet Σ. Then a string corresponding to the optimal fitness value can be found in time $\Theta((Q+1)|\Sigma|^{3K})$, where $Q = \lfloor N/K \rfloor$ (assuming that integer operations can be done in $O(1)$ time).*

Corollary 4.3 *Let $C > 0$ be a constant, and consider the family of adjacent-model additively decomposable fitness functions over the alphabet Σ such that $K \leq C \log N$. There is a polynomial algorithm to find the optimal string for this family of fitness functions.*

Proof. The time complexity of the algorithm of Theorem 4.2 is $O(N|\Sigma|^{3C\log N}) = O(N^{3C\log|\Sigma|+1})$. □

4.3 A $K = 1$ arbitrary case

In this section we show that for $K = 1$ and the arbitrary model where component function f_i of f depends on position i of the string, then there is a polynomial algorithm to optimize f.

We define a graph whose vertices are the integers from 0 to $N - 1$, and whose edges are the terms f_i. The edge f_i connects the vertices of the set of indices corresponding to the projection p_i. In other words, the edge f_i connects the indices corresponding to those components of the string that f_i depends on. Note the graph corresponding to a $K = 1$ adjacent-model additively decomposable fitness function is a cycle.

Let us rearrange the indices $\{0, 1, \ldots, n-1\}$ so that the indices corresponding to the connected components of G are adjacent. Thus, we assume that the vertices $\{n_0 = 0, 1, \ldots, n_1 - 1\}$ correspond to the first connected component of G, the vertices $\{n_1, n_1+1, \ldots, n_2-1\}$ correspond to the second, etc. The let

$$g_j = \sum_{i=n_j}^{n_{j+1}} f_i$$

where g_j depends only on the components $n_j, \ldots n_{j+1}-1$, so the g_j can be optimized independently, and the sum of the optimal values for the g_j is the optimal value for f. Further, the optimal string for f is the concatenation of the optimal strings for the g_j.

Theorem 4.4 *Let f be an additively decomposable fitness function with $K = 1$, where each f_i depends on position i of the string and one other position. Then there is a polynomial algorithm to optimize f.*

Proof. Our assumption is that the edge f_i must be incident on vertex i. Thus, every component of the graph has the same number of edges as vertices, and every component of the graph has exactly one cycle.

If a component of the graph G consists of a cycle, then we can apply Theorem 4.1 and use Algorithm 1.

If a component of G is not a cycle, then there must be some vertex of degree 1. We show how the function f can be replaced by a function f' which does not depend on this string position, and which has the same optimum.

Without loss of generality, we assume that index $N-1$ corresponds to a vertex of G of degree 1. Then f_{N-1} is the only term of f which depends on this position of the string. Without loss of generality, we can also assume that the other position of the string that f_{N-1} depends on is $N-2$.

Let

$$f'_{N-2}(a_{N-1}, a_{N-2}) = f_{N-2}(a_{N-2}, a_{N-1}) + \max\{f_{N-1}(a_{N-2}, c) : c \in \Sigma\}$$

Clearly, if $a_0 \ldots a_{N-2}a_{N-1}$ is optimal for f, then $a_0 \ldots a_{N-2}$ is optimal for f'. □

5 NP-Completeness

In this section we show that when the restriction that f_i depends on position i of the string is removed, then the $K = 1$ arbitrary-model optimization problem is NP-complete. We now assume that the function f maps into the nonnegative integers rather than the nonnegative reals.

Theorem 5.1 *The problem of optimizing an additively decomposable fitness function $f = \sum f_i$ with $K = 1$ and no restrictions on the dependence of the f_i on string positions is NP-complete.*

Proof. To show NP-completeness, we must show that a solution to the corresponding decision problem can be checked in polynomial time, and that some problem known to be NP-complete can be reduced to our problem. The decision problem corresponding to the problem of maximizing f is: Given a fixed natural number k, decide if there exists a string $\mathbf{a} \in \Sigma^N$ such that $f(\mathbf{a}) \geq k$. Since f can be computed in polynomial time, a solution can be checked in polynomial time.

The known NP-complete problem that we will reduce to our problem is the MAXIMUM 2-SAT problem (Garey, 1979): Given a Boolean formula in conjunctive normal form (CNF) with 2 literals per clause, the problem is to maximize the number of true clauses. A 2-CNF formula is a conjunction of clauses of the form $u_i \vee u_j$, where u_i and u_j are literals, where a literal is either a Boolean variable or a negation of a Boolean variable.

Given a 2-CNF formula, we will construct a corresponding additively decomposable fitness function f over the alphabet $\{0, 1\}$ so that the value of f is the number of true clauses in the 2-CNF formula. Let N be the maximum of the number of variables and the number of clauses in the CNF formula. To each Boolean variable of the formula, we associate a string position, and to each clause of the formula, we associate a term f_i of f. If there are string positions that are not associated with Boolean variables, then the symbols in these positions will not affect the value of f, and if there are f_i which are not associated with clauses, then we set these f_i to have value zero.

We assume that the ith Boolean variable of the formula is associated with the ith string position, and that the ith clause of the formula is associated with f_i. Thus, each f_i depends on two string positions. Let a string position value of 1 correspond to a *true* value of the corresponding Boolean variable, and a value of 0 correspond to *false*. The value of f_i will be defined to be 1 if the corresponding clause of the formula is *true* and 0 if the corresponding clause is *false*. For example, if f_i corresponds to the clause $(x_2 \vee \overline{x_4})$, then f_i will depend on string positions 2 and 4, and the values of f_i are given by

a_2	a_4	f_i
0	0	1
0	1	0
1	0	1
1	1	1

Clearly, the value of f is the number of *true* clauses in the 2-CNF formula. □

Under Kauffman's model, f_i was required to depend on string position i. If $K = 2$ and this is the only restriction on the dependence of f_i, then clearly Theorem 5.1 shows that optimizing such an f (with integer values) is an NP-complete problem. Weinberger (1995) has independently discovered this result.

6 Conclusion

The additively decomposable fitness landscapes generalize Kauffman's *NK* fitness landscapes by allowing arbitrary component functions rather than stochastically defined component functions. We have given an algorithm for optimizing the adjacent-model additively decomposable landscapes which is polynomial in N. This implies that if $K \in O(\log N)$, then the problem of optimizing adjacent-model landscapes is in class P. Arbitrary-model additively decomposable landscapes where the ith component function is not required to depend on string component i are NP-complete for $K \geq 1$.

From another point of view, the dependency of the component functions of the fitness function on the positions of the domain string is defined by a graph if $K = 1$ (or a hypergraph if $K > 1$). For $K = 1$, if each connected component of this graph contains a single cycle, the optimization problem is polynomial in N, and for $K \geq 1$ without this restriction, the optimization problem is NP-complete.

Kauffman (1993) empirically compared the adjacent-model and random-model *NK* landscapes using hill-climbing. He compared the mean fitness of local optima under the two models, and found very little difference. He also compared mean walk lengths to local optima, and again found very little difference. Weinberger (1995) computed formulas for the correlation between strings of Hamming distance d under both models. For the random-model landscape, he obtained:

$$R(d) = 1 - \frac{d(k+1)}{N} + \frac{d(d-1)k(k+2)}{2N^2} + O\left(\left(\frac{dk}{N}\right)^3\right)$$

and for the adjacent-model landscape, he obtained:

$$R(d) = 1 - \frac{d(k+1)}{N} + \frac{d(d-1)k(k+1)}{2N^2} + O\left(\left(\frac{dk}{N}\right)^3\right)$$

These are clearly very close. Thus, statistically the adjacent and random model NK landscapes are very close, at least relative to a Hamming-distance topology.

Relative to a crossover topology, we might expect different results. The component functions f_i can be considered as "building blocks", and when K is small relative to N, one-point or two-point crossover preserves most of the values of the f_i from one parent or the other on adjacent-model fitness functions, but does not on random-model fitness functions. (We expect to make this more precise in future research.)

The results of Kauffman and Weinberger make the results of this paper seem somewhat surprising. It may be that the "algorithmically hard" problems are a statistically small subset of the random-model NK fitness landscapes.

References

Garey, Michael R., and Johnson, David S. (1979) *Computers and Intractibility* , W. H. Freeman and Company, San Francisco.

Kauffman, Stuart A. (1993). *The Origins of Order.* New York: Oxford University Press.

Monien, Burkhard, and Sudborough, Ivan H. (1985) Bandwidth Constrained NP-Complete Problems, *Theoretical Computer Science*, vol. 41, pp. 141-167.

Thompson, Richard K., (1995) Fitness Landscapes Investigated, Master's Thesis, University of Montana. (available from the University of Montana Library, Missoula, MT, 59812.)

Wolpert, David H. and Macready, William G. (1995) No Free Lunch Theorems for Search. Santa Fe Technical Report 95-02-010.

Weinberger, E. (1995) $\mathcal{NP}$ Completeness of Kauffman's N-k Model, a Tuneably Rugged Fitness Landscape. Preprint.

Wright, Sewall. (1932) The roles of mutation, inbreeding, crossbreeding, and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, vol. 1, pp. 356-366.