Efficient Algorithms for Protein Sequence Design and the Analysis of Certain Evolutionary Fitness Landscapes *

Jon M. Kleinberg†

Abstract

Protein sequence design is a natural inverse problem to protein structure prediction: given a *target structure* in three dimensions, we wish to design an amino acid sequence that is likely fold to it. A model of Sun, Brem, Chan, and Dill casts this problem as an optimization on a space of sequences of hydrophobic (H) and polar (P) monomers; the goal is to find a sequence which achieves a dense hydrophobic core with few solvent-exposed hydrophobic residues. Sun et al. developed a heuristic method to search the space of sequences, without a guarantee of optimality or near-optimality; Hart subsequently raised the computational tractability of constructing an optimal sequence in this model as an open question. Here we resolve this question by providing an efficient algorithm to construct optimal sequences; our algorithm has a polynomial running time, and performs very efficiently in practice. We illustrate the implementation of our method on structures drawn from the Protein Data Bank. We also consider extensions of the model to larger amino acid alphabets, as a way to overcome the limitations of the binary H/P alphabet. We show that for a natural class of arbitrarily large alphabets, it remains possible to design optimal sequences efficiently.

Finally, we analyze some of the consequences of this sequence design model for the study of evolutionary fitness landscapes. A given target structure may have many sequences that are optimal in the model of Sun et al.; following a notion raised by the work of J. Maynard Smith, we can ask whether these optimal sequences are "connected" by successive point mutations. We provide a polynomial-time algorithm to decide this connectedness property, relative to a given target structure. We develop the algorithm by first solving an analogous problem expressed in terms of *submodular functions*, a fundamental object of study in combinatorial optimization.

*A preliminary version of this work appears in the Proceedings of the 3rd ACM RECOMB Intl. Conference on Computational Molecular Biology

†Department of Computer Science, Cornell University, Ithaca NY 14853. Email: kleinber@cs.cornell.edu.
Supported in part by an Alfred P. Sloan Research Fellowship and by NSF Faculty Early Career Development Award CCR-9701399.

1 Introduction

Protein Sequence Design. Understanding the principles by which proteins adopt their native three-dimensional structures is a fundamental issue, involving a rich set of biophysical and computational problems. The intensively studied problem of *protein structure prediction* begins with a given amino acid sequence and seeks to characterize, by computational means, the structure or range of structures that that this sequence will adopt under physiological conditions [27]. There is a natural "inverse" version of this problem, the object of several recent studies [11, 30, 37, 34, 10, 33, 17, 3], in which one begins with a given three-dimensional protein *structure*, and seeks to determine the sequence or collection of sequences most likely to fold to this structure. Recent work has indicated that this problem of *protein sequence design* — also referred to as the *inverse protein folding* problem — can provide a valuable perspective on the issues surrounding protein structure.

Determining an appropriate model in which to study the protein sequence design problem is a challenging prospect in itself. In an interesting recent development, a set of related approaches has been advanced in the biophysics community (Sun, Brem, Chan, and Dill [33], Shakhnovich and Gutin [34], Deutsch and Kurosky [10]); these approaches cast sequence design as a global optimization problem on the space of amino acid sequences. Roughly, they search for the sequence that optimizes a *fitness function*, constructed to favor the properties that a "good" sequence is presumed to possess. Through the development of an appropriate fitness function, these approaches attempt implicitly to capture the competing requirements of *positive design* — the designed sequence should have low free energy in the target structure — and *negative design* — there should be very few other "competing" structures in which the designed sequence has comparable free energy [9, 37].

The present work: Computing optimal sequences. In this work, we begin by focusing on one of these sequence design models, the *Grand Canonical* (GC) model of Sun, Brem, Chan, and Dill [33]. We will define the model fully in Section 2. For now, it is enough to note that the GC model works with (i) an accurate three-dimensional geometric representation of a target structure with n amino acid residues; (ii) a *binary folding code* in which there is only a distinction between *hydrophobic* (H) and *polar* (P) residues [23, 9]; and (iii) a fitness function Φ defined in terms of the target structure so as to favor sequences with a dense hydrophobic core and to penalize sequences with many solvent-exposed hydrophobic residues. With respect to a given geometric target structure, one is interested in the H/P sequence(s) whose fitness is *optimal*. While the use of a two-letter H/P amino acid alphabet is clearly a simplification, there has been considerable work (see e.g. [23, 9, 21]) suggesting that modeling a protein as a heteropolymer with only two amino acid types captures many of the qualitative aspects of protein structure.

There have been several recent studies aimed at understanding the relationship between such sequences of optimal fitness in the GC model and those found in real proteins [33, 3, 28]. However, the following issue stood in the way of a complete assessment of the model's biological accuracy: It was not known how to compute *optimal* sequences in the GC model, short of a computationally infeasible brute-force search over sequence space. Indeed, Hart raised the computational tractability of computing an optimal sequence, with respect to a three-dimensional geometric target structure, as an open question [17].

We resolve this question by providing an efficient algorithm to find optimal protein sequences in the general GC model with respect to a three-dimensional geometric structure. Our algorithm has a running time that is polynomial in the length of the sequence being designed; and we have produced an implementation of the algorithm (on top of discrete optimization code of Cherkassky and Goldberg [6]) that runs very efficiently on sequences derived from real data. Indeed, the algorithm runs to optimality in a few seconds on target structures more than twice as large as those studied by Sun et al. [33]. (See the plot in Figure 2, which depicts the running time of our implementation (as a function of sequence length) on 25 target protein structures from the Protein Data Bank (PDB) [4].) Our algorithm makes use of techniques from the area of *network flow* [1], a powerful body of algorithmic work that we feel could have promising applications to other biomolecular structure problems as well.

Given an efficient algorithm to design optimal sequences, we are able to perform assessments of the type in [33, 28]: for a protein drawn from the PDB, we can compute an optimal sequence for it under the GC model, and compare this to the protein's actual amino acid sequence. We are also able to use our algorithmic techniques to develop extensions of the GC model in which optimal sequences can still be computed efficiently. One of these models allows for a class of amino acid alphabets of arbitrarily large size, provided that their contact energies satisfy certain conditions. This allows one to overcome the limitations of the binary H/P alphabet and study certain types of 20-letter amino acid alphabets, while still being able to compute optimal sequences. We also consider a natural *fractional* version of the GC model, in which each residue can specify an arbitrary real-valued hydrophobicity value. We show that optimal sequences can be computed efficiently in this model as well; but we also show a surprising sense in which this seemingly more general fractional model degenerates into the standard GC model.

The present work: Evolutionary fitness landscapes. There is now a wealth of evidence that proteins with little sequence similarity can still adopt very similar three-dimensional structures [18, 19]. This suggests that for a physiologically "important" protein structure, a wide range of sequences are capable of folding to it; in this context, one is interested in the evolutionary relationships among such a collection of diverse sequences with common folding behavior. The GC model can provide us with an interesting computational approach to such issues: with respect to a given target structure, we can study the set Ω of all sequences that are *optimal* under the associated fitness function Φ, and understand the structure of this space with respect to mutations. We note that the set Ω can be quite large: it is not difficult to construct examples in which the number of optimal sequences is exponential in n.

The most basic type of mutation in the GC model is a *one-point mutation*: a single position in a protein sequence flips from one type of amino acid to the other. (Recall that the model has a binary amino acid alphabet.) A question of fundamental interest is whether the set Ω of optimal sequences for a given structure has the following natural *connectedness* property: if S and S' are both sequences in Ω, then there is a chain of one-point mutations transforming S to S', so that all intermediate sequences in this transformation lie in Ω as well. Such a chain represents a hypothesized evolutionary "trajectory" by which S and

S' diverged, with the property that all intermediate sequences on this trajectory retained a strong propensity to fold to the target structure. This notion is captured in Maynard Smith's discussion of natural selection of proteins [26], which served to motivate recent evolutionary analyses of lattice protein models [9, 24, 25]: "If evolution by natural selection is to occur, functional proteins must form a continuous network which can be traversed by unit mutational steps without passing through non-functional intermediates" [26]. More generally, the notion of such evolutionary pathways through functional sequences is central to the study of *neutral networks* in sequence space (e.g. [2, 22, 20, 26, 31]), in which one is concerned with the effect of fitness-preserving mutations on the dynamics of evolution.

In this paper, we provide the first polynomial-time algorithm for determining whether the set of optimal sequences for a given target structure in the GC model is *connected* under one-point mutations. We will refer to this as the *global connectivity problem*. Even when the set of optimal sequences is not connected, we may ask, for a *specific* pair of optimal sequences S and S', whether there is a sequence of mutations from S to S' so that all intermediates are optimal. We show how to solve this *pairwise connectivity problem* in polynomial time as well.

We can in fact solve these two problems in a more general framework, in which the "unit mutational step" may consist of the simultaneous mutation of several positions in the sequence; we define this generalization precisely in Section 4. Note that determining the connectedness of the optimal set Ω — or the pairwise connectivity of two specific sequences — involves the following challenge: Ω can have a size that is exponential in n, the number of residues in the target structure; thus, if our algorithm is to run in time polynomial in n, it must operate without explicitly examining more than a negligible fraction of the sequences in Ω.

To solve these connectedness problems, we first develop efficient algorithms for the following basic problems in combinatorial optimization. (See Section 4 for definitions.) Let f be an arbitrary *submodular function*, which maps subsets of an n-element set U to $\mathbf{R}$, and let Ω_f denote the collection of all subsets U' of U on which f attains its minimum value. The classical *submodular function minimization problem* asks for a polynomial-time algorithm to produce a member of Ω_f. In order to deal with the global connectivity problem discussed above, we must solve the problem of determining whether the set Ω_f is *connected* in the following sense: for any two sets $U', U'' \in \Omega_f$, there should be a sequence of insertions and deletions of single elements that transforms U' into U'', in such a way that each intermediate set is also in Ω_f. Analogously, for the pairwise connectivity problem, we are given specific U' and U'' in Ω_f, and we must decide whether there is a sequence of insertions and deletions of single elements that transforms U' into U''. We provide polynomial-time algorithms for these problems, assuming only "black-box" access to f. More generally, suppose we are given an arbitrary collection Λ of subsets of U that is *downward-closed*: if $L \in \Lambda$ and $L' \subseteq L$, then $L' \in \Lambda$. If we view members of Λ as representing allowable unit mutational steps, our notion of connectedness becomes the following: for two sets $U', U'' \in \Omega_f$, there should be a *sequence* of sets $U_1, U_2, \ldots, U_t$, each belonging to Ω_f, so that $U_1 = U'$, $U_t = U''$, and for each i, the symmetric difference $(U_i - U_{i+1}) \cup (U_{i+1} - U_i)$ belongs to Λ. We provide polynomial-time algorithms for our two problem under this generalization, assuming only "black-box" access to f and an "oracle" that tells us, for a given L, whether $L \in \Lambda$.

We feel that "connectedness" problems of this sort represent a natural and interesting genre of combinatorial optimization problems, motivated very cleanly by evolutionary issues of the type discussed above. Our algorithms here are among the first theoretical results we are aware of in this direction, and we hope that they help to encourage further algorithmic exploration of this issue.

Overview. The remainder of the paper is organized as follows.

- In Section 3, we describe our polynomial-time algorithm to compute optimal sequences in the GC model, an efficient implementation of this algorithm, and the results of experiments we performed on target structures drawn from the Protein Data Bank (PDB).
- In Section 4, we discuss the analysis of evolutionary fitness landscapes for sequences in terms of the GC model, providing polynomial-time algorithms to determine whether the fitness function for a target structure satisfies Maynard Smith's "connectedness" criteria on optimal sequences. This builds on efficient algorithms that we develop for the analogous problems involving general submodular functions.
- In Section 5, we describe our extensions to the GC model, which still allow for the efficient computation of optimal sequences. These include classes of arbitrarily large amino acid alphabets, and models with a continuum of possible hydrophobicity values.

We begin, in Section 2, with an overview of the GC model of Sun et al. [33], which forms the initial basis for our algorithms and experiments.

2 The GC Model of Sun et al.

In order to fully define the GC model, we must specify the geometric representation of the target structure, and the fitness function Φ on the set of possible sequences. Following Sun et al. [33], the target structures will be structures of proteins whose native conformations are known; this allows for the most informative test of the computational techniques, since there is a natural "true" sequence corresponding to each such target structure. A simplified geometric representation of such a structure is obtained by constructing a sphere of the appropriate radius at the location of each non-hydrogen backbone atom, and replacing the side chain of each residue with a single "side chain bead," of radius 2Å, at a distance of 3Å along the C_α-C_β bond vector [33]. In this way, the residues in the target structure are made "uniform." (Sun et al. position the side chain bead for a glycine residue at a distance of 3Å along a vector inferred from the tetrahedral geometry of its C_α; we, on the other hand, position the glycine side chain bead at the location of the C_α. We find that the experimental results are affected very little by this decision.) For each side chain bead, one also computes the area of its solvent-accessible *contact surface* with respect to a standard 1.4Å solvent probe [32]. (For this, we used the algorithm and implementation ASC of Eisenhaber and Argos [12, 13].)

We now define the optimization function in the GC model, for a given target structure with n residues. To design a sequence, we must specify which residues in the target structure will be H (hydrophobic), and which will be P (polar); thus, we say that a *protein sequence* $\mathcal{S}$ is a sequence of n symbols, each of which is either H or P. We use $\mathcal{S}_H$ to denote the set of numbers i such that the i^{th} position in the sequence $\mathcal{S}$ is equal to H; we define $\mathcal{S}_P$ analogously. Now, the *fitness* $\Phi(\mathcal{S})$ of a sequence $\mathcal{S}$, with respect to the target structure, is a scoring function motivated by the following (partially conflicting) requirements. We would like the H residues in $\mathcal{S}$ to have low solvent-accessible surface area; we would also like H residues to be close to one another in space, so as to form a compact *hydrophobic core.* Thus one defines Φ by

$$\Phi(\mathcal{S}) = \alpha \sum_{\substack{i,j\in\mathcal{S}_H \\ i<j-2}} g(d_{ij}) + \beta \sum_{i\in\mathcal{S}_H} s_i.$$

Here s_i denotes the area of the solvent-accessible contact surface of the side chain for residue i (in Å^2), and d_{ij} denotes the distance between the side chain centers of residues i and j (in Å). g is a sigmoidal function that rewards small distances; in [33] it is defined to be $\frac{1}{1+\exp(d_{ij}-6.5)}$ for $d_{ij} \leq 6.5$Å, and 0 for $d_{ij} > 6.5$Å. Finally, $\alpha < 0$ and $\beta > 0$ are scaling parameters; in [33] they are given default values of $\alpha = -2$ and $\beta = \frac{1}{3}$. At times it is useful to consider *simplified* definitions of Φ; we can round the contact surface areas s_i to integers and define g to be a *step function*: $g(d_{ij})$ is equal to 1 if $d_{ij} \leq 6.5$Å and is equal to 0 otherwise. A simplified definition of this type can be valuable for providing a "coarser" view of the set of sequences with an affinity for the target structure.

As discussed above, the goal of the GC model is to design a sequence whose fitness value Φ is minimized (i.e. as negative as possible); we will call such a sequence *optimal.* Clearly this corresponds to constructing a sequence with many close-range H-H contacts, and very few solvent-exposed H's.

3 The Basic Algorithm and Experiments

An algorithm to find optimal sequences. Sun et al. [33] noted that there are 2^n possible amino acid sequences in the binary H/P model — too many to perform an exhaustive search — and developed a heuristic method to find sequences of good fitness based on a genetic algorithm. Their method does not provide any measure of how close the final designed sequences are to the optimal sequence(s). In this section, we present a polynomial-time algorithm to produce optimal sequences.

We begin by defining some notions from graph theory for the sake of completeness; we refer the reader to texts such as [1, 7] for further details. A *directed graph* G consists of a pair of sets: V (the *vertices*) and E (the *edges*). Each edge $e \in E$ is an ordered pair of vertices $e = (u, v)$; we call u the *tail* of e and v the *head.* We also assume that each edge has a given *capacity* c_e, which is a positive number. Let s and t be two vertices of G. An s-t *cut* in G is a partition of V into two sets, X and Y, so that $s \in X$ and $t \in Y$; we denote such a cut by the pair (X, Y). We say that an edge *crosses* a cut (X, Y) if it has its tail in X and its head in Y. The *capacity* of a cut (X, Y) is equal to the sum of the capacities

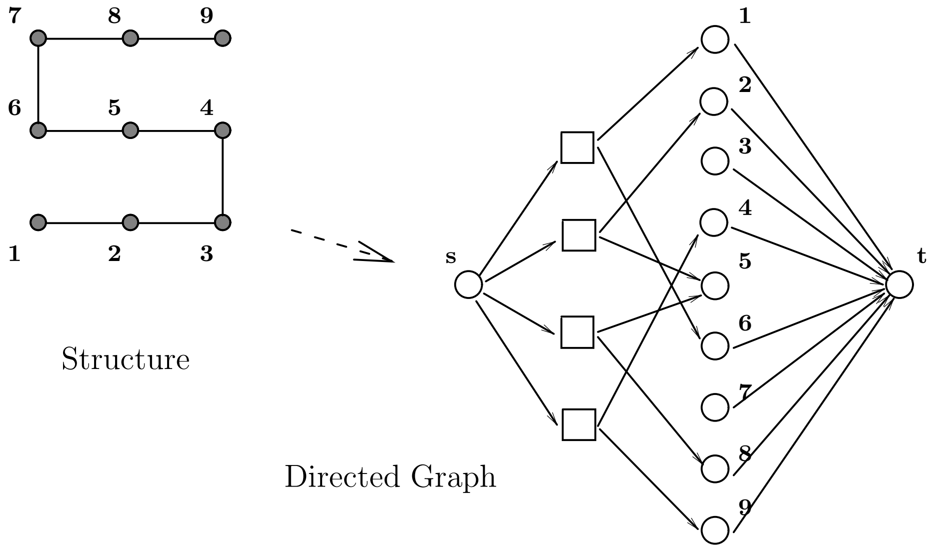

Figure 1: A small example of the construction of a directed graph from a target structure with four possible contacts (1-6, 2-5, 5-8, 4-9).

of all edges that cross (X, Y); we denote it $c(X, Y)$. The *minimum s-t cut problem* asks, for a given graph G and vertices s and t, to find an s-t cut (X, Y) of minimum capacity. Although the details are beyond the scope of this paper, the minimum s-t cut problem can be solved, for an arbitrary graph with n vertices and m edges, by algorithms whose running times are bounded by $O(mn \log n)$ [36, 15], and efficient implementations exist for some of these algorithms (e.g. [6]).

Now, let Φ be the fitness function corresponding to a given target structure of length n. Recall that the target structure determines inter-residue distances d_{ij} and solvent-exposed surface areas s_i; and that Φ is defined via a function g and parameters $\alpha < 0$ and $\beta > 0$. Let B denote the quantity $\sum_{i<j-2} |\alpha| g(d_{ij})$. We define the following graph G based on Φ. The vertex set V of G consists of s, t, a vertex v_i for each of the residue positions $i = 1, 2, \ldots, n$ in the target structure, and a vertex u_{ij} for each pair of residue positions i, j for which $i < j-2$ and $g(d_{ij}) > 0$. The edge set E of G consists of an edge (s, u_{ij}) for each vertex u_{ij}, an edge (v_i, t) for each vertex v_i which has a non-zero solvent-exposed contact surface area s_i, and edges (u_{ij}, v_i) and (u_{ij}, v_j) for each vertex u_{ij}. We refer the reader to Figure 1 for an example of the directed graph constructed by this procedure from an artificial 9-residue structure. We now assign a capacity to each edge as follows. We assign a capacity of $|\alpha| g(d_{ij})$ to the edge (s, u_{ij}), a capacity of βs_i to the edge (v_i, t), and a capacity of $B + 1$ to all edges of the form (u_{ij}, v_i) and (u_{ij}, v_j).

Let us consider the structure of the minimum s-t cut(s) in G. First, we say that a set X of vertices is *closed* if (i) X contains s but not t, and (ii) for each $u_{ij} \in V$, X contains u_{ij} if and only if it contains both v_i and v_j. We now have the following fact.

(3.1) *If (X, Y) is a minimum s-t cut in G, then X is a closed set.*

Proof. First note that G has an s-t of capacity B; in particular, consider the cut $(\{s\}, V -$

$\{s\}$). Now, consider a minimum s-t cut (X, Y) in G. Suppose X contains a vertex u_{ij} but not the vertex v_i (the case of v_j is the same). Then the edge (u_{ij}, v_i) crosses (X, Y) and it capacity $B + 1$; this contradicts the assumption that (X, Y) is a minimum cut. On the other hand, suppose that X contains some pair of vertices v_i and v_j, but not the vertex u_{ij}. Then the s-t cut $(X \cup \{u_{ij}\}, Y - \{u_{ij}\})$ would have smaller capacity than (X, Y), again a contradiction. ■

For an n-symbol H/P sequence $\mathcal{S}$, let Z denote the set of all vertices v_i for which position i in $\mathcal{S}$ is labeled H. Let $X(\mathcal{S})$ denote the closed set consisting of s, the vertices in Z, and all vertices u_{ij} for which $v_i, v_j \in Z$. Conversely, if X is a closed set, let $\mathcal{S}(X)$ denote the H/P sequence in which position i is labeled H if v_i belongs to X, and is labeled P if v_i does not belong to X. From these constructions, we see that there is a one-to-one correspondence between n-symbol H/P sequences and closed sets in G. Now we come to the crucial fact about G.

(3.2) *Let X be a closed set and $\mathcal{S}(X)$ the corresponding H/P sequence. Then the capacity of the s-t cut $(X, V - X)$ is equal to $B + \Phi(\mathcal{S}(X))$.*

Proof. From the definition of *closed set*, we know that the only edges crossing (X, Y) have the form (v_i, t), where $v_i \in X$, or (s, u_{ij}), where one of v_i or v_j does not belong to X. Thus,

$$\begin{aligned} c(X, V - X) &= \sum_{\substack{u_{ij} \in V \\ \{v_i, v_j\} \not\subseteq X}} |\alpha| g(d_{ij}) + \sum_{v_i \in X} \beta s_i \\ &= B - \sum_{\substack{u_{ij} \in V \\ \{v_i, v_j\} \subseteq X}} |\alpha| g(d_{ij}) + \sum_{v_i \in X} \beta s_i \\ &= B + \alpha \sum_{\substack{i < j-2 \\ i,j \in \mathcal{S}(X)_H}} g(d_{ij}) + \beta \sum_{i \in \mathcal{S}(X)_H} s_i. \\ &= B + \Phi(\mathcal{S}(X)). \end{aligned}$$

■

Thus the fitness of an H/P sequence for the target structure differs from capacity of the corresponding cut in G simply by the fixed additive constant B. Consequently, if (X, Y) is a minimum capacity s-t cut in G, then $\mathcal{S}(X)$ is an optimal sequence — so to find an optimal sequence for the target structure, we need only construct the graph G and compute a minimum capacity s-t cut in it. Let p denote the number of residue pairs (i, j) in the target structure for which $g(d_{ij}) > 0$; since the graph G has $O(n + p)$ vertices and edges, we have

(3.3) *An optimal sequence in the GC model can be computed in time $O((n+p)^2 \log(n+p))$.*

Excluded-volume constraints in three dimensions imply that for each residue i, there will only be a small constant number of other residues j for which $g(d_{ij}) > 0$. Thus, p can be assumed to be proportional to n and hence the running time of the algorithm is $O(n^2 \log n)$ — roughly quadratic in the length of the sequence, rather than exponential.

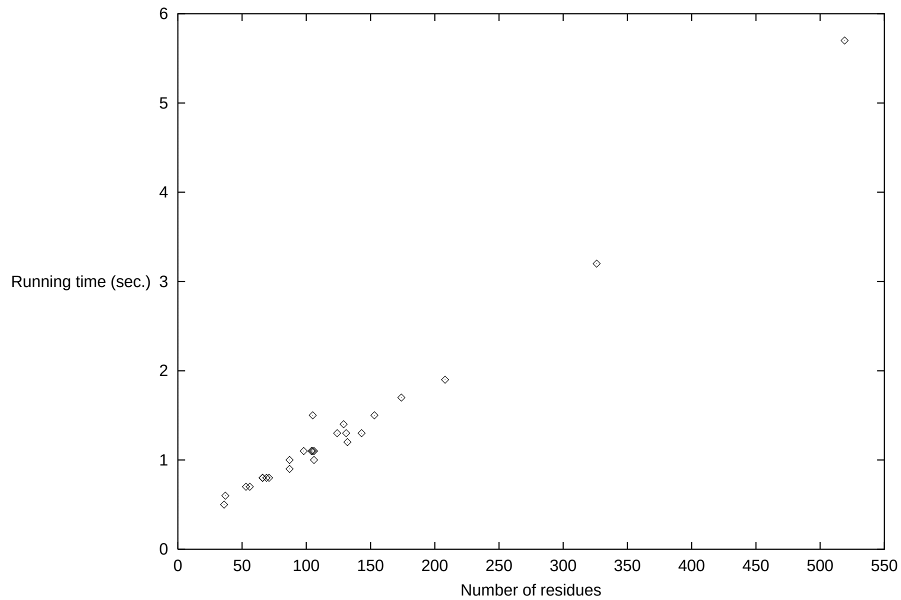

Figure 2: Running time as a function of sequence length

Experiments with PDB structures. We implemented the above algorithm, making use of the highly efficient code of Cherkassky and Goldberg for computing the minimum s-t cut in a graph [6]. We tested the implementation on the 23 PDB structures considered by Sun et al. [33], as well as on two larger protein structures — pepsin (326 residues) and pyruvate kinase (519 residues). The running times of the algorithm (in CPU seconds) on a Sun Sparc 10 are depicted in Figure 2; for the structures of lengths 36–208 considered in [33], the running times ranged from 0.5 to 1.9 seconds; for the largest structure (519 residues), the running time was 5.7 seconds.

An advantage of testing the algorithm on real proteins is that we can compare the sequences we design we produce to the true sequence of the proteins, as in [33, 28]; this is a way to assess the biological relevance of the GC model. For a protein structure from the PDB, let us define its *natural H/P sequence* to be the one obtained by translating the protein's true amino acid sequence into an H/P sequence, according to a designation of each of the twenty amino acids as either hydrophobic or hydrophilic. (Following Sun et al., we map the amino acids A,C,F,I,L,M,V,W,Y to H, and the others to P.) Since the fitness function Φ associated with the model is designed only to approximate the factors favoring the *natural sequence*, the natural sequence is likely to be *sub-optimal* when scored according to Φ with respect to its structure; correspondingly, the optimal sequence under Φ may differ non-trivially from the natural sequence.

In Figure 3, we compute the percentage agreement between the natural and optimal sequences for the 23 structures from [33] (in the column under "Basic Algorithm"); we also reproduce the numbers of Sun et al. for the sake of comparison. It is interesting to note the markedly varied way in which the percentages of overlap change, for different structures, as we move from heuristically designed sequences to the optimal sequences. This is in keeping with the observation above that *optimality* in the GC model is not the same as achieving

Protein (PDB code)	sequence length	Sun et al. algorithm	Basic algorithm	Scaling algorithm
1aaj	105	66	73	72
1aba	87	81	75	70
1aps	98	72	78	77
1arr (mon)	53	62	62	64
1arr (dim)	106	73	71	74
1bba	36	58	58	58
1bbl	37	68	57	68
1bov	69	74	74	70
1brq	174	68	74	71
1cis	66	64	68	64
1cmb (mon)	104	62	63	64
1cmb (dim)	208	70	74	73
1hel	129	74	79	78
1ifb	131	76	69	70
1kba (mon)	66	72	68	76
1kba (dim)	132	73	77	74
2gb1	56	80	79	70
2hpr	87	78	78	78
2il8	71	77	72	79
256b	106	81	77	75
3cln	143	62	72	70
3rn3	124	81	69	69
3trx	105	80	77	81
Length-weighted average		72.1	72.6	72.2

Figure 3: Results for sequences

identity with the natural sequence. For certain of the proteins, the percentage agreement jumped considerably when the designed sequence was computed optimally. For example, the percentage agreement for calmodulin (3cln) increased from Sun et al.'s value of 62% to 72% for our optimal sequence. It is interesting to note that Sun et al. had conjectured the low level of agreement for calmodulin, relative to most of the other structures studied, to be due to intrinsic aspects of its structure. On the other hand, certain structures, such as ribonuclease A (3rn3), showed significantly less agreement with the natural sequence when solved to optimality.

In a "dual" fashion, we can test whether natural sequences have significantly better fitness scores than random sequences with the same ratio of H to P residues. We investigate this in Figure 4, which shows the score obtained by the natural sequence when plugged into the GC optimization function Φ. This is depicted in the second column, and compared with the average results for 100 randomly generated H/P sequences of the same composition in the third column. The scores are linearly normalized so that the fitness value of the optimal sequence is mapped to 1, while the "baseline" quantity $\beta \sum_{i=1}^{n} s_i$ is mapped to 0. For the ensembles of 100 random structures, the standard deviations for almost all are concentrated in the interval $[0.04, 0.065]$. Thus in all cases but one (the short bovine pancreatic polypeptide

Protein (PDB code)	Basic alg., natural seq.	Basic alg., random seq.	Scaling alg., natural seq.	Scaling alg., random seq.
1aaj	0.72	0.60	0.73	0.59
1aba	0.82	0.60	0.83	0.58
1aps	0.83	0.59	0.83	0.58
1arr (mon)	0.68	0.61	0.73	0.63
1arr (dim)	0.77	0.62	0.82	0.60
1bba	0.59	0.61	0.64	0.65
1bbl	0.73	0.62	0.84	0.67
1bov	0.77	0.57	0.77	0.55
1brq	0.77	0.60	0.76	0.58
1cis	0.70	0.57	0.71	0.55
1cmb (mon)	0.73	0.61	0.72	0.58
1cmb (dim)	0.82	0.59	0.83	0.60
1hel	0.84	0.55	0.86	0.58
1ifb	0.78	0.64	0.85	0.66
1kba (mon)	0.78	0.61	0.79	0.61
1kba (dim)	0.81	0.60	0.80	0.59
256b	0.85	0.59	0.84	0.57
2gb1	0.79	0.60	0.82	0.58
2hpr	0.84	0.57	0.84	0.57
2il8	0.81	0.61	0.85	0.60
3cln	0.81	0.61	0.85	0.60
3rn3	0.77	0.58	0.76	0.56
3trx	0.82	0.61	0.83	0.60

Figure 4: Objective function scores of true and random HP sequences

1bba, for which the optimal sequence assumed all P values), the natural sequence scores significantly better than the ensemble of random H/P sequences.

Finally, we can study the relation between the optimal and natural sequences at other levels as well. In Figure 5, we show the percentage agreement on the 23 sample structures, organized by amino acid type. That is — over all target structure positions occupied by a given amino acid, what was the percentage agreement between the natural and optimal structures? It is clear that the GC model produces much better agreement for some amino acids than for others — some notable patterns are that agreement is markedly better for polar residue positions than for non-polar positions, and best for acidic and basic residues such as arginine, lysine, and asparagine. Indeed, alanine and methionine positions were classified as polar more than half the time. One possible explanation for the lower level of agreement among non-polar residue positions, parallel to observations of several previous authors [33, 34], is the recurring presence of exposed non-polar residues on protein surfaces for reasons of biological function, something that the simple optimization function of the GC model does not take into account.

The Scaling Algorithm. Certain of the optimal sequences constructed have a sharp imbalance in the ratio of H to P residues, and in most cases this *a fortiori* prevents them from having a high degree of agreement with the natural sequence. We now describe an

Amino Acid	Basic algorithm (% agreement)	Scaling algorithm (% agreement)
ala	42	47
arg	93	78
asn	90	85
asp	89	82
cys	73	87
gln	75	62
glu	89	86
gly	75	60
his	70	59
ile	61	68
leu	53	71
lys	92	87
met	42	56
phe	57	76
pro	69	54
ser	83	78
thr	84	74
trp	62	76
tyr	55	70
val	56	67

Figure 5: Results by amino acid type

extension of our basic algorithm — the *Scaling Algorithm* — which attempts to construct a sequence in which the ratio of H to P residues is roughly 2/3, matching the relative frequencies of the corresponding amino acids in naturally occurring polypeptide sequences [8].

The relative values of the parameters α and β in the fitness function Φ control the relative proportions of H and P residues in an optimal sequence. Qualitatively, one can see this as follows: as α is made increasingly negative, for fixed β, there is an increasing reward for hydrophobic contacts; as β is made increasingly positive, for fixed α, there is an increasing penalty for solvent-exposed hydrophobic residues. At a more concrete level, we can prove that the minimum number of H residues in an optimal sequence increases monotonically as β is held fixed and α is made increasingly negative.

The Scaling Algorithm, then, simply uses repeated calls to the basic algorithm described above in order to determine the value of α (with $\beta = \frac{1}{3}$) for which the optimal sequence has approximately the appropriate fraction of H residues. We note that a formulation of the sequence design problem due to Shakhnovich and Gutin [34] directly imposes the constraint of a fixed ratio of H residues to P residues, leading to an optimization problem different from what we consider here. In our case, rather than performing an optimization with the value of this ratio imposed as an explicit constraint, we attempt to achieve the ratio indirectly by varying the parameters in the GC model.

4 Evolutionary Fitness Landscapes

The global connectivity problem. We begin by recalling the discussion from the introduction. We are given a fitness function Φ defined by an n-residue target structure in the basic GC model, and we let Ω denote the set of all optimal sequences. One can construct examples in which Ω has size exponential in n; and when we use a simplified definition for Φ as described in Section 2, the set Ω often turns out to be relatively large. The first algorithmic problem we wish to solve is that of determining whether Ω is *connected* under single-point mutations: for any pair of sequences $\mathcal{S}, \mathcal{S}' \in \Omega$, is there a way to transform $\mathcal{S}$ into $\mathcal{S}'$ by flipping the value of one residue position at a time, so that each intermediate sequence in this transformation lies in Ω?

We first rephrase the problem as follows. For a set $X \subseteq \{1, \ldots, n\}$, let $\sigma(X)$ denote the sequence $\mathcal{S}$ for which $\mathcal{S}_H = X$; that is, X denotes precisely the positions at which $\mathcal{S}$ has H residues. Let f be a function that maps subsets of $\{1, \ldots, n\}$ to real numbers, defined by the equation $f(X) = \Phi(\sigma(X))$. It is not difficult to show that f satisfies the following property.

(4.1) *For all sets X and Y, $f(X \cap Y) + f(X \cup Y) \leq f(X) + f(Y)$.*

Functions satisfying (4.1) are called *submodular*.

We say that two sets $X, X' \subseteq \{1, \ldots, n\}$ are *adjacent* if X differs from X' by the insertion or deletion of precisely one element. We say that a sequence of sets $\mathcal{C} = \{X_1, X_2, \ldots, X_t\}$ is a *chain between X_1 and X_t* (briefly, an X_1-X_t chain) if for each i, X_i and X_{i+1} are adjacent. Now, two sequences $\mathcal{S}$ and $\mathcal{S}'$ differ by a one-point mutation if and only if the sets $\mathcal{S}_H$ and $\mathcal{S}'_H$ are adjacent. Moreover, if we let Ω_f denote the collection of subsets of $\{1, \ldots, n\}$ on which f attains its minimum value, then we see that a sequence $\mathcal{S}$ belongs to Ω if and only if $\mathcal{S}_H \in \Omega_f$. Thus, we have shown that our *global connectivity problem* is equivalent to the following "connectedness" problem for f:

> (†) Is it the case that for all pairs of sets $X, X' \in \Omega_f$, there is an X-X' chain contained in Ω_f?

We now show how to solve problem (†) in polynomial time for an *arbitrary* submodular function f. We will assume that f is specified simply by an "oracle" that, in response to a set X, returns $f(X)$. The development of the algorithm involves a sequence of combinatorial lemmas, beginning with two standard facts about submodular functions. The first is a direct consequence of the submodular property.

(4.2) *If $X, Y \in \Omega_f$, then $X \cap Y \in \Omega_f$ and $X \cup Y \in \Omega_f$.*

From (4.2), we obtain a second basic fact.

(4.3) *There exist unique sets $X_*, X^* \in \Omega_f$ with the property that for all $Y \in \Omega_f$, $X_* \subseteq Y \subseteq X^*$.*

We note that standard algorithms for *submodular function minimization* can produce the sets X_* and X^* in polynomial time; see e.g. [16, 29]. We will use `Minimal(`f`)` to denote the polynomial-time algorithm computing X_*, and `Maximal(`f`)` to denote the polynomial-time algorithm computing X^*.

We say that a chain $X_1, \ldots, X_t$ is *monotone* if $X_i \subseteq X_{i+1}$ for each i. We now state the main lemma that will form the basis of the algorithm.

(4.4) *Let $X, Y, Z \in \Omega_f$ have the property that $X \subseteq Y \subseteq Z$. If there is a monotone X-Z chain in Ω_f, then there is a monotone X-Y chain in Ω_f.*

Proof. Let $\mathcal{C} = X_1, \ldots, X_t$ be a monotone chain in Ω_f with $X_1 = X$ and $X_t = Z$. Consider the sequence $\mathcal{C}' = Y_1, \ldots, Y_s$ obtained by removing duplicates from the chain $X_1 \cap Y, \ldots, X_t \cap Y$. For any positive $i < s$, there is a j so that $Y_i = X_j \cap Y$ and $Y_{i+1} = X_{j+1} \cap Y$; since $\mathcal{C}$ is a monotone chain and $Y_i \neq Y_{i+1}$, it follows that Y_{i+1} is obtained from Y_i by the addition of precisely one element. It follows that $\mathcal{C}'$ is a monotone chain. Since $Y_1 = X_1 \cap Y = X$ and $Y_s = X_t \cap Y = Y$, $\mathcal{C}'$ is an X-Y chain. Finally, we wish to show that $\mathcal{C}'$ lies in Ω_f. For each Y_i, there is a j so that $Y_i = X_j \cap Y$; since $X_j, Y \in \Omega_f$, it follows from (4.2) that $Y_i \in \Omega_f$. ∎

As a first consequence of (4.4), we have the following.

(4.5) *Let $Y \in \Omega_f$. If there is an X_*-Y chain in Ω_f, then there is a monotone X_*-Y chain in Ω_f.*

Proof. Consider an X_*-Y chain $\mathcal{C}$ in Ω_f with a minimum number of elements, and suppose $\mathcal{C}$ is not monotone. Consider the maximal prefix $\mathcal{C}'$ of $\mathcal{C}$ that is monotone; this is a chain between X_* and some set $X_i \supseteq X_*$. By the maximality of $\mathcal{C}'$, we know that $X_{i+1} \subseteq X_i$. Let $\mathcal{C}_1$ denote the chain $X_{i+1}, X_{i+2}, \ldots, Y$. By (4.4), there is a monotone X_*-X_{i+1} chain $\mathcal{C}_0$ in Ω_f; but then the concatenation of $\mathcal{C}_0$ and $\mathcal{C}_1$ is an X_*-Y chain in Ω_f with fewer elements than $\mathcal{C}$, a contradiction. ∎

Combining (4.4) and (4.5), we obtain

(4.6) *Ω_f is connected if and only if there is a monotone X_*-X^* chain in Ω_f.*

This already has some interesting consequences; for example, if Ω_f is connected, then there is a short proof of this fact. However, it does not yet provide us with a polynomial-time algorithm to test whether Ω_f is connected. For that we require one more notion.

We say that a set $X \in \Omega_f$ is an *impasse* if $X \neq X_*$, and for every X' obtained from X by the deletion of one element, $X' \notin \Omega_f$. Using (4.5), we can show

(4.7) *Ω_f is connected if and only if it contains no impasse.*

Proof. If Ω_f is connected, then for every $X \in \Omega_f$, there is an X_*-X chain in Ω_f, and hence a monotone X_*-X chain in Ω_f. It follows that no $X \in \Omega_f$ is an impasse. Conversely, suppose Ω_f is not connected, and choose a set $X \in \Omega_f$ that is minimal subject to the property that there is no X_*-X chain in Ω_f. We claim that X is an impasse; for if there is an $X' \in \Omega_f$ that can be obtained by deleting an element from X, then the minimality of X would imply that there is an X_*-X' chain in Ω_f, and hence an X_*-X chain in Ω_f. ∎

Just as (4.6) provided a short proof of the connectedness of Ω_f, (4.7) provides a short proof of the non-connectedness of Ω_f. Together, they show the following algorithm correctly decides if Ω_f is connected.

```
Compute X_* = Minimal(f) and X^* = Maximal(f)
Set W := X^*.
While W ≠ X_*
     Determine whether there exists i ∈ W
          so that f(W − {i}) = f(W)
          (whence W − {i} ∈ Ω_f).
     If there is such an i then
          Update W := W − {i} and iterate.
     If there is no such i then
          W is an impasse;
          Halt and declare that Ω_f is not connected.
end while
Halt and declare that Ω_f is connected.
```

The correctness of the algorithm follows directly from the fact that it produces either an impasse or a monotone X_*-X^* chain. After the determination of the sets X_* and X^*, the algorithm performs at most n^2 evaluations of the function f.

Global connectivity with a more general model of mutations. We have so far discusssed the connectedness problem for Ω_f under the assumption that a "unit step" consists of the mutation of a single position in the sequence. Now consider, more generally, the case in which a unit step could involve the simultaneous mutation of several positions in the sequence. We will suppose that we are given a collection Λ consisting of subsets of the sequence positions $\{1,\ldots,n\}$; if a set $L \subseteq \{1,\ldots,n\}$ belongs Λ, we interpret this to mean that all the sequence positions in L could undergo a mutation in a single unit step. The only assumption we will make about Λ is that it is *downward-closed*: if $L \in \Lambda$ and $L' \subseteq L$, then $L' \in \Lambda$. Thus, if Λ consists of the empty set and all single-element subsets of $\{1,\ldots,n\}$, we have the problem (†) considered above; if Λ consists of all sets of size at most c, we have the problem in which any c positions can simultaneously undergo a mutation.

Now, two sequences $\mathcal{S}$ and $\mathcal{S}'$ differ by the simultaneous mutation of a set L of sequence positions if and only if for the sets $\mathcal{S}_H$ and $\mathcal{S}'_H$ we have $(\mathcal{S}_H - \mathcal{S}'_H) \cup (\mathcal{S}'_H - \mathcal{S}_H) = L$. Thus, we can define our problem in the following way. For two sets $X, X' \subseteq \{1,\ldots,n\}$, we use $X \Delta X'$ to denote the *symmetric difference* $(X - X') \cup (X' - X)$ — the collection of elements in one set but not the other. We say that X and X' are Λ-*adjacent* if $X \Delta X' \in \Lambda$. We say that a sequence of sets $\mathcal{C} = \{X_1, X_2, \ldots, X_t\}$ is a Λ-*chain between* X_1 *and* X_t (briefly, an X_1-X_t Λ-chain) if for each i, X_i and X_{i+1} are Λ-adjacent. Note that since $\phi \in \Lambda$, this allows $X_i = X_{i+1}$ for certain i. Our general connectedness question can now be phrased as follows.

> (†′) Is it the case that for all pairs of sets $X, X' \in \Omega_f$, there is an X-X' Λ-chain contained in Ω_f?

If there is such a chain for all pairs $X, X' \in \Omega_f$, we will say that Ω_f is *connected under* Λ-*adjacency*.

We now show how to solve problem (†′) in polynomial time, assuming only an oracle for f as before, and a second oracle that tells us, for a given L, whether $L \in \Lambda$. The development of the polynomial-time algorithm will proceed roughly as follows. It will turn out that many of the structural properties we established about adjacency carry over to Λ-adjacency, for a general downward-closed collection Λ. In particular, the notion of an *impasse* carries over. However, since Λ is only specified by an oracle, and may have exponential size, it requires more work to determine in polynomial time whether a given set in Ω_f is an impasse. Once we do this, however, we can apply an algorithm that is very similar in structure to the one we developed above.

We say that a chain $X_1, \ldots, X_t$ is *graded* if for each i we have either $X_i \subseteq X_{i+1}$ or $X_{i+1} \subseteq X_i$.

(4.8) *Let $X, Y \in \Omega_f$. If there is an X-Y Λ-chain, then there is a graded X-Y Λ-chain.*

Proof. Let $\mathcal{C} = X_1, \ldots, X_t$ be a chain in Ω_f with $X_1 = X$ and $X_t = Y$. We produce a new chain $\mathcal{C}'$ by inserting, between each pair of consecutive sets X_i and X_{i+1}, the set $X_i \cap X_{i+1}$. By (4.2), $X_i \cap X_{i+1} \in \Omega_f$; since

$$X_i \Delta (X_i \cap X_{i+1}) = X_i - X_{i+1} \subseteq X_i \Delta X_{i+1} \in \Lambda$$

(and analogously for X_{i+1}), $\mathcal{C}'$ is a Λ-chain. ■

We now establish the analogues of (4.4) and (4.5) for Λ-chains.

(4.9) *Let $X, Y, Z \in \Omega_f$ have the property that $X \subseteq Y \subseteq Z$. If there is a monotone X-Z Λ-chain $\mathcal{C}$ in Ω_f, then there is a monotone X-Y Λ-chain $\mathcal{C}'$ in Ω_f such that $\mathcal{C}'$ contains at most as many elements as $\mathcal{C}$.*

Proof. Let $\mathcal{C} = X_1, \ldots, X_t$ be a monotone Λ-chain in Ω_f with $X_1 = X$ and $X_t = Z$. Consider the sequence $\mathcal{C}' = Y_1, \ldots, Y_s$ defined by $Y_i = X_i \cap Y$; note that $Y_1 = X$ and $Y_t = Y \cap Z = Y$. We have $Y_i \subseteq Y_{i+1}$ for each i, and

$$Y_i \Delta Y_{i+1} = Y_{i+1} - Y_i \subseteq X_{i+1} - X_i = X_i \Delta X_{i+1} \in \Lambda,$$

so $\mathcal{C}'$ is a monotone X-Y Λ-chain. Finally, $\mathcal{C}'$ lies in Ω_f since for each i, $Y_i = X_i \cap Y$ and $X_i, Y \in \Omega_f$. ■

(4.10) *Let $Y \in \Omega_f$. If there is an X_*-Y Λ-chain in Ω_f, then there is a monotone X_*-Y Λ-chain in Ω_f.*

Proof. Since there is an X_*-Y Λ-chain in Ω_f, there is a graded one by (4.8); let us choose a graded X_*-Y Λ-chain $\mathcal{C}$ in Ω_f with a minimum number of elements. Suppose $\mathcal{C}$ is not monotone. Consider the maximal prefix $\mathcal{C}'$ of $\mathcal{C}$ that is monotone; this is a chain between X_* and some set $X_i \supseteq X_*$. By the maximality of $\mathcal{C}'$, and the fact that $\mathcal{C}$ is graded, we know that $X_{i+1} \subseteq X_i$. Let $\mathcal{C}_1$ denote the chain $X_{i+1}, X_{i+2}, \ldots, Y$. By (4.9), there is a monotone X_*-X_{i+1} chain $\mathcal{C}_0$ in Ω_f such that $\mathcal{C}_0$ has at most as many elements as $\mathcal{C}'$; but then the concatenation of $\mathcal{C}_0$ and $\mathcal{C}_1$ is an X_*-Y chain in Ω_f with fewer elements than $\mathcal{C}$, a contradiction. ■

We define a Λ-*impasse* to be a set $X \in \Omega_f$ such that $X \neq X_*$, and $X - L \notin \Omega_f$ for every non-empty set $L \in \Lambda$. Using (4.9) and (4.10), we obtain the analogues of (4.6) and (4.7) with essentially the same proofs.

(4.11) *Ω_f is connected under Λ-adjacency if and only if there is a monotone X_*-X^* Λ-chain in Ω_f.*

(4.12) *Ω_f is connected under Λ-adjacency if and only if it contains no Λ-impasse.*

The problem we now encounter is that Λ is only defined implicitly through an oracle, and hence it is not immediately clear how to decide in polynomial time whether a set X is a Λ-impasse — a crucial component of our earlier algorithm for determining connectivity. We now show how to solve this problem in polynomial time.

Let X be a set in $\Omega_f - \{X_*\}$. We say that a collection of sets $\{Y_1, \ldots, Y_t\}$ *supports* X if (i) $Y_j \in \Omega_f$ and $Y_j \subsetneq X$ for each j, and (ii) for all $Y' \in \Omega_f$ satisfying $Y' \subsetneq X$, there is a j such that $Y' \subseteq Y_j$.

(4.13) *For every $X \in \Omega_f - \{X_*\}$, there is a collection of at most n sets that supports X, and it can be constructed in polynomial time.*

Proof. For each $j \in X - X_*$, let f^j denote the restriction of f to the set $X - \{j\}$; that is, while f^j assumes the same values as f, we view it as a submodular function mapping subsets of $X - \{j\}$ to real numbers. Note that for each such j we have $X_* \subseteq X - \{j\}$, and hence the minimum value of f^j over $X - \{j\}$ is equal to the minimum value of f over $\{1, \ldots, n\}$. Let $Y_j = \texttt{Maximal}(f^j)$.

Let $\mathcal{F} = \{Y_j : j \in X - X_*\}$; we claim that $\mathcal{F}$ supports X. Clearly each $Y_j \in \mathcal{F}$ belongs to Ω_f and satisfies $Y_j \subsetneq X$. Now consider any $Y' \in \Omega_f$ satisfying $Y' \subsetneq X$. There is some $k \in X$ such that $Y' \subseteq X - \{k\}$; since $Y' \in \Omega_f$, we know by (4.3) that $k \notin X_*$, and so $Y' \subseteq Y_k \in \mathcal{F}$. It follows that $\mathcal{F}$ supports X.

By computing $\texttt{Maximal}(f^j)$, for each $j \in X - X_*$, we can obtain $\mathcal{F}$ in polynomial time. ∎

(4.14) *Let $X \in \Omega_f - \{X_*\}$, and let $\mathcal{F}$ be a collection of sets that supports X. X is a Λ-impasse if and only if $X - Y \notin \Lambda$ for each $Y \in \mathcal{F}$.*

Proof. If X is a Λ-impasse, then $X - Y \notin \Lambda$ for all $Y \in \Omega_f$ and hence specifically for all $Y \in \mathcal{F}$. Conversely, suppose that there exists $X' \in \Omega_f$, $X' \subsetneq X$, for which $X - X' \in \Lambda$. Since $\mathcal{F}$ supports X, there is a $Y' \in \mathcal{F}$ such that $X' \subseteq Y'$; thus $X - Y' \subseteq X - X'$ and hence $X - Y' \in \Lambda$. ∎

Combining (4.13) and (4.14), we obtain

(4.15) *Let $X \in \Omega_f - \{X_*\}$. There is a polynomial-time algorithm that either returns a non-empty set $L \in \Lambda$ so that $X - L \in \Omega_f$, or reports (correctly) that X is a Λ-impasse.*

Finally, combining (4.11), (4.12), and (4.15), we obtain the following polynomial-time algorithm for determining whether Ω_f is connected under Λ-adjacency.

```
Compute X_* = Minimal(f) and X^* = Maximal(f)
Set W := X^*.
While W ≠ X_*
     Determine if there is a non-empty set L ∈ Λ
          so that W − L ∈ Ω_f.
     If there is such an L then
          Update W := W − L and iterate.
     If there is no such L then
          W is a Λ-impasse;
          Halt and declare that Ω_f is not connected.
end while
Halt and declare that Ω_f is connected.
```

The pairwise connectivity problem. We now consider the problem of determining whether there is a path of mutations between two *given* sequences $\mathcal{S}, \mathcal{S}' \in \Omega$, so that all intermediate sequences lie in Ω as well. Following the discussion at the beginning of this section, we can re-phrase this *pairwise connectivity problem* in terms of the submodular function f, and family Λ of allowable mutations: We are asking whether there is a Λ-chain between two given sets X and X' that is contained in Ω_f. If there is such a chain, we say that X and X' belong to the same *component* of Ω_f, with respect to Λ. We now give a polynomial-time algorithm for deciding whether two sets in Ω_f belong to the same component, and for constructing an appropriate chain between them if they do.

We say that a set $X \in \Omega_f$ is a Λ-*cusp* if $X - L \notin \Omega_f$ for every non-empty set $L \in \Lambda$. Thus, X is a Λ-cusp if and only if X is a Λ-impasse, or $X = X_*$. Hence, by (4.15), we have

(4.16) *Let $X \in \Omega_f$. There is a polynomial-time algorithm that either returns a non-empty set $L \in \Lambda$ so that $X - L \in \Omega_f$, or reports (correctly) that X is a Λ-cusp.*

We begin by showing that each component contains a unique Λ-cusp.

(4.17) *Let $\Gamma \subseteq \Omega_f$ be a component of Ω_f. Then there is a unique Λ-cusp in Γ.*

Proof. There is clearly at least one Λ-cusp in Γ — take any set in Γ of minimum cardinality. So suppose by way of contradiction that X and Y were two distinct Λ-cusps in Γ. Since X and Y belong to the same component, there is a Λ-chain $X_1, \ldots, X_t$ in Ω_f with $X_1 = X$ and $X_t = Y$. For $i = 1, \ldots, t$, define $Z_i = \cap_{j=1}^{i} X_j$. Then we have $X = Z_1 \supseteq Z_2 \supseteq \cdots \supseteq Z_t$; and by (4.2), each Z_i belongs to Ω_f. Choose the minimum index k for which $Z_k \subsetneq X$, and consider the set $L = X - Z_k$. We have $X - L = Z_k \in \Omega_f$. By our choice of k, L is non-empty, and

$$L = \left(\cap_{j=1}^{k-1} X_j\right) - \left(\cap_{j=1}^{k} X_j\right) \subseteq X_{k-1} - X_k \subseteq X_{k-1} \Delta X_k \in \Lambda.$$

This contradicts our assumption that X is a Λ-cusp. ■

For a set $X \in \Omega_f$, let $\gamma(X)$ denote the unique Λ-cusp of the component of Ω_f that contains X. The following statement is a direct consequence of this definition.

(4.18) *Let $X, X' \in \Omega_f$. X and X' belong to the same component if and only if $\gamma(X) = \gamma(X')$.*

Now, given X and X' in Ω_f, our approach to deciding whether they belong to the same component is as follows. We first compute $\gamma(X)$ and $\gamma(X')$ in polynomial time, together with Λ-chains $\mathcal{C}$ and $\mathcal{C}'$ joining X and X' to their respective cusps; we show how to accomplish this below. If $\gamma(X) = \gamma(X')$, then X and X' belong to the same component, and we can return the concatenation of $\mathcal{C}$ and $\mathcal{C}'$ as a Λ-chain from X to X'. If $\gamma(X) \neq \gamma(X')$, then we can correctly report that X and X' belong to different components.

Thus all that remains to show is how, given $X \in \Omega_f$, to compute a Λ-chain from X to $\gamma(X)$. For this, we employ the following algorithm.

```
Set W := X.
Repeat:
     Determine if there is a non-empty set L ∈ Λ
          so that W − L ∈ Ω_f.
     If there is such an L then
          Update W := W − L and iterate.
     If there is no such L then
          W is a Λ-cusp;
          Return the sequence of sets constructed from X to W.
end repeat
```

The cardinality of the set W strictly decreases in each iteration, so the algorithm halts in at most n iterations. Since each iteration takes polynomial time by (4.16), the overall running time is polynomial. When the algorithm halts, the resulting set W is a Λ-cusp reachable by a Λ-chain from X; thus it must indeed be $\gamma(X)$.

5 Extensions to the GC Model

Fractional Hydrophobicity. In the standard GC model, each residue position in a sequence is either entirely hydrophobic (H) or entirely polar (P). Suppose instead that we allowed each residue position i to specify a *hydrophobicity value* z_i, where z_i is an arbitrary real number in the interval $[0, 1]$. Thus, a *protein sequence* in this model would be a sequence $\mathcal{S}'$ of n real numbers, each between 0 and 1. The penalty for exposing residue i to solvent could be scaled by the hydrophobicity z_i, and the reward for a pairwise hydrophobic contact between i and j could be scaled by a product of the form $z_i z_j$. Making these notions concrete, we could define the fitness of a sequence $\mathcal{S}'$ as

$$\Phi'(\mathcal{S}') = \alpha \sum_{i<j-2} z_i z_j g(d_{ij}) + \beta \sum_{i \in \mathcal{S}_H} z_i s_i.$$

Note the standard GC model is precisely the case in which we require each z_i to be either 0 or 1.

One might hope that this generalization would provide an interesting contrast to the discrete H/P model. But in fact, we are able to show the following somewhat surprising result.

(5.1) *For any target structure, with associated fitness function Φ', there exists an optimal sequence $\mathcal{S}'$ in which each z_i takes the value 0 or 1.*

Proof. Consider a fitness function Φ' in the fractional model, and let $\mathcal{S}'$ be an optimal sequence in which the number of residues i with $0 < z_i < 1$ is minimum. We claim that in $\mathcal{S}'$, each z_i is equal to 0 or 1. For suppose not, and choose j so that $0 < z_j < 1$. For $y \in [0,1]$, let $\mathcal{S}'_y$ denote the sequence obtained from $\mathcal{S}'$ by changing the hydrophobicity value for residue j to y. Now define a function $\ell : [0,1] \rightarrow \mathbf{R}$ by $\ell(y) = \Phi'(\mathcal{S}'_y)$. The optimality of $\mathcal{S}'$ implies that $\ell(0) \geq \ell(z_j)$ and $\ell(1) \geq \ell(z_j)$. But ℓ is a linear function, so this implies that $\ell(0) = \ell(z_j) = \ell(1)$. Hence the sequence $\mathcal{S}'_0$ is also optimal, and it has fewer residues i with $0 < z_i < 1$, a contradiction. ■

In other words, there is always an optimal sequence in this *fractional* model that is in fact just an H/P sequence.

Larger Finite Amino Acid Alphabets. The previous result shows that a straightforward "interpolation" of the H/P alphabet by real-valued hydrophobicities does not really produce a new model. However, it is possible to produce models, with finite alphabets of size greater than 2, that do exhibit behavior different from that of the basic GC model with an H/P alphabet.

Let us suppose we wish to define a sequence design model over an amino acid alphabet $\{a_0, a_1, \ldots, a_k\}$, where a_0 will be designated as the most polar residue type. We first define *solvation parameters* $\{\delta_i : 0 \leq i \leq k\}$ that indicate the penalty for exposing each residue type a_i to solvent. We will require that $\delta_0 = 0$ and $\delta_i > 0$ for all i. We then define *contact parameters* $\{\varepsilon_{ij} : 0 \leq i, j \leq k\}$ that indicate the reward for having a contact between residues of type a_i and a_j. We will require that $\varepsilon_{ij} = \varepsilon_{ji} \geq 0$ for all i, j, and $\varepsilon_{0i} = 0$ for each i. Now, in this model, a *protein sequence* $\mathcal{S}''$ consists of a sequence of n numbers $\{t_1, t_2, \ldots, t_n\}$ with each $t_i \in \{0, 1, \ldots, k\}$; the meaning here is that residue i in $\mathcal{S}''$ has amino acid type a_{t_i}. The fitness of $\mathcal{S}''$ is then

$$\Phi''(\mathcal{S}'') = \alpha \sum_{i<j-2} \varepsilon_{t_i t_j} g(d_{ij}) + \beta \sum_{i \in \mathcal{S}_H} \delta_{t_i} s_i.$$

It unlikely that there exists a polynomial-time algorithm to produce optimal sequences with respect to *any* collection of parameters $\{\delta_i\}$ and $\{\varepsilon_{ij}\}$, for we can show how to encode the NP-complete MAXIMUM CUT problem by an appropriate choice of these parameters. However, we now show that it is possible to design optimal sequences efficiently with respect to a large class of parameter sets.

We say that a set of contact parameters $\{\varepsilon_{ij} : 0 \leq i, j \leq k\}$ is *layered* if there exist non-negative numbers $\{\varepsilon'_{ij} : 0 \leq i, j \leq k\}$ so that $\varepsilon_{ij} = \sum_{a \leq i, b \leq j} \varepsilon'_{ab}$. This notion is a useful one, in that many natural sets of contact parameters can be shown to be layered. For example, suppose we have an underlying set of numbers $0 = \psi_0 \leq \psi_1 \leq \cdots \leq \psi_k$, and we define $\varepsilon_{ij} = \psi_i \psi_j$, or $\varepsilon_{ij} = \min(\psi_i, \psi_j)$. Then the resulting set $\{\varepsilon_{ij}\}$ is layered. In particular, the H/P model is easily seen to be derived from a layered set of parameters.

We have developed a polynomial-time algorithm to design optimal sequences with respect to any model with layered contact parameters and arbitrary solvation parameters.

(5.2) *Suppose we are given a set of $k+1$ amino acid types, with $\{\delta_i\}$ a set of solvation parameters and $\{\varepsilon_{ij}\}$ a layered set of contact parameters. Then there is a polynomial-time* *algorithm that, given a target structure, produces a sequence in this model whose fitness is optimal.*

Proof. We define

$$B'' = \sum_{i<j-2} \sum_{\substack{a\le k\\ b\le k}} |\alpha|\varepsilon'_{ab} g(d_{ij}) = |\alpha| \sum_{i<j-2} \varepsilon_{kk} g(d_{ij}).$$

We define the following graph G based on Φ''. The vertex set of G contains

- vertices s and t;
- vertices $v_i^{\langle 1\rangle}, \ldots, v_1^{\langle k\rangle}$ for each residue position i;
- a vertex $u_{ij}^{\langle ab\rangle}$ for each pair of residue positions i, j satisfying $i < j-2$ and $g(d_{ij}) > 0$, and for each pair of numbers (a, b) satisfying $1 \le a, b \le k$ and $\varepsilon'_{ab} > 0$.

The edge set of G contains

- an edge $(s, u_{ij}^{\langle ab\rangle})$ of capacity $|\alpha|\varepsilon'_{ab} g(d_{ij})$ for each vertex $u_{ij}^{\langle ab\rangle}$;
- edges $(u_{ij}^{\langle ab\rangle}, v_i^{\langle a\rangle})$ and $(u_{ij}^{\langle ab\rangle}, v_j^{\langle b\rangle})$ of capacity $B''+1$ for each vertex $u_{ij}^{\langle ab\rangle}$;
- edges $(v_i^{\langle 1\rangle}, v_i^{\langle 2\rangle}), \ldots, (v_i^{\langle k-1\rangle}, v_i^{\langle k\rangle}), (v_i^{\langle k\rangle}, t)$ of capacities $\beta\delta_1 s_i, \beta\delta_2 s_i, \ldots, \beta\delta_k s_i$ for each residue position i.

As in the algorithm of Section 3, we define a notion of *closed sets* in our graph G. In the present context, we say that a set X is closed if

(i) X contains s but not t;

(ii) for each $u_{ij}^{\langle ab\rangle} \in V$, X contains $u_{ij}^{\langle ab\rangle}$ if and only if it contains both $v_i^{\langle a\rangle}$ and $v_j^{\langle b\rangle}$;

(iii) for each residue position i there is a number q_i so that $v_i^{\langle a\rangle} \in X$ if and only if $a \le q_i$. (If $v_i^{\langle 1\rangle} \notin X$, we can take $q_i = 0$.)

One can verify that for a given choice of the numbers $\{q_1, \ldots, q_n\}$, there is a unique closed set X; we will refer to the numbers $\{q_i\}$ as the *indices* of the corresponding closed set X.

By an argument similar to that in the proof of (3.1) we can show that if (X, Y) is a minimum s-t cut in G, then X is a closed set. We now observe that there is a natural one-to-one correspondence between closed sets in G and sequences of n residues over the alphabet $\{a_0, \ldots, a_k\}$. For given a closed set X, with indices $\{q_1, \ldots, q_n\}$, we define a protein sequence $\{t_1, \ldots, t_n\}$ in which $t_i = q_i$; conversely, given a protein sequence $\{t_1, \ldots, t_n\}$, we construct the closed set whose indices are the numbers $\{t_1, \ldots, t_n\}$. We let $\mathcal{S}''(X)$ denote the sequence associated with the closed set X.

Finally, we claim that for any closed set X, the capacity of the s-t cut $(X, V-X)$ is equal to $B'' + \Phi(\mathcal{S}''(X))$. Let $\{q_1, \ldots, q_n\}$ be the indices of X. For the sake of notation, we define $v_i^{\langle k+1\rangle}$ to be t, for all i. Now, note that the definition of *closed set* implies that the edges crossing (X,Y) consist of $(v_i^{\langle q_i\rangle}, v_i^{\langle q_i+1\rangle})$ for those i with $q_i > 0$, and $(s, u_{ij}^{\langle ab\rangle})$ where one of $v_i^{\langle a\rangle}$ or $v_i^{\langle b\rangle}$ does not belong to X. Thus

$$\begin{aligned}
c(X, V-X) &= \sum_{\substack{u_{ij}^{\langle ab\rangle}\in V\\ \{v_i^{\langle a\rangle}, v_j^{\langle b\rangle}\}\not\subseteq X}} |\alpha|\varepsilon'_{ab} g(d_{ij}) + \beta \sum_i \delta_{q_i} s_i. \\
&= B'' - \sum_{\substack{u_{ij}^{\langle ab\rangle}\in V\\ \{v_i^{\langle a\rangle}, v_j^{\langle b\rangle}\}\subseteq X}} |\alpha|\varepsilon'_{ab} g(d_{ij}) + \beta \sum_i \delta_{q_i} s_i. \\
&= B'' - \sum_{\substack{u_{ij}^{\langle ab\rangle}\in V\\ a\le q_i,\ b\le q_j}} |\alpha|\varepsilon'_{ab} g(d_{ij}) + \beta \sum_i \delta_{q_i} s_i. \\
&= B'' + \alpha \sum_{i<j-2} \varepsilon_{q_i q_j} g(d_{ij}) + \beta \sum_i \delta_{q_i} s_i. \\
&= B'' + \Phi''(\mathcal{S}''(X)).
\end{aligned}$$

Hence, to compute an optimal sequence, we construct the graph G and compute a minimum s-t cut (X,Y). We know that X will be a closed set, and that $\mathcal{S}''(X)$ achieves the minimum value of Φ''. Thus we return $\mathcal{S}''(X)$ as an optimal sequence. ∎

Acknowledgements. We thank Ron Elber for valuable discussions on this topic.

References

[1] R. Ahuja, T. Magnanti, J. Orlin, *Network Flows*, Prentice-Hall, 1993.

[2] A. Babajide, I. Hofacker, M. Sippl, P. Stadler, *Folding and Design* 2(1997), pp. 261–269.

[3] J. Banavar, M. Cieplak, A. Maritan, G. Nadig, F. Seno, S. Vishveshwara, *Proteins: Structure, Function, and Genetics* 31(1998), pp. 10–20.

[4] F.C. Bernstein, T.F. Koetzle, G.J.B. Williams, E.F. Meyer, M.D. Brice, J.R. Rodgers, O. Kennard, T. Shimanouchi, M.Tasumi, *J. Molecular Bio.* 112(1977), pp. 535–542.

[5] H.S. Chan, K.A. Dill, *Proteins: Structure, Function, and Genetics* 24(1996), pp. 335–344.

[6] B. Cherkassky, A.V. Goldberg, *Proc. MPS Symp. on Int. Prog. and Comb. Optimization* 1995, pp. 157–171. Implementation obtained from `http://www.neci.nj.nec.com/homepages/avg/`.

[7] T. Cormen, C. Leiserson, R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[8] T.E. Creighton, *Proteins: Structure and Molecular Properties*, Freeman, 1993.

[9] K.A. Dill, S. Bromberg, K. Yue, K. Fiebig, D. Yee, P. Thomas, H.S. Chan, *Protein Science* 4(1995), pp. 561–602.

[10] J.M. Deutsch, T. Kurosky, *Phys. Rev. Letters* 76(1996), pp. 323–326.

[11] K.E. Drexler, *Proc. Nat. Acad. Sci.* 78(1981), pp. 5275–5278.

[12] F. Eisenhaber, P. Argos, *J. Computational Chemistry* 14(1993), pp. 1272–1280.

[13] F. Eisenhaber, P. Lijnzaad, P. Argos, C. Sander, M. Scharf, *J. Computational Chemistry* 16(1995), pp. 273–284.

[14] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[15] A.V. Goldberg, R.E. Tarjan, *Journal of the ACM* 35(1988), pp. 921–940.

[16] M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer–Verlag, 1987.

[17] W. Hart, Proc. RECOMB Conf. on Computational Molecular Biology 1997, 128–136.

[18] L. Holm, C. Sander. *Science* 273 (1996), 595.

[19] L. Holm, C. Sander. *Proteins: Structure, Function, and Genetics* 19(1994), 165.

[20] M. Huynen, P. Stadler, W. Fontana, *Proc. Nat. Acad. Sci.* 93(1996), pp. 397–401.

[21] S. Kamtekar, J. Schiffer, H. Xiong, J. Babik, M. Hecht, *Science* 262(1993), 1680–1685.

[22] M. Kimura, *The Neutral Theory of Molecular Evolution*, Cambridge University Press, 1983.

[23] K.F. Lau, K. Dill, *Macromolecules* 22(1989), pp. 3986–3997.

[24] K.F. Lau, K. Dill, *Proc. Nat. Acad. Sci.* 87(1990), pp. 638–642.

[25] D. Lipman, W. Wilbur, *Proc. Royal Soc. London B* 245(1991), pp. 7–11.

[26] J. Maynard Smith, *Nature* 225(1970), pp. 563–564.

[27] K. Merz, S. LeGrand, Eds., *The Protein Folding Problem and Tertiary Structure Prediction*, Birkhäuser, 1994,

[28] C. Micheletti, F. Seno, A. Maritan, J. Banavar, *Proteins: Structure, Function, and Genetics* 32(1998), pp. 80–87.

[29] G. Nemhauser, L. Wolsey, *Integer and Combinatorial Optimization*, *Wiley*, 1988.

[30] J. Ponder, F.M. Richards, *J. Molecular Bio.* 193(1987), pp. 63–89.

[31] C. Reidys, P. Stadler, P. Schuster, *Bull. Math. Bio.* 59(1997), pp. 339-397.

[32] T.J. Richmond, F.M. Richards, *J. Molecular Bio.* 119(1978), pp. 537–555.

[33] S. Sun, R. Brem, H.S. Chan, K.A. Dill, *Protein Engineering* 8(1995), pp. 1205–1213.

[34] E.I. Shakhnovich, A.M. Gutin, *Protein Engineering* 6(1993), pp. 793–800.

[35] E.I. Shakhnovich, G. Farztdinov, A.M. Gutin, M. Karplus, *Phys. Rev. Letters* 67(1991), pp. 1665–1668.

[36] D. Sleator, R.E. Tarjan, *J. Computer and System Sciences* 26(1983), pp. 362–391.

[37] K. Yue, K.A. Dill, *Proc. Nat. Acad. Sci.* 89(1992), pp. 4163–4167.