A Combinatorial Toolbox for Protein Sequence Design and Landscape Analysis in the Grand Canonical Model*

James Aspnes[1**], Julia Hartling[2], Ming-Yang Kao[3***], Junhyong Kim[245†], and Gauri Shah[1]

[1] Department of Computer Science, Yale University, New Haven, CT 06520, USA.
[2] Department of Ecology and Evolutionary Biology, Yale University.
[3] Department of EECS, Tufts University, Medford, MA 02155, USA.
[4] Department of Molecular, Cellular, and Developmental Biology, Yale University.
[5] Department of Statistics, Yale University.

Abstract. In modern biology, one of the most important research problems is to understand how protein sequences fold into their native 3D structures. To investigate this problem at a high level, one wishes to analyze the *protein landscapes*, i.e., the structures of the space of all protein sequences and their native 3D structures. Perhaps the most basic computational problem at this level is to take a target 3D structure as input and *design* a fittest protein sequence with respect to one or more fitness functions of the target 3D structure. We develop a toolbox of combinatorial techniques for protein landscape analysis in the Grand Canonical model of Sun, Brem, Chan, and Dill. The toolbox is based on linear programming, network flow, and a linear-size representation of all minimum cuts of a network. It not only substantially expands the network flow technique for protein sequence design in Kleinberg's seminal work but also is applicable to a considerably broader collection of computational problems than those considered by Kleinberg. We have used this toolbox to obtain a number of efficient algorithms and hardness results. We have further used the algorithms to analyze 3D structures drawn from the Protein Data Bank and have discovered some novel relationships between such native 3D structures and the Grand Canonical model.

1 Introduction

In modern biology, one of the most important research problems is to understand how protein sequences fold into their native 3D structures. This problem can be investigated at two complementary levels. At a low level, one wishes to determine how an individual protein sequence folds. A fundamental computational problem

* aspnes@cs.yale.edu, julia.kreychman@yale.edu, kao@eecs.tufts.edu, junhyong.kim@yale.edu, gauri.shah@yale.edu.
** Supported in part by NSF Grant CCR-9820888.
*** NSF Grant CCR-9531028.
† Merck Genome Research Institute Grant and NSF Grant DEB-9806570.

at this level is to take a protein sequence as input and find its native 3D structure. This problem is sometimes referred to as the protein *structure prediction* problem and has been shown to be NP-hard (e.g., [1]). At a high level, one wishes to analyze the *protein landscapes*, i.e., the structures of the space of all protein sequences and their native 3D structures. Perhaps the most basic computational problem at this level is to take a target 3D structure as input and ask for a fittest protein sequence with respect to one or more fitness functions of the target 3D structure. This problem has been called the protein *sequence design* problem and has been investigated in a number of studies (e.g., [2]).

The focus of this paper is on protein landscape analysis, for which several quantitative models have been proposed in the literature (e.g., [9]). As some recent studies on this topic have done (e.g., [6]), this paper employs the Grand Canonical (GC) model of Sun, Brem, Chan, and Dill [9], whose definition is given in Section 2. Generally speaking, the model is specified by (1) a 3D geometric representation of a target protein 3D structure with n amino acid residues, (2) a *binary folding code* in which the amino acids are classified as *hydrophobic* (H) or *polar* (P), and (3) a fitness function Φ defined in terms of the target 3D structure that favors protein sequences with a dense hydrophobic core and with few solvent-exposed hydrophobic residues.

In this paper, we develop a toolbox of combinatorial techniques for protein landscape analysis based on linear programming, network flow, and a linear-size representation of all minimum cuts of a network [7]. This toolbox not only substantially expands the network flow technique for protein sequence design in Kleinberg's seminal paper [6] but also is applicable to a considerably broader collection of computational problems than those considered by Kleinberg. We have used this toolbox to obtain a number of efficient algorithms and hardness results. We have further used the algorithms to analyze 3D structures drawn from Protein Data Bank at `http://www.rcsb.org/pdb` and have discovered some novel relationships between such native 3D structures and the Grand Canonical model (Section 6). Specifically, we report new results on the following problems, where Δ is the number of terms in the fitness function or functions as further defined in Section 3. Many of the results depend on computing a maximum network flow in a graph of size $O(\Delta)$; in most cases, this network flow only needs to be computed once for each fitness function Φ.

- P1 Given a 3D structure, find all its fittest protein sequences. Note that there can be exponentially many fittest protein sequences. We show that these protein sequences together have a representation of size $O(\Delta)$ that can be computed in $O(\Delta)$ time after a certain maximum network flow is computed (Theorem 1), and that individual fittest protein sequences can be generated from this representation in $O(n)$ time per sequence (Theorem 5).
- P2 Given f 3D structures, find the set of all protein sequences that are the fittest simultaneously for all these 3D structures. This problem takes $O(\Delta)$ time after f maximum network flow computations (Theorem 4).
- P3 Given a protein sequence $\hat{x}$ and its native 3D structure, find the set of all fittest protein sequences that are also the most (or least) similar to $\hat{x}$ in

terms of unweighted (or weighted) Hamming distances. This problem takes $O(\Delta)$ time after a certain maximum network flow is computed (Theorem 3).

P4 Count the number of protein sequences in the solution to each of Problems P1, P2, and P3. These counting problems are computationally hard (Theorem 11).

P5 Given a 3D structure and a bound e, enumerate the protein sequences whose fitness function values are within an additive factor e of that of the fittest protein sequences. This problem takes polynomial time to generate each desired protein sequence (Theorem 8).

P6 Given a 3D structure, find the largest possible unweighted (or weighted) Hamming distance between any two fittest protein sequences. This problem takes $O(\Delta)$ time after a certain maximum network flow is computed (Theorem 6).

P7 Given a protein sequence $\hat{x}$ and its native 3D structure, find the average unweighted (or weighted) Hamming distance between $\hat{x}$ and the fittest protein sequences for the 3D structure. This problem is computationally hard (Theorem 11).

P8 Given a protein sequence $\hat{x}$, its native 3D structure, and two unweighted Hamming distances d_1 and d_2, find a fittest protein sequence whose distance from $\hat{x}$ is also between d_1 and d_2. This problem is computationally hard (Theorem 12(1)).

P9 Given a protein sequence $\hat{x}$, its native 3D structure, and an unweighted Hamming distance d, find the fittest among the protein sequences which are at distance d from $\hat{x}$. This problem is computationally hard (Theorem 12(2)). We have a polynomial-time approximation algorithm for this problem (Theorem 9).

P10 Given a protein sequence $\hat{x}$ and its native 3D structure, find all the ratios between the scaling factors α and β in Equation 1 in Section 2 for the GC model such that the smallest possible unweighted (or weighted) Hamming distance between $\hat{x}$ and any fittest protein sequence is minimized over all possible α and β. (This is a problem of tuning the GC model.) We have a polynomial-time algorithm for this problem (Theorem 10).

P11 Given a 3D structure, determine whether the fittest protein sequences are *connected*, i.e., whether they can mutate into each other through allowable mutations, such as point mutations, while the intermediate protein sequences all remain the fittest (e.g., [8]). This problem takes $O(\Delta)$ time after a certain maximum network flow is computed (Theorem 7).

P12 Given a 3D structure, in the case that the set of all fittest protein sequences is not connected, determine whether two given fittest protein sequences are connected. This problem takes $O(\Delta)$ time after a certain maximum network flow is computed (Theorem 7).

P13 Given a 3D structure, find the smallest set of allowable mutations with respect to which the fittest protein sequences (or two given fittest protein sequences) are connected. This problem takes $O(\Delta)$ time after a certain maximum network flow is computed (Theorem 7).

Previously, Sun *et al.* [9] developed a heuristic algorithm to search the space of protein sequences for a fittest protein sequence without a guarantee of optimality or near-optimality. Hart [5] subsequently raised the computational tractability of constructing a single fittest protein sequence as an open question. Kleinberg [6] gave the first polynomial-time algorithm for this problem, which is based on network flow. In contrast, Problem P1 asks for all fittest protein sequences and yet can be solved with the same time complexity. Kleinberg also formulated more general versions of Problems P11 and P12 by extending the fitness function to a submodular function and gave polynomial-time algorithms. Our formulations of these two problems and Problem P13 are directly based on the fitness function of the GC model; furthermore, as is true with several other problems above, once a solution to Problem P1 is obtained, we can solve these three problems in $O(\Delta)$ time. Among the above thirteen problems, those not yet mentioned in this comparison were not considered by Kleinberg.

The remainder of this paper is organized as follows. Section 2 defines the GC model and states the basic computational assumptions. Section 3 describes our three basic tools based on linear programming, network flow, and an $O(\Delta)$-size representation of minimum cuts. Section 4 extends these tools to optimize multiple objectives, analyze the structures of the space of all fittest protein sequences, and generate near-fittest protein sequences. Section 5 gives some hardness results related to counting fittest protein sequences and finding fittest protein sequences under additional restrictions. Finally, Section 6 discusses our analysis of empirical 3D structures from the Protein Data Bank.

2 The Grand Canonical Model

The Original Model Throughout this paper, all protein sequences are of n residues, unless explicitly stated otherwise. The GC model is specified by a fitness function Φ over all possible protein sequences x with respect to a given 3D structure of n residues [9]. In the model, to design a protein sequence x is to specify which residues are hydrophobic (H) and which ones are polar (P). Thus, we model x as a binary sequence $x_1, \ldots, x_n$ or equivalently as a binary vector $(x_1, \ldots, x_n)$, where the i-th residue in x is H (respectively, P) if and only if $x_i = 1$ (respectively, 0). Then, $\Phi(x)$ is defined as follows, where the smaller $\Phi(x)$ is, the fitter x is, as the definition is motivated by the requirements that H residues in x (1) should have low solvent-accessible surface area and (2) should be close to one another in space to form a compact hydrophobic core.

$$\Phi(x) = \alpha \sum_{i,j \in H(x), i<j-2} g(d_{i,j}) + \beta \sum_{i \in H(x)} s_i \tag{1}$$

$$= \alpha \sum_{i<j-2} g(d_{i,j}) x_i x_j + \beta \sum_i s_i x_i, \text{ where} \tag{2}$$

- $H(x) = \{i \mid x_i = 1\}$,

- the scaling parameters $\alpha < 0$ and $\beta > 0$ have default values -2 and $\frac{1}{3}$ respectively and may require tuning for specific applications (see Section 4),
- $s_i \geq 0$ is the area of the solvent-accessible contact surface for the residue (in Å) [4],
- $d_{i,j} > 0$ is the distance between the residues i and j (in Å), and
- g is a sigmoidal function, defined by

$$g = \begin{cases} \frac{1}{1+\exp(d_{i,j}-6.5)} & \text{when } d_{i,j} \leq 6.5 \\ 0 & \text{when } d_{i,j} > 6.5. \end{cases}$$

Extending the Model with Computational Assumptions Let $\text{opt}(\Phi)$ be the set of all protein sequences x that minimize Φ. This paper is generally concerned with the structure of $\text{opt}(\Phi)$. Our computational problems assume that Φ is given as input; in other words, the computations of $\alpha, \beta, s_i, g(d_{i,j})$ are not included in the problems. Also, for the sake of computational generality and notational simplicity, we assume that α may be any nonpositive number, β any nonnegative number, s_i any arbitrary number, and $g(d_{i,j})$ any arbitrary nonnegative number; and that the terms $g(d_{i,j})$ may range over $1 \leq i < j \leq n$, unless explicitly stated otherwise. Thus, in the full generality of these assumptions, Φ need not correspond to an actual protein 3D structure. Note that the relaxation that s_i is any number is technically useful for finding Φ-minimizing protein sequences x that satisfy additional constraints.

We write $a_{i,j} = -\alpha \cdot g(d_{i,j}) \geq 0$ and $b_i = \beta \cdot s_i$ and further assume that the coefficients $a_{i,j}$ and b_i are rational with some common denominator, that these coefficients are expressed with a polynomial number of bits, and that arithmetic operations on these coefficients take constant time.

With these assumptions, we define the following sets of specific assumptions about Φ to be used at different places of this paper.

F1 Let $\Phi(x) = -\sum_{1 \leq i < j \leq n} a_{i,j} x_i x_j + \sum_{1 \leq i \leq n} b_i x_i$, where $a_{i,j} \geq 0$, b_i is arbitrary, and m of the coefficients $a_{i,j}$ are nonzero. Let $\Delta = n + m$.

F2 For each $\beta \geq 0$, let $\Phi_\beta(x) = -\sum_{1 \leq i < j \leq n} a_{i,j} x_i x_j + \beta \sum_{1 \leq i \leq n} s_i x_i$, where $a_{i,j} \geq 0$, $s_i \geq 0$, and m of the coefficients $a_{i,j}$ are nonzero. Let $\Delta = n + m$.

F3 For each ℓ from 1 to f, define the ℓ-th fitness function $\Phi^\ell(x) = -\sum_{1 \leq i < j \leq n} a^\ell_{i,j} x_i x_j + \sum_{1 \leq i \leq n} b^\ell_i x_i$, where $a^\ell_{i,j} \geq 0$ and b^ℓ_i is arbitrary. Let $\Delta = fn^2$.

Sometimes we measure the dissimilarity between a fittest protein sequence x and a target protein sequence $\hat{x}$ in terms of Hamming distance. This distance is essentially the count of the positions i where $x_i \neq \hat{x}_i$ and can be measured in two ways. The *unweighted* Hamming distance is $|x - \hat{x}|$, where $|y|$ denotes the *norm* of vector y, i.e., $\sum_{i=1}^n |y_i|$. The *weighted* Hamming distance is $\sum_{i=1}^n w_i \cdot |x_i - \hat{x}_i|$. Throughout this paper, the weights $w_1, \ldots, w_n$ are all arbitrary unless explicitly stated otherwise.

3 Three Basic Tools

This section describes our basic tools for computing fittest and near-fittest protein sequences. For instance, Lemma 1 gives a representation of the problem of minimizing Φ as a linear program. Lemma 2 further gives a representation of this problem as a minimum-cut problem, which generalizes a similar representation of Kleinberg [6]. Theorem 1 gives a compact representation of the space $\mathrm{opt}(\Phi)$ using a Picard-Queyranne graph [7].

Linear Programming From Equation 2, minimizing $\Phi(x)$ is an optimization problem in quadratic programming. Fortunately, because all the coefficients $a_{i,j}$ are nonnegative, it can be converted to a linear program, as shown in Lemma 1.

Lemma 1 (characterizing Φ via linear program). *Let Φ be as defined in Assumption F1. Consider the following linear program whose variables consist of the variables x_i, together with new variables $y_{i,j}$ for all i, j with $a_{i,j} \neq 0$:*

$$\begin{array}{l} \textit{minimize } \Phi'(x,y) = -\sum a_{i,j} y_{i,j} + \sum b_i x_i \\ \textit{subject to} \\ \quad 0 \leq x_i \leq 1 \qquad \forall i \\ \left.\begin{array}{l} 0 \leq y_{i,j} \leq 1 \\ y_{i,j} \leq x_i \\ y_{i,j} \leq x_j \end{array}\right\} \forall i,j : a_{i,j} \neq 0 \end{array} \tag{3}$$

There is a one-to-one correspondence that preserves x between the protein sequences that minimize $\Phi(x)$ and the basic optimal solutions to Linear Program (3).

Note that any x_i with a negative coefficient b_i is set to 1 in any optimal solution, as in this case all terms containing x_i have negative coefficients and are minimized when $x_i = 1$. So an alternative to allowing negative coefficients is to prune out any x_i with a negative coefficient. This process must be repeated recursively, since setting x_i to 1 reduces terms of the form $-a_{i,j}x_i x_j$ to $-a_{i,j}x_j$, and may yield more degree-1 terms with negative coefficients. To simplify our discussion, we let the linear program (or, in Section 3, the minimum-cut algorithm) handle this pruning.

Network Flow Recall that an *s-t cut* is a partition of the nodes of a digraph into two sets V_s and V_t, with $s \in V_s$ and $t \in V_t$. Also, a *minimum s-t* cut is an s-t cut with the smallest possible total capacity of all edges from nodes in V_s to nodes in V_t.

In Kleinberg's original construction [6], $\Phi(x)$ was minimized by solving an s-t minimum cut problem in an appropriate digraph G. Lemma 2 describes a more general construction that includes additional edges (s, v_i) to handle negative values for b_i.

Lemma 2 (characterizing Φ via network flow). *Let Φ be as defined in Assumption F1. Let G^Φ be a graph with a source node s, a sink node t, a node v_i for each i, and a node $u_{i,j}$ for each i, j with $a_{i,j} \neq 0$, for a total of $n + m + 2 = \Delta + 2$ nodes. Let the edge set of G^Φ consist of*

– $(s, u_{i,j})$ *for each* $u_{i,j}$*, with capacity* $a_{i,j}$*,*
– (v_i, t) *for each* v_i *with* $b_i > 0$*, with capacity* b_i*,*
– (s, v_i) *for each* v_i *with* $b_i < 0$*, with capacity* $-b_i$*, and*
– $(u_{i,j}, v_i)$ *and* $(u_{i,j}, v_j)$*, for each* $u_{i,j}$*, with infinite capacity,*

for a total of $\Theta(\Delta)$ *edges.*

There is a one-to-one correspondence between the minimum s-t cuts in G^{Φ} *and the protein sequences in* $\mathrm{opt}(\Phi)$*, such that* v_i *is in the s-component of a cut if and only if* $x_i = 1$ *in the corresponding protein sequence.*

Lemma 3. *Let* Φ *be as defined in Assumption F1. Given* Φ *as the input, we can find an* $x \in \mathrm{opt}(\Phi)$ *in* $O(\Delta^2 \log \Delta)$ *time.*

A Compact Representation of Minimum Cuts A given Φ may have more than one fittest protein sequence. Theorem 1 shows that $\mathrm{opt}(\Phi)$ can be summarized compactly using the Picard-Queyranne representation of the set of all minimum s-t cuts in a digraph G [7], which is computed by the following steps:

1. computing any maximum flow ϕ in G;
2. computing strongly connected components in the residual graph G_ϕ whose edge set consists of all edges in G that are not saturated by ϕ, plus edges (v, u) for any edge (u, v) that has nonzero flow in ϕ;
3. contracting G_ϕ by contracting into single supernodes the set of all nodes reachable from s, the set of all nodes that can reach t, and each strongly connected component in the remaining graph.

The resulting graph $G_{s,t}$ is a dag in which s and t are mapped to distinct supernodes by the contraction. Furthermore, there is a one-to-one correspondence between the minimum s-t cuts in G and the ideals in $G_{s,t}$, where an *ideal* is any node set I with the property that any predecessor of a node in I is also in I.

Lemma 4 (see [7]). *Given a digraph* G *with designated nodes* s *and* t*, there is a graph* $G_{s,t}$ *together with a mapping* κ *from* $V(G)$ *to* $V(G_{s,t})$ *with the following properties:*

1. $|V(G_{s,t})| \leq |V(G)|$.
2. *The node* $\kappa(s)$ *has out-degree* 0 *while* $\kappa(t)$ *has in-degree* 0.
3. *Given* G *as the input,* $G_{s,t}$ *and* κ *can be computed using one maximum-flow computation and* $O(|E(G)|)$ *additional work.*
4. *A partition* (V_s, V_t) *of* $V(G)$ *is an* s-t *minimum cut in* G *if and only if* $V_t = \kappa^{-1}(I)$ *for some ideal* I *of* $G_{s,t}$ *that contains* $\kappa(t)$ *but not* $\kappa(s)$*.*

Combining Lemmas 2 and 4 gives the desired compact representation of the space of all fittest protein sequences, as stated in the next theorem.

Theorem 1 (characterizing Φ via a dag). *Let* Φ *be as defined in Assumption F1. There exists a dag* $G^{\Phi}_{s,t}$ *with designated nodes* s' *and* t' *and a mapping* ρ *from* $\{1, \ldots, n\}$ *to* $V(G^{\Phi}_{s,t})$ *with the following properties:*

1. *$G^{\Phi}_{s,t}$ has at most $n+2$ nodes.*
2. *Given Φ as the input, $G^{\Phi}_{s,t}$ and ρ can be computed in $O(\Delta^2 \log \Delta)$ time.*
3. *There is a one-to-one correspondence between the protein sequences $x \in \text{opt}(\Phi)$ and the ideals of $\widehat{G}^{\Phi}_{s,t} = G^{\Phi}_{s,t} - \{s', t'\}$, in which $x_i = 0$ if and only if $\rho(i) = t'$ or $\rho(i)$ is in the ideal corresponding to x.*

Intuitively, what Theorem 1 says is the following. For any Φ, the residues in fittest protein sequences are grouped into clusters, where the cluster $\rho^{-1}(s)$ is always H, the cluster $\rho^{-1}(t)$ is always P, and for each of the remaining clusters, all residues in the cluster are either all H or all P. In addition, there is a dependence given by the edges of $\widehat{G}^{\Phi}_{s,t}$, such that if a cluster corresponding to the source of an edge is all H then the cluster at the other end is also all H.

There is no additional restriction on the structure of the space of all fittest protein sequences beyond those that follow from correspondence with the ideals of some digraph. As shown in Theorem 2, any graph may appear as $\widehat{G}^{\Phi}_{s,t}$, with any number of residues mapped to each supernode.

Theorem 2 (characterizing a dag via Φ). *Let $\hat{G}$ be an arbitrary digraph with n nodes, labeled 1 to n, and m edges. Let $\hat{G}_0$ be the* component graph *of $\hat{G}$ obtained by contracting each strongly connected component of $\hat{G}$ to a single supernode through a contraction map κ. Then, there exists some Φ as defined in Assumption F1 such that for the $G^{\Phi}_{s,t}$ and ρ defined in Theorem 1, an isomorphism exists between $\widehat{G}^{\Phi}_{s,t}$ and $\hat{G}_0$ mapping each $\rho(i)$ to $\kappa(i)$.*

4 Further Tools for Protein Landscape Analysis

Optimizing Multiple Objectives We can extend the results of Section 3 beyond optimizing a single fitness function.

With more than one fittest protein sequence to choose from, we may wish to find a fittest protein sequence x that is the closet to some target protein sequence $\hat{x}$ in unweighted or weighted Hamming distance. Theorem 3 shows that this optimization problem is as easy as finding an arbitrary fittest protein sequence.

We may also wish to consider what protein sequences are simultaneously the fittest for more than one fitness function. Theorem 4 shows how to compute a representation of this set similar to that provided by Theorem 1.

Theorem 3 (optimizing Hamming distances and H-residue counts over opt(Φ)). *Let Φ be as defined in Assumption F1.*

1. *Given a target protein sequence $\hat{x}$, some weights w_i, and Φ as the input, we can find in $O(\Delta^2 \log \Delta)$ time an $x \in \text{opt}(\Phi)$ with the minimum weighted Hamming distance $\sum_i w_i |x_i - \hat{x}_i|$ over $\text{opt}(\Phi)$.*
2. *Given Φ as the input, we can find in $O(\Delta^2 \log \Delta)$ time an $x \in \text{opt}(\Phi)$ with the largest (or smallest) possible number of H residues over $\text{opt}(\Phi)$.*

Theorem 4 (minimizing multiple fitness functions). *Let $\Phi^1, \ldots, \Phi^f$ be as defined in Assumption F3. For each ℓ, let $G_{s,t}^{\Phi^\ell}$ and ρ^ℓ be the dag and map computed from Φ^ℓ in Theorem 1. Given all $G_{s,t}^{\Phi^\ell}$ and ρ^ℓ as the input, there is an $O(\Delta)$-time algorithm that either (a) determines that there is no protein sequence x that simultaneously minimizes Φ^1 through Φ^f, or (b) constructs a dag $G_{s,t}^{\Phi^*}$ with designated nodes s' and t' and a mapping ρ^* from $\{1, \ldots, n\}$ to $V(G_{s,t}^{\Phi^*})$, such that there is a one-to-one correspondence between the protein sequences x that simultaneously minimize all $\Phi^\ell(x)$ and the ideals of $\widehat{G}_{s,t}^{\Phi^*} = G_{s,t}^{\Phi^*} - \{s', t'\}$, in which $x_i = 0$ if and only if $\rho^*(i) = t'$ or $\rho^*(i)$ is in the ideal corresponding to x.*

The Space of All Fittest Protein Sequences Here we discuss some applications of the representation of the space $\mathrm{opt}(\Phi)$ given by Theorem 1. Theorem 5 gives an algorithm to enumerate this space. Theorem 6 gives an algorithm to compute the diameter of the space in nonnegatively weighted Hamming distance. Theorem 7 gives an algorithm to determine connectivity properties of the space with respect to various classes of mutations.

Theorem 5 (enumerating all protein sequences). *Let Φ be as defined in Assumption F1. Given the $G_{s,t}^{\Phi}$ and ρ defined in Theorem 1 as the input, the protein sequences in* $\mathrm{opt}(\Phi)$ *can be enumerated in $O(n)$ time per protein sequence.*

Theorem 6 (computing the diameter). *Let Φ be as defined in Assumption F1. Given the $G_{s,t}^{\Phi}$ and ρ defined in Theorem 1 as the input, it takes $O(n)$ time to compute the diameter of* $\mathrm{opt}(\Phi)$ *in weighted Hamming distance where the weights w_i are all nonnegative.*

We can use $\widehat{G}_{s,t}^{\Phi}$ to determine whether $\mathrm{opt}(\Phi)$ is connected for various models of mutations. For instance, we can determine whether the space is connected for one-point mutations, in which at most one residue changes with each mutation and all intermediate protein sequences must remain the fittest. More generally, we can determine the minimum k so that the space is connected where each mutation modifies at most k residues.

We adopt a general model proposed by Kleinberg [6]. In the model, there is a system Λ of subsets of $\{1, \ldots, n\}$ that is *closed downward*, i.e., if $A \subseteq B \in \Lambda$, then $A \in \Lambda$. Two protein sequences x and y are *Λ-adjacent* if they are in $\mathrm{opt}(\Phi)$ and differ exactly at the positions indexed by elements of some member of Λ. A *Λ-chain* is a sequence of protein sequences in $\mathrm{opt}(\Phi)$ where each adjacent pair is Λ-adjacent. Two protein sequences x and y are *Λ-connected* if there exists a Λ-chain between x and y. A set of protein sequences is *Λ-connected* if every pair of elements of the set are *Λ-connected*. We would like to tell for any given Λ and Φ whether particular protein sequences are Λ-connected and whether the entire $\mathrm{opt}(\Phi)$ is Λ-connected.

Kleinberg [6] gives polynomial-time algorithms for these problems that take Λ as input (via oracle calls) and depend only on the fact that Φ is submodular. We describe a much simpler algorithm that uses $\widehat{G}_{s,t}^{\Phi}$ from Theorem 1. This algorithm not only determines whether two protein sequences (alternatively, all

protein sequences in $\mathrm{opt}(\Phi)$) are connected for any given Λ, but also determines the unique minimum Λ for which the desired connectivity holds. Almost all of the work is done in the computation of $\widehat{G}^{\Phi}_{s,t}$; once we have this representation, we can read off the connectivity of $\mathrm{opt}(\Phi)$ directly.

Theorem 7 (connectivity via mutations). *Let Φ be as defined in Assumption F1. The following problems can both be solved in $O(n)$ time.*

1. *Given the $G^{\Phi}_{s,t}$ and ρ defined in Theorem 1 and two protein sequences x and x' in $\mathrm{opt}(\Phi)$ as the input, compute the maximal elements of the smallest downward-closed set system Λ such that x and x' are Λ-connected.*
2. *Given the $G^{\Phi}_{s,t}$ and ρ defined in Theorem 1 as the input, compute the maximal elements of the smallest downward-closed set system Λ such that $\mathrm{opt}(\Phi)$ is Λ-connected.*

Generating Near-Fittest Protein Sequences Finding good protein sequences other than the fittest is trickier, as Lemma 1 breaks down if we are not looking at the fittest protein sequences. Here we give two algorithms that avoid this problem. Theorem 8 describes an algorithm to generate all protein sequences x in order of increasing $\Phi(x)$. Theorem 9 describes an algorithm to generate the fittest protein sequences at different unweighted Hamming distances, which is useful for examining the trade-off between fitness and distance.

Theorem 8 (enumerating all protein sequences). *Let Φ be as defined in Assumption F1. With Φ as the input, we can enumerate all protein sequences x in order of increasing $\Phi(x)$ in time $O(n\Delta^2 \log \Delta)$ per protein sequence.*

Let $\hat{x}$ be a target protein sequence. For $d \in \{0, \ldots, n\}$, let $F(d)$ be the smallest $\Phi(x)$ over all protein sequences x at unweighted Hamming distance d from $\hat{x}$. A basic task of landscape analysis is to plot the graph of F. As Theorem 12(2) in Section 5 shows, this task is computationally difficult in general. Therefore, one way to plot the graph of F would be to use Theorem 8 to enumerate all protein sequences x in order of increasing $\Phi(x)$ until for each d, at least one protein sequence at distance d from $\hat{x}$ has been enumerated. This solution may require processing exponentially many protein sequences before F is fully plotted. As an alternative, Theorem 9 gives a tool for plotting F approximately in polynomial time.

Theorem 9 (approximately plotting the energy-distance landscape). *Let Φ be as defined in Assumption F1. For each ϵ, let $\Phi_\epsilon(x) = \Phi(x) + \epsilon \cdot |x - \hat{x}|$. Let $\overline{\Phi}(\epsilon)$ be the minimum $\Phi_\epsilon(x)$ over all x.*

1. *$\overline{\Phi}$ is a continuous piecewise linear concave function defined on $\mathbf{R}$ with at most $n+1$ segments and thus at most $n+1$ corners.*
2. *Let $(\epsilon_1, \overline{\Phi}(\epsilon_1)), \ldots, (\epsilon_k, \overline{\Phi}(\epsilon_k))$ be the corners of $\overline{\Phi}$, where $\epsilon_1 < \cdots < \epsilon_k$. Let d_i be the slope of the segment immediately to the right of ϵ_i. Let d_0 be the slope of the segment immediately to the left of ϵ_1. Then, $n = d_0 > d_1 > \cdots > d_k = 0$.*
3. *Given Φ and $\hat{x}$ as the input, we can compute $(\epsilon_1, \overline{\Phi}(\epsilon_1)), \ldots, (\epsilon_k, \overline{\Phi}(\epsilon_k))$ and $d_0, \ldots, d_k$ in $O(n\Delta^2 \log \Delta)$ time.*

Tuning the Parameters of the GC Model Here we show how to systematically tune the parameters α and β so that a fittest protein sequence for a given 3D structure matches the 3D structure's native protein sequence as closely as possible in terms of unweighted or weighted Hamming distance. For this purpose, we assume $s_i \geq 0$. Furthermore, since the fitness function does not have an absolute scale, we may fix α at -1 and vary β.

Theorem 10 (tuning α and β). *Let Φ be as defined in Assumption F2. Given a target protein sequence $\hat{x}$ and Φ as the input, we can find in $O(n\Delta^2 \log \Delta)$ time the set of all β where the closest unweighted (or weighted) Hamming distance between $\hat{x}$ and any protein sequence in $\mathrm{opt}(\Phi_\beta)$ is the minimum over all β.*

5 Computational Hardness Results

Theorem 11 (hardness of counting and averaging). *Let Φ be as defined in Assumption F1. The following problems are all #P-complete:*

1. *Given Φ as the input, compute the cardinality of $\mathrm{opt}(\Phi)$.*
2. *Given $\Phi^1, \ldots, \Phi^f$ as the input, where f is any fixed positive integer and $\Phi^1, \ldots, \Phi^f$ are as defined in Assumption F3, compute the number of protein sequences x that simultaneously minimize $\Phi^\ell(x)$ for all $\ell = 1, \ldots, f$.*
3. *Given Φ as the input, compute the average norm $|x|$, i.e., the average number of H residues in x, over all $x \in \mathrm{opt}(\Phi)$.*
4. *Given Φ and a target protein sequence $\hat{x}$ as the input, compute the average unweighted Hamming distance $|x - \hat{x}|$ over all $x \in \mathrm{opt}(\Phi)$.*
5. *Given Φ, a target protein sequence $\hat{x}$, and an integer d as the input, compute the number of protein sequences in $\mathrm{opt}(\Phi)$ at unweighted Hamming distance d from $\hat{x}$.*

Theorem 12 (hardness of plotting the energy-distance landscape). *Let Φ be as defined in Assumption F1.*

1. *Given Φ and two integers d_1, d_2 as the input, it is NP-complete to determine whether there is an Φ-minimizing x with $d_1 \leq |x| \leq d_2$.*
2. *Let $\hat{x}$ be a target protein sequence. For $d \in \{0, \ldots, n\}$, let $F(d)$ be the smallest $\Phi(x)$ over all protein sequences x at unweighted Hamming distance d from $\hat{x}$. Given Φ and d as the input, it is NP-hard to compute $F(d)$.*

6 Applications to Empirical Protein 3D Structures

To demonstrate our algorithms, we chose 34 proteins with known 3D structures from the Protein Data Bank (PDB) at `http://www.rcsb.org/pdb`. These 3D structures included 8 from Kleinberg's study [6] but excluded the protein fragments and multimeric proteins used in that study. The chosen 3D structures were then represented by centroids for each side chain calculated from the coordinates of each atom in the side chain; in the case of 3D structures solved by NMR, hydrogen atoms were included into centroid calculations. For glycine,

the centroid was taken to be the position of C_α. For each side chain, the area of solvent accessible surface was computed via the Web interface of the ASC program with default parameters [4]. In accordance to the GC model, each of the chosen native protein sequences was converted into a binary H/P sequence following Sun *et al.* [9], where A, C, F, I, L, M, V, W, Y are H, and the other amino acids are P.

The detailed results of this small-scale empirical study can be found in the full version of this paper, which is deposited at Computing Research Repository as `http://xxx.lanl.gov/abs/cs.CE/0101015`.

As anticipated, our algorithms computed fittest protein sequences that are closer to native protein sequences than found by Kleinberg [6]. We further conjectured a significant relationship between a computed fittest protein sequence's similarity to a native protein sequence and the diversity of the native protein in nature. Such a relationship would be highly intriguing biologically. We examined this conjecture by assessing the diversity of native proteins using the database PFAM at `http://pfam.wustl.edu`, which is a database of protein families determined through Hidden Markov Models [3]. Our study confirmed this conjecture.

We are currently planning a large-scale analysis of further empirical protein 3D structures; the results will be reported in a subsequent paper.

References

1. J. Atkins and W. E. Hart. On the intractability of protein folding with a finite alphabet of amino acids. *Algorithmica*, 25(2-3):279–294, 1999.
2. J. Banavar, M. Cieplak, A. Maritan, G. Nadig, F. Seno, and S. Vishveshwara. Structure-based design of model proteins. *Proteins: Structure, Function, and Genetics*, 31:10–20, 1998.
3. A. Bateman, E. Birney, R. Durbin, S. R. Eddy, K. L. Howe, and E. L. L. Sonnhammer. PFAM– A database of protein domain family alignments and HMMs. *Nucleic Acids Research*, 28:263–266, 2000.
4. F. Eisenhaber, P. Lijnzaad, P. Argos, C. Sander, and M. Scharf. The double cube lattice method: Efficient approaches to numerical integration of surface area and volume and to dot surface contouring of molecular assemblies. *Journal of Computational Chemistry*, 16(N3):273–284, 1995.
5. W. E. Hart. On the computational complexity of sequence design problems. In *RECOMB*, pages 128–136, 1997.
6. J. M. Kleinberg. Efficient algorithms for protein sequence design and the analysis of certain evolutionary fitness landscapes. In *RECOMB*, pages 226–237, 1999.
7. J.-C. Picard and M. Queyranne. On the structure of all minimum cuts in a network and applications. *Mathematical Programming Study*, (13):8–16, 1980.
8. C. Reidys, P. Stadler, and P. Schuster. Generic properties of combinatory maps: Neutral networks of RNA secondary structures. *Bulletin of Mathematical Biology*, 59:339–397, 1997.
9. S. J. Sun, R. Brem, H. S. Chan, and K. A. Dill. Designing amino acid sequences to fold with good hydrophobic cores. *Protein Engineering*, 8(12):1205–1213, Dec. 1995.