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

James Aspnes* Julia Hartling† Ming-Yang Kao‡ Junhyong Kim§ Gauri Shah¶

January 19, 2001

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.

*Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: aspnes@cs.yale.edu. Supported in part by NSF Grant CCR-9820888.

†Department of Ecology and Evolutionary Biology, Yale University, New Haven, CT 06520-8285, USA. Email: julia.kreychman@yale.edu.

‡Department of Electrical Engineering and Computer Science, Tufts University, Medford, MA 02155, USA. Email: kao@eecs.tufts.edu. Supported in part by NSF Grant CCR-9531028.

§Department of Ecology and Evolutionary Biology, Department of Molecular, Cellular, and Developmental Biology, and Department of Statistics, Yale University, New Haven, CT 06520-8285, USA. Email: junhyong.kim@yale.edu. Supported in part by Merck Genome Research Institute Grant and NSF Grant DEB-9806570.

¶Department of Computer Science, Yale University, New Haven, CT 06520-8285, USA. Email: gauri.shah@yale.edu.

1 Introduction

In modern biology, one of the most important research problems is to understand how protein sequences fold into their native 3D structures [23]. 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 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 (see, e.g., [1, 5, 6]). 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 [3, 7, 9, 16, 27, 30, 33, 35].

The focus of this paper is on protein landscape analysis, for which several quantitative models have been proposed in the literature [7, 30, 33]. As some recent studies on this topic have done [3, 18, 24], this paper employs the Grand Canonical (GC) model of Sun, Brem, Chan, and Dill [33], 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) [8, 19], 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 [26]. This toolbox not only substantially expands the network flow technique for protein sequence design in Kleinberg's seminal paper [18] 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 (Figure 1). 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 3.1. 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 5), and that individual fittest protein sequences can be generated from this representation in $O(n)$ time per sequence (Theorem 9).

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 8).

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 7).

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 17).

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 12).

P6 Given a 3D structure, determine 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 10).

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 17).

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 18(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 18(2)). We have a polynomial-time approximation algorithm for this problem (Theorem 13).

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 16).

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 [2, 8, 17, 20, 22, 29, 31]. This problem takes $O(\Delta)$ time after a certain maximum network flow is computed (Theorem 11).

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 11).

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 11).

Previously, Sun *et. al.* [33] 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 [16] subsequently raised the computational tractability of constructing a single fittest protein sequence as an open question. Kleinberg [18] 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 and Computational Assumptions

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 [18, 33]. 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.

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

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

- $H(x) = \{i \mid x_i = 1\}$,
- 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 4.4),
- $s_i \geq 0$ is the area of the solvent-accessible contact surface for the residue (in Å) [10, 11],
- $d_{i,j} > 0$ 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, let 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 [18]. Theorem 5 gives a compact representation of the space $\mathrm{opt}(\Phi)$ using a Picard-Queyranne graph [26].

3.1 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 $a_{i,j} \neq 0$:*

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

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).

Proof: First, we show that for each 0-1 assignment to x there is a unique value of y that minimizes $\Phi'(x, y)$. Choose some $y_{i,j}$, and suppose that either x_i or x_j is 0. Then $y_{i,j}$ is also 0 by the constraint $y_{i,j} \leq x_i$ or $y_{i,j} \leq x_j$. Alternatively, suppose x_i and x_j are both 1; then if $y_{i,j}$ is 0, Φ' can be decreased by $a_{i,j}$ by setting $y_{i,j}$ to 1 without violating any constraints. Thus, in any optimal integral solution to Linear Program (3), $y_{i,j} = \min(x_i, x_j) = x_i x_j$.

Note that substituting $x_i x_j$ for $y_{i,j}$ in Φ' gives precisely $-\sum_{i,j} a_{i,j} x_i x_j + \sum_i b_i x_i = \Phi(x)$; thus minimizing $\Phi'(x, y)$ is equivalent to minimizing $\Phi(x)$.

We now must show that all solutions to Linear Program (3) are integral. Every element of the constraint matrix is either zero or ± 1. Each row has either a single nonzero element (e.g, for the 0-1 bounds) or consists of zeroes and exactly one $+1$ and one -1. Thus the matrix is totally unimodular, e.g., using [25, Theorem 13.3]. Since the right-hand side is integral, any vertex of the polytope defined by Linear Program (3) is integral [25, Theorem 13.2]. Thus, all basic feasible solutions to Linear Program (3) are 0-1 vectors.

So if (x, y) is a basic optimal solution to Linear Program (3), then $x \in \text{opt}(\Phi)$. Conversely, if $x \in \text{opt}(\Phi)$, then the vector (x, y) in which $y_{i,j} = x_i x_j$ whenever $a_{i,j}$ is nonzero is an optimal solution to Linear Program (3), which is a basic optimal solution since an appropriate subset of the constraints $0 \leq x_i \leq 1$ and $0 \leq y_{i,j} \leq 1$ form a basis. ∎

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 3.2, the minimum-cut algorithm) handle this pruning.

3.2 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 [18], $\Phi(x)$ 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.*

Proof: We will show that the minimum s-t cuts in G^Φ correspond to Φ-minimizing protein sequences via Linear Program (3) of Lemma 1. Given a minimum s-t cut in G, let x_i be 1 if v_i is in the s component, and 0 otherwise. Similarly, let $y_{i,j}$ be 1 if $u_{i,j}$ is in the s component, and 0 otherwise. Since no infinite-capacity edge $(u_{i,j}, v_i)$ or $(u_{i,j}, v_j)$ can appear in the cut, if $u_{i,j}$ is in the s-component then v_i and v_j are as well. In terms of the x and y variables, we have $y_{i,j} \leq x_i$ and $y_{i,j} \leq x_j$ whenever $a_{i,j}$ is nonzero, precisely the same constraints as in Linear Program (3). Conversely, any 0-1 assignment (x, y) for which these constraints hold defines an s-t cut that does not include any infinite-capacity edge.

Turning to the objective function, the total capacity of all edges in the cut is

$$\sum_{a_{i,j} \neq 0} a_{i,j}(1 - y_{i,j}) + \sum_{b_i > 0} b_i x_i + \sum_{b_i < 0} -b_i(1 - x_i)$$
$$= \sum_{i,j} a_{i,j} - \sum_{b_i < 0} b_i - \sum_{i,j} a_{i,j} y_{i,j} + \sum_i b_i x_i = K + \Phi'(x, y),$$

where K is a constant and $\Phi'(x, y)$ is the objective function of Linear Program (3). Thus, the capacity of the cut is minimized when $\Phi'(x, y)$ is. The rest follows from Lemma 1. ∎

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.*

Proof: Given a digraph $G = (V, E)$ as input, the Goldberg-Tarjan maximum-flow algorithm takes $O(|V||E| \log(|V|^2/|E|))$ time [14]. We first apply Lemma 2 to Φ to obtain G^Φ. We next use this maximum-flow algorithm to find a minimum s-t cut in G^Φ and then an optimal x from this cut. All these steps take $O(\Delta^2 \log \Delta)$ total time. ∎

3.3 A Compact Representation of Minimum Cuts

A given Φ may have more than one fittest protein sequence. Theorem 5 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 [26], 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 [26]) *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 5 (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 $x \in \mathrm{opt}(\Phi)$ 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.*

Proof: The graph $G^{\Phi}_{s,t}$ is obtained by applying Lemmas 2 and 4. Let κ be the contraction map from Lemma 4. Let $s' = \kappa(s)$ and $t' = \kappa(t)$. The mapping $\rho(i)$ is defined as $\kappa(v_i)$.

To show that $G^{\Phi}_{s,t}$ has at most $n+2$ nodes, consider any node $u_{i,j}$ in G^{Φ}. Let ϕ be the maximum flow used to define $G^{\Phi}_{s,t}$. If $\phi(s, u_{i,j}) = 0$, then $u_{i,j}$ is reachable from s in the residual graph G^{Φ}_{ϕ}, and $u_{i,j}$ is contracted onto the $\kappa(s)$ supernode; if $\phi(s, u_{i,j}) > 0$, then at least one of $\phi(u_{i,j}, v_i)$ or $\phi(u_{i,j}, v_j)$ is nonzero and $u_{i,j}$ is in the same strongly connected component in G^{Φ}_{ϕ} as at least one of v_i and v_j. In either case $u_{i,j}$ is contracted onto a supernode that contains s or some v_i; since the same thing happens to all $u_{i,j}$, there are at most $n+2$ supernodes in $G^{\Phi}_{s,t}$: one for each v_i, plus one for each of s and t.

Using ideals of $\widehat{G}^{\Phi}_{s,t}$ is justified by the observation that requiring t' to be in an ideal and s' to be out of it has no effect on the presence or absence of other nodes, as t' has no predecessors and s' has no successors in $G^{\Phi}_{s,t}$; thus there is a one-to-one correspondence preserving all nodes except s' and t' between the ideals of $G^{\Phi}_{s,t}$ containing t' but not s' and the ideals of $\widehat{G}^{\Phi}_{s,t}$. ∎

Remark. At some additional cost in time, Assumption F1 can be replaced in Theorem 5 by the weaker assumption that Φ is submodular (i.e., that $\Phi(X \cup Y) + \Phi(X \cap Y) \leq \Phi(X) + \Phi(Y)$ for all X, Y, where each protein sequence x in Φ's domain is regarded as the set $X = \{i \mid x_i = 1\}$ for the purposes of taking unions and intersections). The reason is that a representation similar to the Picard-Queyranne graph exists for the set of minima of any such submodular function. These minima form a family closed under union and intersection, and any such family corresponds to the ideals of an appropriate digraph [15, Proposition 10.3.3]. Such a representation can be computed efficiently, as shown by Gabow [12].

Intuitively, what Theorem 5 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 6, any graph may appear as $\widehat{G}^{\Phi}_{s,t}$, with any number of residues mapped to each supernode.

Theorem 6 (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 5, an isomorphism exists between $\widehat{G}^{\Phi}_{s,t}$ and $\hat{G}_0$ mapping each $\rho(i)$ to $\kappa(i)$.*

Proof: Represent each node i in $\hat{G}$ by the variable x_i. For each i, j, let $e_{i,j} = 1$ if there is a directed edge (i, j) in $\hat{G}$, and 0 otherwise. To define Φ, let $a_{i,j} = e_{i,j} + e_{j,i}$ for each $i < j$; and, for each i, let b_i equal i's out-degree $\delta^+(i) = \sum_j e_{i,j}$. Apply Lemma 2 to the resulting function Φ to get a graph G^{Φ}. Define a flow ϕ in G^{Φ} as follows:

$$\begin{array}{rcll} \phi(s, u_{i,j}) & = & e_{i,j} + e_{j,i} & \forall u_{i,j} \\ \phi(u_{i,j}, v_i) & = & e_{i,j} & \forall u_{i,j} \\ \phi(u_{i,j}, v_j) & = & e_{j,i} & \forall u_{i,j} \\ \phi(v_i, t) & = & \delta^+(i) & \forall v_i \end{array}$$

Note that this flow is a maximum flow because it saturates all edges leaving s as well as all edges entering t. (It happens that this is the unique maximum flow, but we do not need this fact, as the Picard-Queyranne construction works for any maximum flow.)

Our next goal is to show that the residual graph G^{Φ}_{ϕ} of this flow contracts to $\hat{G}_0$. G^{Φ}_{ϕ} has the following classes of edges:

$(u_{i,j}, s)$ $(u_{i,j}, v_i)$ $(u_{i,j}, v_j)$	$\forall u_{i,j}$
$(v_i, u_{i,j})$	when $(i, j) \in \hat{G}$
$(v_j, u_{i,j})$	when $(j, i) \in \hat{G}$
(v_i, t)	$\forall v_i$

Since s has no successors and t has no predecessors in G^{Φ}_{ϕ}, the supernodes in $G^{\Phi}_{s,t}$ containing s and t consist of only s and t, respectively. Each node $u_{i,j}$ is in the same strongly-connected component as at least one of v_i or v_j, so no other supernodes exist in $G^{\Phi}_{s,t}$ that do not contain at least one of the nodes v_i. Note that every node-simple path from v_i to v_j in G^{Φ}_{ϕ} is of the form $v_i, u_{i,q_1}, v_{q_1}, u_{q_1,q_2}, \ldots, u_{q_{k-1},j}, v_j$ (with the subscripts of each $u_{q_i,q_{i+1}}$ possibly reversed), which corresponds to a node-simple path $i = q_1, q_2, \ldots, q_k = j$ in $\hat{G}$. The converse also holds. Therefore, i and j are in the same strongly connected component in $\hat{G}$ if only if v_i and v_j are in the same strongly connected component in $G^{\Phi}_{s,t}$. Now recall that for each i, $\rho(i)$ is defined in Lemma 2 as the supernode in $G^{\Phi}_{s,t}$ containing v_i. So an edge from $\rho(i)$ to $\rho(j)$ in $G^{\Phi}_{s,t}$ corresponds to a node-simple path from v_i to v_j $G^{\Phi}_{s,t}$. By the above path-to-path correspondence, every directed edge from $\rho(i)$ to $\rho(j)$ in $G^{\Phi}_{s,t}$ corresponds to an edge

from $\kappa(i)$ to $\kappa(j)$, and vice versa. In summary, $\widehat{G}^{\Phi}_{s,t}$ is isomorphic to $\hat{G}_0$, with the correspondence $\rho(i) \leftrightarrow \kappa(i)$ for all i. ∎

4 Further Tools for Protein Landscape Analysis

4.1 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 7 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 8 shows how to compute a representation of this set similar to that provided by Theorem 5.

Theorem 7 (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 $x \in \mathrm{opt}(\Phi)$ with the minimum weighted Hamming distance $\sum_i w_i |x_i - \hat{x}_i|$ over $\mathrm{opt}(\Phi)$.*

2. *Given Φ as the input, we can find in $O(\Delta^2 \log \Delta)$ time an $x \in \mathrm{opt}(\Phi)$ with the largest (or smallest) possible number of H residues over $\mathrm{opt}(\Phi)$.*

Proof: The statements are proved as follows.

Statement 1. Let ϵ be a positive constant at most $\frac{1}{4Wnc}$, where $W \geq \max |w_i|$ and c is the common denominator of all coefficients $a_{i,j}$ and b_i. Below we show how to find a desired fittest protein sequence by minimizing $\Phi_\epsilon(x) = \Phi(x) + \sum_i \epsilon w_i |x_i - \hat{x}_i|$.

First of all, since x and $\hat{x}$ are 0-1 sequences, $|x_i - \hat{x}_i| = (x_i - \hat{x}_i)^2 = x_i - 2\hat{x}_i x_i + \hat{x}_i$. Then, since $\hat{x}$ is given, $\Phi_\epsilon(x)$ can be minimized using Lemma 3 in $O(\Delta^2 \log \Delta)$ time.

Now suppose that y and z are two protein sequences with $y \in \mathrm{opt}(\Phi)$ and $z \notin \mathrm{opt}(\Phi)$. Then, $\Phi(z) - \Phi(y) \geq \frac{1}{c}$. Also, $|\sum_i w_i |x_i - \hat{x}_i|| \leq Wn \leq \frac{1}{4c}$. Therefore, $\Phi_\epsilon(z) - \Phi_\epsilon(y) \geq \frac{1}{c} - 2Wn \geq \frac{1}{2c}$. Thus every x that minimizes $\Phi_\epsilon(x)$ must also minimize $\Phi(x)$. The Hamming distance term in Φ_ϵ guarantees that from all x that do minimize $\Phi(x)$, minimizing $\Phi_\epsilon(x)$ selects one that also minimizes this distance.

Statement 2. To find an $x \in \mathrm{opt}(\Phi)$ with the largest (respectively, smallest) possible number of H residues, apply Statement 1 with all $w_i = 1$ and all $\hat{x}_i = 1$ (respectively, $\hat{x}_i = 0$). ∎

Suppose we are given fitness functions $\Phi^1, \ldots, \Phi^f$ corresponding to multiple 3D structures, and we wish to find protein sequences that are simultaneously optimal for each 3D structure. A simple approach is to observe that the function $\Phi^* = \Phi^1 + \cdots + \Phi^f$ satisfies Assumption F1, and that any protein sequence that simultaneously optimizes each Φ^ℓ optimizes Φ^*. However, we must check any minimum solution for Φ^* to see that it is in fact a minimum solution for each Φ^ℓ, as it may be that the sets of minimum solutions of the Φ^ℓ have empty intersection. Performing both the optimization of Φ^* and of the f individual fitness functions requires $f + 1$ network flow computations.

It turns out that we can reduce this cost to f network flows at the cost of some additional work to compute a composite Picard-Queyranne graph $G^{\Phi^*}_{s,t}$ directly from the individual graphs $G^{\Phi^\ell}_{s,t}$. This

approach, described in Theorem 8, is especially useful if we have already computed the individual graphs for some other purpose.

Theorem 8 (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 5. 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.*

Proof: By Theorem 5, the conditions below are necessary and sufficient for x to minimize Φ^ℓ:

- $x_i = 1$ if $\rho^\ell(i) = s'$.
- $x_i = 0$ if $\rho^\ell(i) = t'$.
- $x_i = x_j$ if $\rho^\ell(i) = \rho^\ell(j)$.
- $x_i \leq x_j$ if $(\rho^\ell(i), \rho^\ell(j))$ is an edge in $\widehat{G}_{s,t}^{\Phi^\ell}$.

We will build a graph G whose nodes are s, t, and $1, \ldots, n$, and put in an edge (u, v) between any nodes for which the constraint $u \leq v$ is required to minimize some Φ^ℓ. In particular, we have the following classes of edges, for each ℓ, where each class represents one of the above conditions:

- (s, v) and (v, s) whenever $\rho^\ell(v) = s'$.
- (t, v) and (v, t) whenever $\rho^\ell(v) = t'$.
- (u, v) and (v, u) whenever $\rho^\ell(u) = \rho^\ell(v)$.
- (u, v) whenever $(\rho^\ell(u), \rho^\ell(v) \in E(\widehat{G}_{s,t}^{\Phi^\ell}))$.

An assignment of 1 to s, 0 to t, and x_i to each node i satisfies $u \leq v$ whenever $(u, v) \in E(G)$ if and only if all of the constraints required for x to simultaneously minimize all Φ^ℓ are satisfied. Note that such an assignment might not exist.

To convert G into the desired graph $G_{s,t}^{\Phi^*}$, and to check whether there exist any assignments meeting the constraints, contract each strongly connected component of G. If s and t are in the same strongly connected component, no simultaneous fittest solutions exist. Otherwise, let s' in $G_{s,t}^{\Phi^*}$ be the supernode that contains s from G; let t' be the supernode that contains t. Also, for each u in $1, \ldots, n$, let $\rho^*(u)$ be the supernode into which u is contracted. Then x simultaneously minimizes all Φ^ℓ if and only if $x_i = 1$ when $\rho^*(i) = s'$, $x_i = 0$ when $\rho^*(i) = t'$, and $x_i \leq x_j$ when $(\rho^*(i), \rho^*(j))$ is an edge in $G_{s,t}^{\Phi^*}$—precisely the condition that the zeroes in x correspond to nodes in some ideal of $G_{s,t}^{\Phi^*}$ that contains t' but not s'.

To show the running time, observe that constructing the graph G takes $O(\Delta)$ time, which dominates the contraction step. ∎

4.2 The Space of All Fittest Protein Sequences

This section discusses some applications of the representation of the space $\mathrm{opt}(\Phi)$ given by Theorem 5. Theorem 9 gives an algorithm to enumerate this space. Theorem 10 gives an algorithm to compute the diameter of the space in nonnegatively weighted Hamming distance. Theorem 11 gives an algorithm to determine connectivity properties of the space with respect to various classes of mutations.

Theorem 9 (enumerating all protein sequences) *Let Φ be as defined in Assumption F1. Given the $G^{\Phi}_{s,t}$ and ρ defined in Theorem 5 as the input, the protein sequences in* $\mathrm{opt}(\Phi)$ *can be enumerated in $O(n)$ time per protein sequence.*

Proof: An algorithm of Steiner [32] enumerates the ideals of $\widehat{G}^{\Phi}_{s,t}$ in time $O(|V(\widehat{G}^{\Phi}_{s,t}|) = O(n)$ per ideal. For each ideal, invert the mapping ρ (in $O(n)$ time) to recover the corresponding protein sequence x. ∎

Theorem 10 (computing the diameter) *Let Φ be as defined in Assumption F1. Given the $G^{\Phi}_{s,t}$ and ρ defined in Theorem 5 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.*

Proof: Any two fittest protein sequences x and y can differ only at indices i where $\rho(i) \notin \{s, t\}$. Let d be the total weight of indices $i \in \rho^{-1}(V(\widehat{G}^{\Phi}_{s,t}))$. Then, d is an upper bound on the diameter. It is also a lower bound, as $\emptyset$ and $V(\widehat{G}^{\Phi}_{s,t})$ are both ideals of $\widehat{G}^{\Phi}_{s,t}$, and these ideals correspond to two protein sequences at distance d from each other. ∎

We can use $\widehat{G}^{\Phi}_{s,t}$ 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 [18]. 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 [18] 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 $\widehat{G}^{\Phi}_{s,t}$ from Theorem 5. 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 11 (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 5 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 5 as the input, compute the maximal elements of the smallest downward-closed set system Λ such that* $\mathrm{opt}(\Phi)$ *is Λ-connected.*

Proof: The statements are proved as follows.

Statement 1. Let I, I' be the ideals in $\widehat{G}^{\Phi}_{s,t}$ such that the sets of zeros in x, x' are $\rho^{-1}(I), \rho^{-1}(I')$, respectively. Let the maximal elements of Λ be the sets $\rho^{-1}(v)$ over all $v \in I \ominus I'$, where $\ominus$ is the symmetric difference operator. Thus, Λ consists of these sets and all of their subsets. We will show that Λ is the smallest downward-closed set system such that there is a Λ-chain between x and x' in $\mathrm{opt}(\Phi)$.

First, consider some set system Λ' where for some $v \in I \ominus I'$, $A = \rho^{-1}(v)$ is not in Λ'. Recall that x must be constant at the positions indexed by elements of A, where the constant depends on whether or not v is in the ideal in $\widehat{G}^{\Phi}_{s,t}$ corresponding to x. Partition $\mathrm{opt}(\Phi)$ into sets Ω^0 and Ω^1, where Ω^j consists of all z with $z_i = j$ for all $i \in A$. Since x and x' differ on $\rho^{-1}(v)$, one of them is in Ω^0 and the other is in Ω^1. However, since $A \notin \Lambda'$, no protein sequence in Ω^0 is Λ'-adjacent to one in Ω^1. So there is no Λ'-chain between x and x'.

Conversely, to exhibit a Λ-chain from x to x', it suffices to show by iterations a Λ-chain from x to the protein sequence y whose zeroes are given by $\rho^{-1}(I \cap I')$; the case of x' is symmetric. Let $I_0 = I$. If at any iteration $I_i = I \cap I'$, we are done. Otherwise, let v be a maximal element in $I_i - (I \cap I')$. Then, $I_{i+1} = I_i - \{v\}$ is also an ideal. Since $v \in I \ominus I'$, $\rho^{-1}(v) \in \Lambda$ and the protein sequences corresponding to I_i and I_{i+1} are Λ-adjacent. After at most n such iterations, we reach y.

Statement 2. Let x and x' be the protein sequences in $\mathrm{opt}(\Phi)$ with the largest and the smallest possible numbers of H residues, respectively. In other words, x and x' correspond to $\widehat{G}^{\Phi}_{s,t}$ and its empty ideal, respectively. If Λ includes $\rho^{-1}(v)$ for all $v \in V(\widehat{G}^{\Phi}_{s,t})$, then by Statement 1, there are Λ-chains between any $y \in \mathrm{opt}(\Phi)$ and x' and thus between any two protein sequences in $\mathrm{opt}(\Phi)$. If it does not, then there is no Λ-chain between x and x'. In summary, the maximal elements of Λ are the sets $\rho^{-1}(v)$ over all $v \in V(\widehat{G}^{\Phi}_{s,t})$. ∎

4.3 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. This section gives two algorithms that avoid this problem. Theorem 12 describes an algorithm to generate all protein sequences x in order of increasing $\Phi(x)$. Theorem 13 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.

The algorithm for generating all protein sequences x in increasing order by $\Phi(x)$ is based on Lemma 3 and a general technique for enumerating suboptimal solutions to combinatorial optimization problems due to Lawler [21]. It is similar to an algorithm of Vazirani and Yannakakis [34] for enumerating suboptimal cuts. We cannot use the Vazirani-Yannakakis algorithm directly because suboptimal cuts in G^{Φ} might include cuts corresponding to assignments in which $y_{i,j}$ is not equal to $x_i x_j$ for some i, j.

Theorem 12 (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.*

Proof: For any length-k 0-1 sequence $y = y_1, y_2, \ldots, y_k$, let A_y be the set of all length-n 0-1 sequences x with $x_i = y_i$ for each y_i with $1 \leq i \leq k$. Let ε be the empty sequence. Then, A_ε is the set of all length-n sequences. Observe that we can find an element $z \in A_y$ that minimizes $\Phi(z)$ over A_y in time $O(\Delta^2 \log \Delta)$ by setting $z_i = y_i$ in $\Phi(z)$ for each y_i and applying Lemma 3. Furthermore, the set $A_y - \{z\}$ is the disjoint union of the sets A_{r_i} for $k+1 \leq i \leq n$, where $r_i = z_1, z_2, \ldots, z_{i-1}, (1 - z_i)$.

To enumerate x in order of increasing $\Phi(x)$, we maintain a data structure that represents all protein sequences less those already returned as a disjoint union of sets of the form A_y, together with an Φ-minimizing element z for each, organized as a priority queue with key $\Phi(z)$. Initially, the queue contains only (A_ε, z), where $z \in \mathrm{opt}(\Phi)$ is computed using Lemma 3 in time $O(\Delta^2 \log \Delta)$. At each step, the smallest pair (A_y, z) is removed from the priority queue and is replaced by up to n pairs (A_{r_i}, p_i), where p_i is an Φ-minimizing element of A_{r_i}; z is then returned. Each such step requires no more than n applications of Lemma 3, and the cost of the at most $n+1$ priority queue operations is at most $O(n \log(2^n)) = O(n^2)$, giving a total cost of $O(n\Delta^2 \log \Delta)$ per value returned. ∎

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 18(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 12 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 13 gives a tool for plotting F approximately in polynomial time.

Theorem 13 (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. *Let $d \in \{0, 1, \ldots, n\}$.*
   (a) *$F(d_0) = \overline{\Phi}(\epsilon_1) - \epsilon_1 \cdot d_0$. $F(d_k) = \overline{\Phi}(\epsilon_k) - \epsilon_k \cdot d_k$. For $0 < i < k$, $F(d_i) = \overline{\Phi}(\epsilon_i) - \epsilon_i \cdot d_i = \overline{\Phi}(\epsilon_{i+1}) - \epsilon_{i+1} \cdot d_i$.*
   (b) *For $d_i > d > d_{i+1}$ with $0 \leq i < k$, $F(d) \geq \lambda F(d_i) + (1 - \lambda) F(d_{i+1})$, where $\lambda = \frac{d - d_{i+1}}{d_i - d_{i+1}}$.*
4. *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.*

Proof: The statements are proved as follows.

Statement 1. The concavity follows from the minimality of $\overline{\Phi}(\epsilon)$ and the fact that for any fixed x, $\Phi_\epsilon(x)$ is linear in ϵ with slope $|x - \hat{x}|$. Then, the continuous piecewise linearity and the counts of segments and corners follow from the fact that $|x - \hat{x}| \in \{0, 1, \ldots, n\}$.

Statement 2. By the concavity of $\overline{\Phi}$, $d_0 > d_1 > \cdots > d_k$. Let $W = 1 + \sum_{1 \leq i < j \leq n} a_{i,j} + \sum_{i=1}^{n} |s_i|$. For all $\epsilon \leq -W$, $\Phi_\epsilon(x)$ is minimized if and only if x is at distance n from $\hat{x}$. Similarly, for all $\epsilon \geq W$, $\Phi_\epsilon(x)$ is minimized if and only if x is at distance 0 from $\hat{x}$. Therefore, $d_0 = n$ and $d_k = 0$.

Statement 3. Case 3a is straightforward. To prove Case 3b, let x be a protein sequence that has the smallest $\Phi(x)$ over all protein sequences at distance d from $\hat{x}$. Then, $F(d) = \Phi(x)$, and $\Phi_{\epsilon_{i+1}}(x) = F(d) + \epsilon_{i+1} \cdot d$. On the other hand, by the minimality of $\overline{\Phi}$, $\Phi_{\epsilon_{i+1}}(x) \geq \overline{\Phi}(\epsilon_{i+1})$. Furthermore, by Case 3a, $\overline{\Phi}(\epsilon_{i+1}) = F(d_{i+1}) + \epsilon_{i+1} \cdot d_{i+1} = F(d_i) + \epsilon_{i+1} \cdot d_i$. Thus,

$$\begin{aligned} F(d_{i+1}) &\leq F(d) + \epsilon_{i+1} \cdot d - \epsilon_{i+1} \cdot d_{i+1}; \\ F(d_i) &\leq F(d) + \epsilon_{i+1} \cdot d - \epsilon_{i+1} \cdot d_i. \end{aligned}$$

Case 3b follows from algebra and these two inequalities.

Statement 4. For given ϵ and x, let $\text{line}(\epsilon, x)$ be the line through the point $(\epsilon, \Phi_\epsilon(x))$ and with slope $|x - \hat{x}|$. Let L_ϵ (respectively, R_ϵ) be the protein sequence x such that $|x - \hat{x}|$ is the largest (respectively, smallest) possible over $\text{opt}(\Phi_\epsilon)$. Note that L_ϵ, R_ϵ, $\text{line}(\epsilon, L_\epsilon)$, and $\text{line}(\epsilon, R_\epsilon)$ can be computed in $O(\Delta^2 \log \Delta)$ total time using Theorem 7. Furthermore, $\text{line}(\epsilon, L_\epsilon)$ and $\text{line}(\epsilon, R_\epsilon)$ contain the segments of $\overline{\Phi}$ immediately to the left and the right of ϵ, respectively. Consequently, ϵ is a corner of $\overline{\Phi}$ if and only if $\text{line}(\epsilon, L_\epsilon) \neq \text{line}(\epsilon, R_\epsilon)$.

To compute the corners and slopes of $\overline{\Phi}$, we first describe a recursive corner-slope finding subroutine as follows. The subroutine takes as input an interval $[\epsilon', \epsilon'']$ where $\epsilon' < \epsilon''$ together with $\text{line}(\epsilon', R_{\epsilon'})$ and $\text{line}(\epsilon'', L_{\epsilon''})$. It outputs all the corners $(\epsilon, \overline{\Phi}(\epsilon))$ of $\overline{\Phi}$ together with slopes $|L_\epsilon - \hat{x}|$ and $|R_\epsilon - \hat{x}|$ where $\epsilon' < \epsilon < \epsilon''$. There are two cases.

Case 1: $\text{line}(\epsilon', R_{\epsilon'}) = \text{line}(\epsilon'', L_{\epsilon''})$. Then, there is no corner over the interval (ϵ', ϵ''), and thus the subroutine call ends without reporting any new corner or slope.

Case 2: $\text{line}(\epsilon', R_{\epsilon'}) \neq \text{line}(\epsilon'', L_{\epsilon''})$. Then, compute ϵ''' at which $\text{line}(\epsilon', R_{\epsilon'})$ and $\text{line}(\epsilon'', L_{\epsilon''})$ intersect; by the concavity of $\overline{\Phi}$ stated in Statement 1, $\epsilon' < \epsilon''' < \epsilon''$. Also, compute $\text{line}(\epsilon''', L_{\epsilon'''})$ and $\text{line}(\epsilon''', R_{\epsilon'''})$. There are two subcases:

Case 2a: $\text{line}(\epsilon''', L_{\epsilon'''}) \neq \text{line}(\epsilon''', R_{\epsilon'''})$. Then the subroutine returns $(\epsilon''', \overline{\Phi}(\epsilon'''))$ as a new corner together with slopes $|L_{\epsilon'''} - \hat{x}|$ and $|R_{\epsilon'''} - \hat{x}|$ and recurses on the intervals $[\epsilon', \epsilon''']$ and $[\epsilon''', \epsilon'']$.

Case 2b: $\text{line}(\epsilon''', L_{\epsilon'''}) = \text{line}(\epsilon''', R_{\epsilon'''})$. The subroutine returns no new corner or slope but recurses on the intervals $[\epsilon', \epsilon''']$ and $[\epsilon''', \epsilon'']$. In this case, the subroutine has found the line containing a new segment of $\overline{\Phi}$, i.e., the segment through the point $(\epsilon''', \overline{\Phi}(\epsilon'''))$.

This completes the description of the subroutine. The running time of this subroutine is dominated by that for computing $L_{\epsilon'''}$ and $R_{\epsilon'''}$ and thus is $O(\Delta^2 \log \Delta)$.

With this subroutine, we can find the corners and slopes of $\overline{\Phi}$ as follows. Recall W from the proof of Statement 2. Note that if $\epsilon \leq -W$ or $\epsilon \geq W$, then $\overline{\Phi}$ has no corner at ϵ. So we compute $\text{line}(-2W, R_{-2W})$ and $\text{line}(2W, L_{2W})$ and apply the subroutine to the interval $[-2W, 2W]$ to find all the corners and slopes of $\overline{\Phi}$. This algorithm makes $O(n)$ recursive calls to the subroutine since by Statement 1, there are only $O(n)$ corners and segments, and each recursive call finds at least one new corner or segment. The running time of the algorithm is dominated by the total running time of these calls and thus is $O(n\Delta^2 \log \Delta)$ as stated in the statement. ∎

4.4 Tuning the Parameters of the GC Model

This section shows 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. In summary, this section adopts Assumption F2.

Let Π_Φ be the set of indices i with $s_i \neq 0$. The next lemma shows that for any fittest protein sequence x of Φ_β, the set $H(x) \cap \Pi_\Phi$ of H residues with nonzero surface area s_i is monotone in β.

Then, as shown in Theorem 16, to tune β, we only need to consider at most $n+1$ possible values of β.

Lemma 14 *Let Φ be as defined in Assumption F2. Let y and z be any fittest protein sequences for Φ_{β_1} and Φ_{β_0}, respectively. If $\beta_1 < \beta_0$, then $H(y) \cap \Pi_\Phi \supseteq H(z) \cap \Pi_\Phi$.*

Proof: Let $H_1 = H(y)$ and $H_0 = H(z)$. Let $H_{10} = H_1 - H_0$, i.e., the set of indices i for which x_i changes from 1 to 0 when β changes from β_1 to β_0. Similarly, let $H_{01} = H_0 - H_1$. Further, let $\mathcal{B}_{10} = \sum_{i \in H_{10}} s_i$ and $\mathcal{B}_{01} = \sum_{i \in H_{01}} s_i$.

Let $\mathcal{A} = \sum_{i,j \in H_0} a_{i,j} - \sum_{i,j \in H_1} a_{i,j}$. Then, $\Phi_\beta(z) - \Phi_\beta(y) = -\mathcal{A} + \beta(\mathcal{B}_{01} - \mathcal{B}_{10})$. Let $\mathcal{A}_{01} = \sum_{i,j \in H_0 \wedge \{i,j\} \cap H_{01} \neq \emptyset} a_{i,j}$. Let $\mathcal{A}_{10} = -(\mathcal{A} - \mathcal{A}_{01})$, which is the sum of the terms $a_{i,j}$ that Φ_β loses when x_i changes from 1 to 0. Similarly, $\mathcal{A}_{01}$ is the sum of the terms $a_{i,j}$ that Φ_β gains when x_i changes from 1 to 0.

To show $H(y) \cap \Pi_\Phi \supseteq H(z) \cap \Pi_\Phi$, we need to prove $H_{01} \cap \Pi_\Phi = \emptyset$ or equivalently $\mathcal{B}_{01} = 0$. To do so by contradiction, suppose $\mathcal{B}_{01} > 0$. There are two cases:

Case 1: $-\mathcal{A}_{01} + \beta_0 \mathcal{B}_{01} > 0$. Notice that the protein sequence z' with $H(z') = H_0 - H_{01}$ has a smaller fitness value for β_0 than z does, contradicting the minimality of z.

Case 2: $-\mathcal{A}_{01} + \beta_0 \mathcal{B}_{01} \leq 0$. Then, $-\mathcal{A}_{01} + \beta_1 \mathcal{B}_{01} < 0$. Therefor, the protein sequence y' with $H(y') = H_1 \cup H_{01}$ has a smaller fitness value for β_1 than y does, contradicting the minimality of y. ∎

Let $\overline{\Phi}(\beta)$ be the minimum $\Phi_\beta(x)$ over all x. The next lemma characterizes the structure of $\overline{\Phi}(\beta)$. This structure is then used to tune β in Theorem 16.

Lemma 15 *Let Φ be as defined in Assumption F2.*

1. *$\overline{\Phi}$ is a continuous piecewise linear concave function defined on $[0, \infty)$ with at most $n+1$ segments and thus at most $n+1$ corners.*

2. *For all $\beta_1 < \beta_3 < \beta_4 < \beta_2$ where $(\beta_1, \overline{\Phi}(\beta_1))$ and $(\beta_2, \overline{\Phi}(\beta_2))$ are adjacent corners of $\overline{\Phi}$, we have $\mathrm{opt}(\Phi_{\beta_3}) = \mathrm{opt}(\Phi_{\beta_4})$, $\mathrm{opt}(\Phi_{\beta_3}) \subseteq \mathrm{opt}(\Phi_{\beta_1})$, and $\mathrm{opt}(\Phi_{\beta_3}) \subseteq \mathrm{opt}(\Phi_{\beta_2})$. Similarly, for all $\beta_1 < \beta_3 < \beta_4$ where $(\beta_1, \overline{\Phi}(\beta_1))$ is the rightmost corner, we have $\mathrm{opt}(\Phi_{\beta_3}) = \mathrm{opt}(\Phi_{\beta_4})$ and $\mathrm{opt}(\Phi_{\beta_3}) \subseteq \mathrm{opt}(\Phi_{\beta_1})$.*

3. *Given Φ as the input, it takes $O(n\Delta^2 \log \Delta)$ time to find the set of all β such that $(\beta, \overline{\Phi}(\beta))$ is a corner of $\overline{\Phi}$.*

Proof: The statements are proved as follows.

Statement 1. The concavity follows from the minimality of $\overline{\Phi}(\beta)$ and the fact that for any fixed x, $\Phi_\beta(x)$ is linear in β with slope $\sum_{i \in H(x) \cap \Pi_\Phi} s_i$. Then, the continuous piecewise linearity and the counts of segments and corners follow from Lemma 14.

Statement 2. The proofs for the case that $(\beta_1, \overline{\Phi}(\beta_1))$ is the rightmost corner and the complementary case are similar. So we only detail the proof for the former. Note that $(\beta_3, \overline{\Phi}(\beta_3))$ is not a corner. So for every fixed $x \in \mathrm{opt}(\Phi_{\beta_3})$, the line $\Phi_\beta(x)$ goes through the corner $(\beta_1, \overline{\Phi}(\beta_1))$ and the point $(\beta_4, \overline{\Phi}(\beta_4))$. Thus, $x \in \mathrm{opt}(\Phi_{\beta_1})$ and $x \in \mathrm{opt}(\Phi_{\beta_4})$. By symmetry, for every $y \in \mathrm{opt}(\Phi_{\beta_4})$, we have $y \in \mathrm{opt}(\Phi_{\beta_3})$. In summary, $\mathrm{opt}(\Phi_{\beta_3}) = \mathrm{opt}(\Phi_{\beta_4})$ and $\mathrm{opt}(\Phi_{\beta_3}) \subseteq \mathrm{opt}(\Phi_{\beta_1})$.

Statement 3. For any given β and x, let $\mathrm{line}(\beta, x)$ be the line through the point $(\beta, \Phi_\beta(x))$ and with slope $\sum_{i \in H(x) \cap \Pi_\Phi} s_i$. For each β, let L_β (respectively, R_β) be the protein sequence x such that $H(x)$ has the largest (respectively, smallest) possible cardinality over $\mathrm{opt}(\Phi_\beta)$. Note that L_β, R_β, $\mathrm{line}(\beta, L_\beta)$, and $\mathrm{line}(\beta, R_\beta)$ can be computed in $O(\Delta^2 \log \Delta)$ total time using Theorem 7.

Furthermore, $\text{line}(\beta, L_\beta)$ and $\text{line}(\beta, R_\beta)$ contain the segments of $\overline{\Phi}$ immediately to the left and the right of β, respectively. Consequently, for $\beta > 0$, β is a corner of $\overline{\Phi}$ if and only if $\text{line}(\beta, L_\beta) \neq \text{line}(\beta, R_\beta)$. Also, $(0, \overline{\Phi}(0))$ is the leftmost corner, and the segment of $\overline{\Phi}$ to the right of 0 is contained by $\text{line}(0, R_0)$.

To compute the corners of $\overline{\Phi}$, we first describe a recursive corner-finding subroutine as follows. The subroutine takes as input an interval $[\beta_1, \beta_2]$ where $\beta_1 < \beta_2$ together with $\text{line}(\beta_1, R_{\beta_1})$ and $\text{line}(\beta_2, L_{\beta_2})$. It outputs all the corners $(\beta, \overline{\Phi}(\beta))$ of $\overline{\Phi}$ with $\beta_1 < \beta < \beta_2$. There are two cases.

Case 1: $\text{line}(\beta_1, R_{\beta_1}) = \text{line}(\beta_2, L_{\beta_2})$. Then, there is no corner over the interval (β_1, β_2), and thus the subroutine call ends without reporting any new corner.

Case 2: $\text{line}(\beta_1, R_{\beta_1}) \neq \text{line}(\beta_2, L_{\beta_2})$. Then, compute β_3 at which $\text{line}(\beta_1, R_{\beta_1})$ and $\text{line}(\beta_2, L_{\beta_2})$ intersect; by the concavity of $\overline{\Phi}$ stated in Lemma 15(1), $\beta_1 < \beta_3 < \beta_2$. Also, compute $\text{line}(\beta_3, L_{\beta_3})$ and $\text{line}(\beta_3, R_{\beta_3})$. There are two subcases:

Case 2a: $\text{line}(\beta_3, L_{\beta_3}) \neq \text{line}(\beta_3, R_{\beta_3})$. Then the subroutine returns $(\beta_3, \overline{\Phi}(\beta_3))$ as a new corner and recurses on the intervals $[\beta_1, \beta_3]$ and $[\beta_3, \beta_2]$.

Case 2b: $\text{line}(\beta_3, L_{\beta_3}) = \text{line}(\beta_3, R_{\beta_3})$. The subroutine returns no new corner but recurses on the intervals $[\beta_1, \beta_3]$ and $[\beta_3, \beta_2]$. In this case, the subroutine has found the line containing a new segment of $\overline{\Phi}$, i.e., the segment through the point $(\beta_3, \overline{\Phi}(\beta_3))$.

This completes the description of the subroutine. The running time of this subroutine is dominated by that for computing L_{β_3} and R_{β_3} and thus is $O(\Delta^2 \log \Delta)$.

With this subroutine, we can find the corners of $\overline{\Phi}$ as follows. If every $s_i = 0$, then $(0, \overline{\Phi}(0))$ is the only corner. Otherwise, let β_∞ be $1 + \sum_{1 \leq i < j \leq n} a_{i,j}$ divided by the smallest nonzero s_i. Note that for every $\beta \geq \beta_\infty$, $\overline{\Phi}$ has no corner at β, Then, we compute $\text{line}(0, L_0)$ and $\text{line}(0, R_{\beta_\infty})$. We report the leftmost corner $(0, \overline{\Phi}(0))$ and apply the subroutine to the interval $[0, \beta_\infty]$ to find all the other corners of $\overline{\Phi}$.

This algorithm makes $O(n)$ recursive calls to the subroutine since by Lemma 15, there are only $O(n)$ corners and segments, and each recursive call finds at least one new corner or segment. The running time of the algorithm is dominated by the total running time of these calls and thus is $O(n\Delta^2 \log \Delta)$ as stated in the lemma. ∎

Theorem 16 (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 $\text{opt}(\Phi_\beta)$ is the minimum over all possible β.*

Proof: The proofs for the cases of unweighted and weighted Hamming distances are similar. So we only detail the proof for the unweighted case. Our algorithm for finding all distance-minimizing choices of β has three stages.

Stage 1. Use Lemma 15(3) to find all the corners of $\overline{\Phi}$.

Stage 2. For each corner $(\beta, \overline{\Phi}(\beta))$, use Theorem 7 to compute the closest Hamming distance d_β between $\hat{x}$ and any protein sequence in $\text{opt}(\Phi_\beta)$. Let $d_{\min}$ be the smallest d_β over all corners. Then, by Lemma 15(2), report all β with $d_\beta = d_{\min}$ as desired choices of distance-minimizing β.

Stage 3. Consider each segment of $\overline{\Phi}$. Let β_1 and β_2 be the vertical coordinates of the left and right endpoints of the segment. Find a suitable β_3 in the open interval (β_1, β_2) as follows. If β_2 is finite, then set $\beta_3 = (\beta_1 + \beta_2)/2$; otherwise, set $\beta_3 = \beta_1 + 1$. Then use Theorem 7 to compute the closest unweighted or weighted Hamming distance d_{β_3} between $\hat{x}$ and any protein sequence in $\text{opt}(\Phi_{\beta_3})$. If $d_{\beta_3} = d_{\min}$, then by Lemma 15(2), report that every β in the interval (β_1, β_2) is a desired distance-minimizing β.

This completes the description of the algorithm. By Lemma 15(3), Stage 1 takes $O(n\Delta^2 \log \Delta)$ time. By Theorem 7 and Lemma 15 (1), Stages 2 and 3 also take $O(n\Delta^2 \log \Delta)$ time. Thus, the total running time is as stated in the theorem. ∎

5 Computational Hardness Results

Theorem 17 (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 $\hat{x}$.*

Proof: Note that each of the problems is in #P, because we can recognize an element of $\mathrm{opt}(\Phi)$ in polynomial time using Lemma 3. So to prove #P-completeness we must only show that each problem is #P-hard.

Statement 1. Reduce from the problem of counting the number of ideals in a dag, which is #P-hard [28]. Given a dag $\hat{G}$, apply Theorem 6 to get a function Φ for which $\widehat{G}^{\Phi}_{s,t}$ is isomorphic to $\hat{G}$. By Theorem 5, counting $\mathrm{opt}(\Phi)$ is then equivalent to counting the number of ideals of $\widehat{G}^{\Phi}_{s,t} \cong \hat{G}$.

Statement 2. The problem in Statement 1 is a special case of the problem in this statement.

Statement 3. Using the same construction as in Statement 1, we can reduce from the problem of computing the average cardinality of an ideal in $\hat{G}$. To see that this latter problem is #P-hard, suppose that we can compute the average cardinalities of ideals in an n-node $\hat{G}$ and in an augmented graph $\hat{G}'$ obtained from $\hat{G}$ by adding a single new node s and edges from every $v \in V(\hat{G})$ to s. Let c be the average for $\hat{G}$ and c' the average for $\hat{G}'$. Let N be the number of ideals in $\hat{G}$. Then $c = K/N$ for some K, while $c' = (K + n + 1)/(N + 1)$, since the only new ideal in $\hat{G}'$ consists of s and all other nodes, and thus has size $n + 1$. Solving for N gives $N = (n + 1 - c')/(c' - c)$, which can be computed from c, c', and n.

Statement 4. This problem has the problem in Statement 3 as a special case with all $\hat{x}_i = 0$.

Statement 5. To reduce the problem of counting protein sequences in $\mathrm{opt}(\Phi)$ to counting protein sequences at a given unweighted Hamming distance, take the dag $\widehat{G}^{\Phi}_{s,t}$ given by Theorem 5, and add to each node v_i a node q_i with edges (v_i, q_i) and (q_i, v_i). Apply Theorem 6 to this new graph $\hat{G}'$ to obtain a function Φ' for which $\mathrm{opt}(\Phi')$ is in one-to-one correspondence with the set of ideals of the strongly-connected component graph $\hat{G}'_0$ of $\hat{G}'$. Each strongly connected component of $\hat{G}'$ consists of v_i and q_i for some i, so $\hat{G}'_0$ is isomorphic to $\widehat{G}^{\Phi}_{s,t}$. Now we choose $\hat{x}$ with $\hat{x}_{v_i} = 0$ and $\hat{x}_{q_i} = 1$. Then, the contribution to $|x - \hat{x}|$ of each pair x_{v_i}, x_{q_i} is 1 regardless of their common value. Thus, the

number of Φ-minimizing protein sequences equals the number of Φ'-minimizing protein sequences at distance d from $\hat{x}$, where d is the number of nodes in $\widehat{G}^{\Phi}_{s,t}$. ∎

Theorem 18 (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 $F(d)$.*

Proof: Statement 2 follows from the fact that the problem in Statement 1 can be reduced to the problem in this statement in polynomial time. Statement 1 is proved as follows.

Since we can recognize Φ-minimizing protein sequences using Lemma 3, the problem is clearly in NP. To show that it is NP-hard, we reduce from PARTIALLY ORDERED KNAPSACK, problem MP12 from Garey and Johnson [13, pp. 247–248].

The input to PARTIALLY ORDERED KNAPSACK consists of a partially-ordered set U, each element u of which is assigned a size $s(u) \in \mathbf{Z}^+$ and a value $v(u) \in \mathbf{Z}^+$, together with a upper bound B_1 on total size and a lower bound B_2 on total value. The problem is to determine whether there exists an ideal I in U such that $\sum_{u \in I} s(u) \leq B_1$ and $\sum_{u \in I} v(u) \geq B_2$. Garey and Johnson note that the problem, even with $s(u) = v(u)$ for all $u \in U$, is NP-complete in the strong sense (meaning that there is some polynomial bound on the size of all numbers in the input with which it remains NP-complete).

Given an instance of PARTIALLY ORDERED KNAPSACK with $s(u) = v(u)$ for all u and all numbers bounded by some polynomial p, build a graph $\hat{G}$ where each $u \in U$ is represented by a clique $C(u)$ of $s(u)$ nodes, and there is an edge from $C(u)$ to $C(u')$ if and only if $u \prec u'$. Note that because $s(u) \leq p$, $\hat{G}$ has polynomial size. Apply Theorem 6 to generate a function Φ (in polynomial time) such that $\widehat{G}^{\Phi}_{s,t}$ is isomorphic to the component graph $\hat{G}_0$ obtained by contracting all strongly connected components of $\hat{G}$. Since the strongly connected components of $\hat{G}$ are precisely the cliques $C(u)$, $\widehat{G}^{\Phi}_{s,t} \cong \hat{G}_0$ is isomorphic to U, interpreted as a dag. In particular any ideal of $\widehat{G}^{\Phi}_{s,t}$ corresponds to an ideal I of U. Let $N = \sum_{u \in U} s(u)$. The norm of the corresponding vector $|x|$ is $N - \sum_{u \in I} |C(u)| = N - \sum_{u \in I} s(u) = N - \sum_{u \in I} v(u)$. Set $d_1 = N - B_1$, $d_2 = N - B_2$, and we have the problem stated in the theorem. ∎

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 [18] 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 [10]. In accordance to the GC model, each of the chosen native

Name	Solvent/Length	Length	α/β	% Similarity	Description
1a7m	51.23	180	-295.8	74.44	cytokine
1a8y	81.37	338	-155.7	73.37	Ca binding protein
1ab3	399.05	88	-326.7	78.41	ribosomal protein
1ab7	451.48	89	-79.5	80.90	ribonuclease inhibitor
1agi	384.30	125	-93	77.60	endonuclease
1air	173.96	352	-0.3	69.89	pectate lyase
1b71	374.23	191	-23.7	69.11	electron transport
1ble	498.45	161	-269.4	72.67	phosphotransferase
1bpi	1453.75	58	-31.5	68.97	proteinase inhibitor
1bw3	732.31	125	-33.9	70.40	lectin
1clh	607.48	166	-9.3	69.28	cyclophilin
1ehs	2178.29	48	-32.1	72.92	enterotoxin
1gym	392.97	296	-4.5	73.99	phospholipase
1nar	447.48	289	-6.3	75.78	plant seed protein
1prn	498.38	289	-2.7	56.40	porin
1thv	741.41	207	-10.5	71.01	sweet tasting protein
1xnb	871.63	185	-135.6	65.95	glycosidase
2aak	1130.39	150	-149.1	78.67	ubiquitin conjugation
2bnh	412.43	456	-253.8	78.51	ribonuclease inhibitor
2cba	773.98	258	-2.4	73.64	lyase
2erl	5064.83	40	-47.7	80.00	pheromone
2stv	1153.80	184	-3.6	64.67	viral coat protein
6yas	904.99	256	-12.6	67.19	lyase
8cho	2054.72	125	-423.6	72.80	isomerase
9rat	2126.95	124	-89.4	75.00	ribonuclease A
1aaj	51.89	105	-155.1	70.48 (72)	electron transport
1aba	125.39	87	-245.4	78.16 (70)	electron transport
1bba	584.25	36	-69.6	66.67 (58)	pancreatic hormone
1brq	272.54	174	-27.3	72.99 (71)	retinol transport
1cis	780.88	66	-454.8	69.70 (64)	lysozyme
1hel	539.14	129	-18.6	76.74 (78)	fatty acid bind protein
1ifb	583.57	131	-40.5	79.39 (70)	Ca binding protein
3cln	835.80	143	-309.9	72.03 (70)	ribonuclease
3rn3	1018.66	124	-88.8	72.58 (69)	electron transport

Table 1: This table gives some statistics of our computed fittest H/P protein sequences for the 34 empirical protein 3D structures chosen from PDB. Column 1 contains the PDB name of the native protein sequence, where the proteins from Kleinberg [18] are in bold. Column 2 shows the length normalized solvent accessibility, i.e., $\sum_{i \in H(x)} s_i/n$, of the computed fittest protein sequence. Column 4 lists the protein sequence length n, i.e., the count of amino acids. Column 4 provides the computed optimal ratio of the α and β parameters (see text). Column 5 displays the percentage of similarity of the computed fittest protein sequence to the native protein sequence. Column 6 describes the function of the native protein.

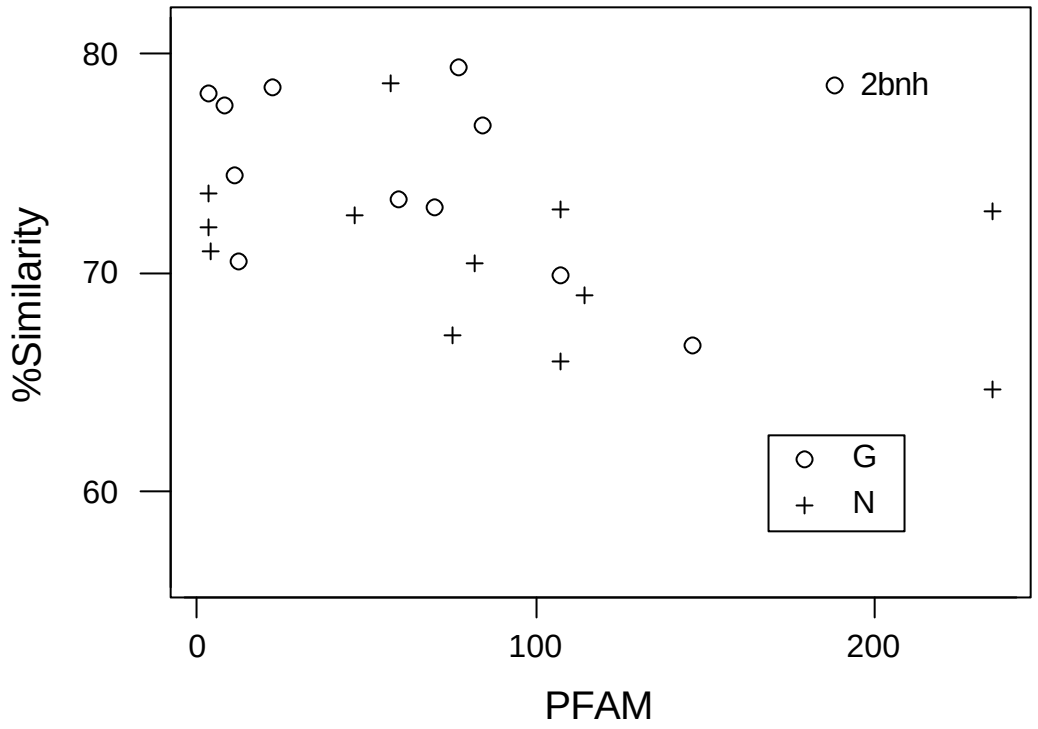

Figure 1: This plot displays the relationship between the percent similarity of computed fittest protein sequences to native protein sequences versus the PFAM family size of native protein sequences. Globular native proteins are shown as open circles while nonglobular ones are shown as crosses.

protein sequences was converted into a binary H/P sequence following Sun *et. al.* [33], where A, C, F, I, L, M, V, W, Y are H, and the other amino acids are P.

We used Equation 1 in the GC model to calculate fitness values of protein sequences to determine minimal energy values and consequently to compute the "fittest" protein sequences. These fitness values consist of two terms in Equation 1. The first term accounts for the idea that hydrophobic residues tend to cluster together due to stacking forces from the solvent. The second term accounts for the idea that hydrophobic residues tend to avoid solvent accessible surfaces of the molecule. The arbitrary parameters α and β represent scaling factors for the relative importance of these two tendencies. We expected the appropriate ratio of these two values to depend on the type of a protein (globular, nonglobular, and membrane) and the length of the protein. Therefore, we optimized the scale of the two parameters to find a ratio that maximizes the similarity of a fittest protein sequence to the native protein sequence, similarly in spirit to Kleinberg's scaling algorithm [18].

Results of this optimization are shown in Table 1. As anticipated, our algorithms computed fittest protein sequences that are closer to native protein sequences than found by Kleinberg [18], whose results are shown in parenthesis in Table 1. (Note that protein 1aaj is an exception to this improvement on proximity—perhaps due to the fact that the input data are not exactly the same.) However, proximity to native protein sequences is not a good proxy for biological relevance of the algorithms. Determination of protein 3D structure involves an energy landscape given by a statistical thermodynamic energy function $E(\mathcal{X}, \mathcal{D})$ where $\mathcal{X}$ is the set of amino acid sequences and $\mathcal{D}$ is the set of possible folded 3D structures. On the one hand, for a fixed protein sequence x, the 3D structure D

is determined by a temperature-dependent folding process that minimizes $E(x, D)$ over $\mathcal{D}$. *Ab initio* solutions to this problem for protein sequences of practical length currently do not exist. On the other hand, for a fixed 3D structure D, no known thermodynamic reasons connect the folding process with a protein sequence x that minimizes $E(x, D)$ over $\mathcal{X}$. However, we can invoke evolutionary processes as possibly selecting for those protein sequences that produce the most stable 3D structures, i.e., those with the lowest $E(x, D)$, at a given temperature. In other words, given a suite of protein sequences that fold into a particular 3D structure to perform a biological function, there might be selection for the most thermodynamically stable protein sequences.

From the view point of evolutionary selection, the difference between computed fittest protein sequences and native protein sequences may be attributed to three factors. First, the GC toy thermodynamic model is inappropriate. Second, the biological function of a protein actually requires structural lability. Last, a native protein is part of a diverse family and other members of the family lie closer to the computed fittest protein sequence. This last factor can be augmented by the argument that if a computed fittest protein sequence exists in nature but is very different from the native protein sequence, it is likely that many other protein sequences (thus a diverse family of protein sequences) exist in nature and fold into the same or similar 3D structures. All of these factors are likely to play in the data shown in Table 1. However, we 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 [4]. The database contained information on the putative family size of 25 of our 34 chosen native proteins. Figure 1 shows the plot of the percent similarity of computed fittest protein sequences to native protein sequences versus the PFAM family size of native proteins. In the figure, globular proteins are shown as circles and nonglobular ones as crosses. There is a negative linear trend as suggested by our conjecture. Linear regression is nearly significant at 0.05 level with $p = 0.088$. The figure shows three outliers, 2bnh, 2stv, and 8cho. Of these, 2bnh is an exceedingly strange 3D structure with a protein sequence of alpha helixes forming a horseshoe shaped sheet, resulting in a 3D structure that is very deviant from globular proteins which are the genesis of the original thermodynamic model. Leaving out this outlier results in a significant linear regression with $p = 0.015$.

There is still considerable uncertainty about the appropriateness of the GC toy model. The average percentage of the hydrophobic residues is 42% in the native protein sequences compared to 35% in the computed fittest protein sequences. More importantly, the standard deviation of the percentage of hydrophobic residues is 0.054 in the native protein sequences compared to 0.143 for the computed fittest protein sequences. Thus, the percentage of hydrophobic residues is relatively constant in the native protein sequences, reflecting perhaps a functional need or unknown structural factors. In contrast, the percentage of hydrophobic residues in a computed fittest protein sequence tends to vary depending on the 3D structure. This suggests that it might be important to introduce a hydrophobic residue percentage constraint into optimization algorithms in the future as suggested in the sliding algorithm of Kleinberg [18]. Nevertheless, our preliminary results show that even such a simplified toy model might be useful for exploratory investigations of protein evolution especially when coupled to computationally efficient algorithms to allow systematic investigation of the roughly 13,000 empirical protein 3D structures. We are currently planning a large-scale analysis of further empirical protein 3D structures; the results will be reported in a subsequent paper.

Acknowledgments

We wish to thank Jon Kleinberg for generously providing LaTeX entries for many of the references; thank Lisa Fleischer and Hal Gabow for their help in tracking down references used to generalize Theorem 5 for submodular functions; and thank Mark Gerstein for helpful discussions.

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] A. Babajide, I. Hofacker, M. Sippl, and P. Stadler. Neutral networks in protein space: A computational study based on knowledge-based potentials of mean force. *Folding and Design*, 2:261–269, 1997.

[3] 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.

[4] 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.

[5] B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology*, 5(1):27–40, 1998.

[6] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, pages 423–466, 1998.

[7] J. M. Deutsch and T. Kurosky. New algorithm for protein design. *Physical Review Letters*, 76:323–326, 1996.

[8] K. A. Dill, S. Bromberg, K. Yue, K. Fiebig, D. Yee, P. Thomas, and H. S. Chan. Principles of protein folding — A perspective from simple exact models. *Protein Science*, 4:561–602, 1995.

[9] K. E. Drexler. Molecular engineering: An approach to the development of general capabilities for molecular manipulation. *Proceedings of the National Academy of Sciences of the U.S.A.*, 78:5275–5278, 1981.

[10] F. Eisenhaber and P. Argos. Improved strategy in analytic surface calculation for molecular systems: Handling of singularities and computational efficiency. *Journal of Computational Chemistry*, 14(N11):1272–1280, 1993.

[11] 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.

[12] H. N. Gabow. Applications of a poset representation to edge connectivity and graph rigidity. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1991.

[13] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, NY, 1979.

[14] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, Oct. 1988.

[15] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer-Verlag, New York, NY, 1988.

[16] W. E. Hart. On the computational complexity of sequence design problems. In *Proceedings of the 1st Annual International Conference on Computational Molecular Biology*, pages 128–136, 1997.

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

[18] J. M. Kleinberg. Efficient algorithms for protein sequence design and the analysis of certain evolutionary fitness landscapes. In *Proceedings of the 3rd Annual International Conference on Computational Molecular Biology*, pages 226–237, 1999.

[19] K. F. Lau and K. A. Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22:3986–3997, 1989.

[20] K. F. Lau and K. A. Dill. Theory for protein mutability and biogenesis. *Proceedings of the National Academy of Sciences of the U.S.A.*, 87:638–642, 1990.

[21] E. L. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18:401–405, 1972.

[22] D. Lipman and W. Wilbur. Modeling neutral and selective evolution of protein folding. *Proceedings of Royal Society of London Series B*, 245:7–11, 1991.

[23] K. M. Merz and S. M. L. Grand, editors. *The Protein Folding Problem and Tertiary Structure Prediction*. Birkhauser, Boston, MA, 1994.

[24] C. Micheletti, F. Seno, A. Maritan, and J. Banavar. Design of proteins with hydrophobic and polar amino acids. *Proteins: Structure, Function, and Genetics*, 32:80–87, 1998.

[25] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Upper Saddle River, NJ, 1982.

[26] 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.

[27] J. Ponder and F. M. Richards. Tertiary templates for proteins. *Journal of Molecular Biology*, 193:63–89, 1987.

[28] J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, Nov. 1983.

[29] 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.

[30] E. I. Shakhnovich and A. M. Gutin. A new approach to the design of stable proteins. *Protein Engineering*, 6:793–800, 1993.

[31] J. M. Smith. Natural selection and the concept of a protein space. *Nature*, 225:563–564, 1970.

[32] G. Steiner. An algorithm to generate the ideals of a partial order. *Operations Research Letters*, 5:317–320, 1986.

[33] 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.

[34] V. V. Vazirani and M. Yannakakis. Suboptimal cuts: Their enumeration, weight and number (extended abstract). In W. Kuich, editor, *Lecture Notes in Computer Science 623: Proceedings of the 19th International Colloquium on Automata, Languages, and Programming*, pages 366–377. Springer-Verlag, New York, NY, 1992.

[35] K. Yue and K. A. Dill. Inverse protein folding problem: Designing polymer sequences. *Proceedings of the National Academy of Sciences of the U.S.A.*, 89:4163–4167, 1992.