Optimal Hydrophobic and Polar Interaction Models for Protein Inverse Folding Problem

BY

Ying Wang
B.S. Beijing Jiaotong University, 2001
M.S. Beijing Jiaotong University, 2005

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Bioengineering
in the Graduate College of the
University of Illinois at Chicago, 2011

Chicago, Illinois

Defense Committee:

Jie Liang, Chair and Advisor
Yang Dai
Bhaskar DasGupta, Computer Science

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Dr. Jie Liang. He has provided precious guidance and support for my research project. Without his help, I can not accomplish this thesis. I would also like to thank Dr. Bhaskar DasGupta and Dr. Yang Dai for the aid and suggestions on my thesis. At last, I also want to say “Thank You” to the members of Jie’s lab for helpful discussion.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURES PAGE

LIST OF ABBREVIATIONS

GCSE	Grand Canonical Sequence Evolution
H	Hydrophobic
LP	Linear Programming
P	Polar
PDB	Protein Data Bank

SUMMARY

Protein inverse folding problem is a natural inverse problem to protein structure prediction: given a target structure in three dimensions, we desire to design an amino acid sequence that is likely to fold to the given structure. However, for a structure of length N, there will be a total of 20^N possible sequences even without considering different orientations of the side chains ("rotamer configuration"), since there are 20 amino acid types. Exhaustive enumeration of all possible sequences and then selection of the best sequence for given structure is beyond the scope of the computational power. Developing an appropriate model to study the protein inverse folding problem is challenging.

The inverse folding problem is considered as an optimization problem based on a fitness function. A model of Sun *et al*. (1995) casted this problem as an optimization problem on a space of sequences of hydrophobic (H) and polar (P) monomers; the goal is to find a sequence which achieves a dense hydrophobic core with few solvent-exposed hydrophobic residues without guarantee of optimality or near-optimality (Sun *et al.*, 1995); Kleinberg converted this problem to an efficient flow network algorithm in order to construct optimal sequences (Kleinberg, 1999).

In this study, our tasks encompass finding out the optimal sequences for a given protein three-dimensional structure. After simulating and verifying Kleinberg's flow network algorithm, we explore a new linear fitness function based on the GCSE model of Sun *et al.*(1995) and

Kleinberg's flow network transformation(1999), and we have solved this protein inverse folding problem by providing a simple and efficient linear programming method to construct optimal sequences. We demonstrate our model's effectiveness by implementation on 23 structures drawn from the Protein Data Bank. Furthermore, we also consider the extensions of this linear programming method with specified residue composition in a sequence, as a way to overcome the limitations that a sharp imbalance in the ratio of H to P residues prevents designed sequences from having a high degree of agreement with the natural sequence in most cases.

The linear programming method for solving the inverse folding problem provides a general model for studying a variety of problems in protein design, including the design of new proteins and modification of existing proteins in order to alter their functions, structures, and folding properties. It can be further extended to solve the inverse binding problem, namely, to seek to design protein sequence for the optimal binding, which arises in many situations. This has significant importance in protein design, protein engineering, and also in searching therapeutic agents.

1. INTRODUCTION

1.1 Background

Protein sequence design is a natural inverse problem to protein structure prediction: given a target structure in three dimensions, we desire to design an amino acid sequence that is likely to fold to the given structure. A model of Sun *et al*. (1995) casted this problem as an optimization problem on a space of sequences of hydrophobic (H) and polar (P) monomers; the goal is to find a sequence which achieves a dense hydrophobic core with few solvent-exposed hydrophobic residues, but Sun *et al.*'s heuristic method did not guarantee the optimality or near-optimality (Sun *et al.*, 1995); Hart subsequently raised the computational tractability of constructing an optimal sequence in Sun *et al*.'s model as an open question (Hart, 1997); Kleinberg converted this problem to an efficient flow network algorithm in order to construct optimal sequences (Kleinberg, 1999). Here, we solve the protein inverse folding problem by developing a new solution based on network linear programming, which provides a general model for studying a variety of problems in protein design, instead of seeking to improve the computing time of the algorithm for finding the optimal sequence. Solving the protein inverse folding problem is important with applications in the design of new proteins and modification of existing proteins to alter their functions, structures, and folding properties.

1.2 Statement of the problem

Understanding the principles of how proteins adopt their native three-dimensional structures is a fundamental problem in biology. The intensively studied problem of protein structure prediction begins with a given amino acid sequence and seeks to characterize, by computational means, the structure or range of structures which this sequence will adopt under physiological conditions (Merz and LeGrand, 1994). There exists a typical “inverse” version of this structure prediction problem, which is determined the subject of several previous studies (Banavar *et al.*, 1998; Deutsch and Kurosky, 1996; Drexler, 1981; Hart, 1997; Ponder and Richards, 1987; Sun *et al.*, 1995; Shakhnovich and Gutin, 1993; Yue and Dill, 1992). In these studies, a three-dimensional protein structure was given, and the goal is to identify the sequence or collection of sequences most likely to fold to this structure. The protein inverse folding problem can be regarded as a “bead coloring” problem (Sun *et al.*, 1995), namely, the given structure can be thought as a chain of “colorless beads” with coordinates on each atom, *i.e.* generic amino acids that do not yet have side-chain structural identities. The design process then “paints” each bead a “color”, representing each of the 20 amino acids.

In view of the observation that proteins can adopt only a limited number of folds (Chothia 1992; Banavar and Maritan 2003), efficient and robust algorithms to identify all possible sequences for a given structure help to assign structures to a large, rapidly increasing number of sequences in the databases. The protein sequence design problem, i.e. the protein inverse folding

problem, is also very important. For example, in addition to the design of new proteins that fold to a desired conformation, solving the inverse folding problem can shed light on understanding the principles underlying protein folding and the variability in the sequences of naturally occurring proteins (Zou and Saven 2000).

However, for a structure of length N, there will be a total of 20^N possible sequences even without considering different orientations of the side chains ("rotamer configuration"), since there are 20 amino acid types. Exhaustive enumeration of all possible sequences and then selection of the best sequence for given structure is beyond the scope of the computational power. Developing an appropriate model to study the protein inverse folding problem is challenging.

1.3 Literature review

In the recent studies, a lot of models and algorithms have been developed for solving the protein inverse folding problem. These include stochastic and deterministic methods. For example, Desmet *et al.* (1992) proposed a dead-end elimination algorithm to screen out improbable sequences efficiently. Hellinga and Richards (1994) used Monte Carlo methods. Desjarlais and Handel (1995) used genetic algorithms, and Deutsche and Kurosky (1996) used simulated annealing. Saven and Wolynes (1997) used statistical mean field theory based methods to determine site-specific probabilities for most probable amino acid types using deterministic optimization algorithms. Sanjeev *et al.* (2001) used a graph spectral method, which ranks the sites

for amino acid types with very little computation and thereby designs a sequence. Raha *et al.* (2000) used a combinational algorithm depending on filtering, sampling, and optimization procedures, and a relatively straightforward scoring function.

For the aforesaid methods and algorithms, there is no commonly accepted criterion to distinguish the folded structure from other conformation of a protein based on its amino acid sequence. Usually, the criterions include minimum energy, maximum gap in energy from the average energy of unfolded conformations, maximum entropy, etc. Based on the above criterions, a set of related approaches were developed in the biophysics community (Sun *et al.*, 1995; Shakhnovich and Gutin, 1993; Deutsch and Kurosky, 1996; Gupta *et al.*, 2005; Lippow and Tidor, 2007). These approaches regarded protein inverse folding problem as a global optimization problem on the space of amino acid sequences. The optimization problem is to set up an objective function, i.e. the fitness function, and select the solution as the optimal sequence for the inverse folding problem. The simplest approach is to minimize the energy of the target structure by varying sequence. Any of such design processes must solve two problems (Yue and Dill, 1992): (1) positive design – the designed sequences should have low free energy (minimize the fitness function) in the target structure; and (2) negative design – there should be very few other "competing" structures in which the designed sequence has comparable free energy (fitness function).

In their studies of inverse folding problem, often two types of residues are considered:

Hydrophobic (H) and Polar (P). This is supported by a widely accepted finding that the hydrophobicity of some amino acid types is one of the principal driving forces for protein folding. Kamtekar *et al.* (1993) used the burial algorithm to design protein sequence in H/P binary format with a given structure. It assigns an H monomer to any solvent-inaccessible position in the given structure, and P otherwise. Kamtekar *et al.* (1993) started with the natural coordinates of a known biological four-helix bundle, and then designed sequences to fold to it. Their design strategy was simply to bury H monomers and expose P monomers. This burial algorithm simply codified the standard lore that hydrophobic residues should be buried. The "Grand Canonical Sequence Evolution (GCSE)" method of Sun *et al.* (1995) is also based on the H/P binary model. Sun *et al.* (1995) proposed an energy fitness function in the GCSE model for identifying search for folding sequences by using genetic algorithms, and the designed sequences only include H and P. There is no guarantee that this heuristic method will find sequences with optimality or near-optimality (Sun *et al.*, 1995). In addition, the GCSE model does not constrain the overall composition when generating heteropolymeric sequences upon optimization. Kleinberg (1999) formulated the fitness function in GCSE model with network flow with a polynomial running time (Kleinberg, 1999). Through the development of an appropriate fitness function, these approaches attempt implicitly to capture the competing requirements of positive design and negative design.

1.4 Purpose of the study

We consider the protein inverse folding problem as an optimization problem based on a

fitness function. There are three steps. First of all, identifying of the criterion of a fitness function, i.e. the establishment of the objective function in optimization problem; secondly, searching a sequence space to find the optimal sequences; and thirdly, testing and verifying the model's efficiency and robustness.

In this study, our tasks encompass finding out the optimal sequences for a given protein three-dimensional structure. We explore a new fitness function based on the GCSE model of Sun *et al.* and Kleinberg's network flow transformation, and we have solved this protein inverse folding problem by providing a simple and efficient linear programming method to construct optimal sequences. We demonstrate our model's effectiveness by implementation on 23 structures drawn from the Protein Data Bank. Furthermore, we also consider the extensions of our network linear programming method with specified residue composition in a sequence, as a way to overcome the limitations that a sharp imbalance in the ratio of H to P residues prevents designed sequences from having a high degree of agreement with the natural sequence in most cases.

The linear programming method for solving the inverse folding problem provides a general model for studying a variety of problems in protein design, including the design of new proteins and the prescribed modification of existing proteins in order to alter their functions, structures, and folding properties. It can be further extended to solve the inverse binding problem, namely, to seek to design protein sequence for the optimal binding, which arises in many situations. Given a protein structure with a fixed binding region, how can we design an optimal

sequence that is most stable among all sequences encoding the fixed binding region? Alternatively, given a combined structure of protein-ligand complex or protein-protein complex, if the sequence of the ligand or the second protein is fixed, how to obtain the optimal sequence for the first protein to achieve most stable complex structure? We can further allow both ligand and protein, or the two proteins to be designed for optimal stability for the bounded complex structure. Additionally, we can have the key positions in either or both proteins fixed to reduce the degeneracy of the designed sequence. Finally, we can ask how we can design the sequence of a tethering protein or peptide modulator that binds two proteins simultaneously. Our linear programming method for inverse folding problem may be used in all of these problems. This has significant importance in protein design, protein engineering, and also in searching therapeutic agents.

2. THEORY AND METHOD

The protein inverse folding problem is regarded as an optimization problem. We need to set up the objective function and find the optimal solution.

2.1 Energy fitness function

We firstly study the fitness function in Sun *et al.*'s Grand Canonical Sequence Evolution (GCSE) model (Sun *et al.*, 1995). Given the coordinates of a desired target structure, the optimal sequences are exported. This choice is because of the following advantages of Sun *et al.*'s GCSE model (Sun *et al.*, 1995). (1) Rather than using 20 amino acid types (Ponder and Richards, 1987; Lee and Levitt, 1991; Shakhnovich, 1994), only two monomer types, hydrophobic (H) and polar (P), are considered. The limited searching space allow the study of other possible foldable polymers in addition to proteins could be explored; (2) The amino acid composition is not fixed. It allows a variable composition of sequences and includes those obtained from bead painting and all other possible H/P compositions. It does not converge to homo-polymer sequences because the term of the hydrophobic residue-solvent interaction, encoding the avoidance of contacts between solvent and the hydrophobic monomers, is incorporated into the fitness function; (3) Unlike the lattice model methods (Yue and Dill, 1992; Shakhnovich, 1994; Gutin *et al.*, 1995), Sun *et al.*'s model allows real-space coordinates and can be applied to real molecules. Sun *et al.*'s GCSE model has been applied to many theoretical model studies and has brought important insights on

practical protein sequences research and experiments.

2.1.1 Grand Canonical Sequence Evolution algorithm

Following the GCSE model, for a protein structure from the PDB, we define its natural H/P sequence to be the one obtained by translating the protein's true amino acid sequence into an H/P sequence, according to a designation of each of the 20 amino acids as either hydrophobic or polar. Respectively, hydrophobic (H) = A, C, I, L, M, F, W, Y and V; and polar (P) = R, N, D, E, Q, G, H, K, P, S, and T in the one-letter code of amino acids (Sun *et al.*, 1995).

The geometric representation of the target structure and the fitness function Φ on the set of possible sequences need to be specified, in order to fully define the GCSE model. A simplified geometric representation of such a structure is obtained by constructing a sphere of the appropriate radius at the location of each non-hydrogen backbone atom, and replacing the side chain of each residue with a single "side chain bead," of radius 2 Å, at a distance of 3 Å from the C_α along the C_α-C_β bond vector. For the residue positions occupied by glycine in the target structure, no native C_α-C_β bond vector was available, so positioned the glycine side chain bead at the location of the $C\alpha$ (Kleinberg, 1999). In this way, the residues in the target structure are made "uniform". On the other hand, the fitness function Φ is formulated as follows,

$$\Phi(S) = \alpha \sum_{\substack{i,j \in S_H \\ i<j-2}} g(d_{ij}) + \beta \sum_{i \in S_H} s_i \qquad (1)$$

Here s_i denotes the area of the solvent-accessible contact surface of the side chain for residue i (in Å^2), and d_{ij} denotes the distance between the side chain centers of residues *i* and *j* (in Å). g is a sigmoid function that rewards small distances; in Sun *et al.* (1995), it is defined to be $\frac{1}{1+\exp(d_{ij}-6.5)}$ for $d_{ij} < 6.5\overset{\circ}{A}$ and 0 for $d_{ij} > 6.5\overset{\circ}{A}$. Finally, $\alpha < 0$ and $\beta > 0$ are scaling parameters; they are given default values of $\alpha = -2$ and $\beta = 1/3$ (Kleinberg, 1999). To design a sequence, we must specify which residues in the target structure will be H (hydrophobic), and which will be P (polar); thus, a protein sequence S is a sequence of n symbols, each of which is either H or P. We use S_H to denote the set of numbers *i* such that the i^{th} position in the sequence S is equal to H; we define S_p analogously. Now, the fitness function Φ(S) of a sequence S, with respect to the target structure, is a scoring function motivated by the following (partially conflicting) requirements. The H residues in S would like to have low solvent-accessible surface area; and H residues are expected to be close to one another in space, so as to form a compact hydrophobic core.

2.1.2 Definition of coarser Grand Canonical Sequence Evolution model

We consider the *simplified* definitions of Φ. The contact surface areas $s_i = 1$, if residue *i* is hydrophobic and contacts a solvent site; otherwise, $s_i = 0$. The sum of the second term in Equation (1) is over solvent contacts. One solvent contact is defined as every 3 Å^2 of surface area exposed of a side chain bead, using the default 1.4 Å probe. *g* is assigned to be a *step*

function: $g(d_{ij})$ is equal to 1 if $d_{ij} \leq 6.5\,\mathring{A}$; and $g(d_{ij})$ is equal to 0, otherwise. This simplified definition provides a "coarser" view of the set of sequences for the target structure (Kleinberg, 1999).

The objective function in the sequence design optimization problem is the given structure's energy fitness function. The goal of the GCSE model is to design a sequence whose fitness value Φ is minimized (*i.e.* as negative as possible); we will call such a sequence *optimal*. This corresponds to constructing a sequence with many close-range *H-H* contacts, and very few solvent-exposed *H* residues.

2.2 Formulation based on Graph Theory

With the optimization function defined, searching for the optimal solution is the next step. As there are 2^n possible amino acid sequences in the binary H/P model, an exhaustive search is not possible. Sun *et al.* developed a heuristic method to find sequences of good fitness based on a genetic algorithm. However, their method did not provide any measure of how close the final designed sequences are to the optimal sequence(s). For a given target structure, the problem of designing sequence with lowest energy of the form of Equation (1) has been solved by Kleinberg (1999) using techniques from combinatorial optimization of network flow (Kleinberg, 1999). Kleinberg's transformation of the original GCSE model to a network flow optimization problem for the inverse folding problem allows efficient construction of optimal sequences.

A key observation of Kleinberg (1999) is that the structure of a protein can be represented geometrically by a directed graph G, and any H/P sequence corresponds to a partition of the set of nodes V in G into two sets. We now recall the basic knowledge of Graph Theory.

2.2.1 Basic knowledge of Graph Theory

A *directed graph G* consists of a pair of sets: *V* (the *vertices*) and *E* (the *edges*). Each edge $e \in E$ is an ordered pair of vertices $e = (u, v)$; *u* is called as the *tail* of *e* and *v* the *head*. It is also assumed that each edge has a given *capacity* c_e, which is a positive number. Let *s* and *t* be two vertices of *G*. An *s-t cut* in *G* is a partition of *V* into two sets, *X* and *Y*, so that $s \in X$ and $t \in Y$; such a cut is denoted by the pair (*X, Y*). An edge is defined as "*crosses"* a cut (*X, Y*), if it has its tail in *X* and its head in *Y*. The *capacity* of a cut (*X, Y*) is equal to the sum of the capacities of all edges that cross (*X, Y*); it is denoted as *c*(*X, Y*). The *minimum s-t cut problem* asks, for a given graph *G* and vertices *s* and *t*, to find an *s-t* cut (*X, Y*) of minimum capacity (Ahuja *et al.*, 1993 and Cormen et al., 1990). The maximum flow problem is to find a feasible flow through a single-source, single-sink flow network that is maximum. The ***max-flow min-cut theorem*** states that in a flow network, the maximum amount of flow passing from the source to the sink is equal to the minimum capacity which when removed in a specific way from the network causes the situation that no flow can pass from the source to the sink. The minimum s-t cut / maximum flow problem can be solved for an arbitrary directed graph with *n* vertices and *m* edges by algorithms of running times bounded by $O(mn \log n)$ (Goldberg and Tarjan, 1988; Sleator and Tarjan,

1983), and efficient implementations exist for some of these algorithms (Cherkassky and Goldberg, 1995).

2.2.2 Establishment of the flow network optimization

Now, let Φ be the fitness function corresponding to a given target structure of length n. Recall that the target structure determines inter-residue distances d_{ij} and solvent-exposed surface areas s_i; and that Φ is defined via a function g and parameters $\alpha < 0$ and $\beta > 0$. Let B denote the quantity $\sum_{i<j-2} |\alpha| g(d_{ij})$. The following graph G based on Φ is defined by Kleinberg (Kleinberg, 1999). The vertex set V of G consists of s, t, a vertex v_i for each of the residue positions $i = 1, 2, \ldots, n$ in the target structure, and a vertex u_{ij} for each pair of residue positions i, j for which $i < j - 2$ and $g(d_{ij}) > 0$. The edge set E of G consists of an edge (s, u_{ij}) for each vertex u_{ij}, an edge (v_i, t) for each vertex v_i which has a non-zero solvent-exposed contact surface area s_i, and edges (u_{ij}, v_i) and (u_{ij}, v_j) for each vertex u_{ij} (Kleinberg, 1999). See Figure 1 for an example of the directed graph constructed by this procedure from an artificial 9-residue structure. A capacity is further assigned to each edge e in the graph. The edges of (s, u_{ij}) type are assigned a capacity of $|\alpha|$, the parameter for non-bonded contact interaction; the edges of (v_i, t) type are assigned a capacity of β, the solution parameter; and all edges connecting contact node to corresponding residue nodes (u_{ij}, v_i) and (u_{ij}, v_j) are assigned a capacity of B+1, which is a large value, $B + 1 > \sum_{(s,u_{ij})} |\alpha|$.

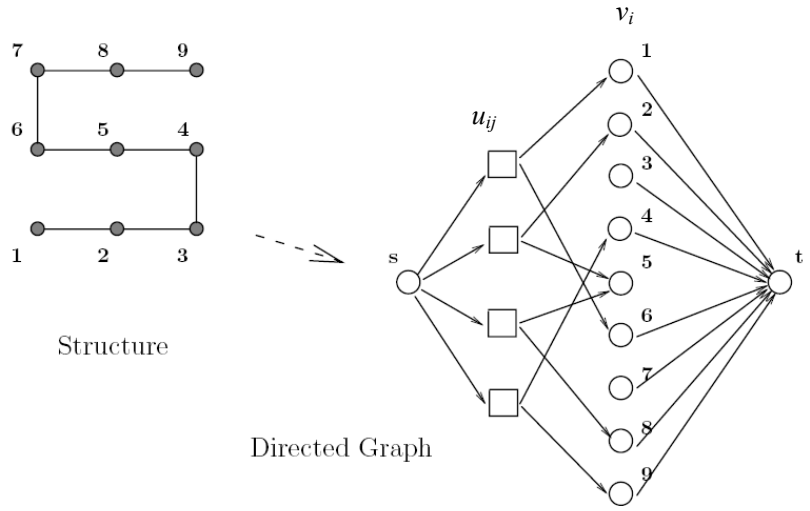

Figure 1. A small example of the construction of a directed graph from a target structure with four possible contacts (1-6, 2-5, 5-8, 4-9) (Kleinberg, 1999).

A set X of vertices is ***closed*** if (i) X contains s but not t, and (ii) for each $u_{ij} \in V$, X contains u_{ij} if and only if it contains both v_i and v_j. Then the fact below can be proved.

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

Proof. First note that G has an s-t of capacity B; in particular, consider the cut $(\{s\}, V - \{s\})$. Now, consider a minimum s-t cut (X, Y) in G. Suppose X contains a vertex u_{ij} but not the vertex v_i (the case of v_j is the same). Then the edge (u_{ij}, v_i) crosses (X, Y) and it capacity $B + 1$; this

contradicts the assumption that (X, Y) is a minimum cut. On the other hand, suppose that X contains some pair of vertices v_i and v_j, but not the vertex u_{ij}. Then the s-t cut $(X \bigcup \{u_{ij}\}, Y - \{u_{ij}\})$ would have smaller capacity than (X, Y), again a contradiction.

■ (Kleinberg, 1999).

For an n-symbol H/P sequence S, Z is denoted to the set of all vertices v_i for which position i in S is labeled H. By the above "*closed*" fact, $X(S)$ is denoted to the *closed* set consisting of s, the vertices in Z, and all vertices u_{ij} for which v_i, $v_j \in Z$. Conversely, if X is a closed set, $S(X)$ is denoted to the H/P sequence in which position i is labeled H if v_i belongs to X, and is labeled P if v_i does not belong to X. From these constructions, a one-to-one correspondence between n-symbol H/P sequences and closed sets in G exists (Kleinberg, 1999). Consequently, a crucial fact about G is stated.

Let X be a closed set and S(X) the corresponding H/P sequence. Then the capacity of the s-t cut (X, V − X) is equal to B +Φ(S(X)).

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

$$c(X, V-X) = \sum_{\substack{u_{ij}\in V \\ \{v_i, v_j\}\not\subset X}} |\alpha| g(d_{ij}) + \sum_{v_i \in X} \beta s_i$$

$$= B - \sum_{\substack{u_{ij}\in V \\ \{v_i, v_j\}\subseteq X}} |\alpha| g(d_{ij}) + \sum_{v_i \in X} \beta s_i$$

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

$$= B + \Phi[S(X)].$$ ■ (Kleinberg, 1999)

Thus the fitness of an H/P sequence for the target structure differs from capacity of the corresponding cut in *G* simply by the fixed additive constant *B*. Consequently, if (*X, Y*) is a minimum capacity *s*-*t* cut in *G*, then *S*(*X*) is an optimal sequence — so to find an optimal sequence for the target structure, what is only needed is to construct the graph *G* and compute a minimum capacity *s*-*t* cut in it (Kleinberg, 1999).

Through the above discussion and proof, we can draw the conclusion, to design an optimal sequence under the energy fitness function of Equation (1), we need to mark each residue either as member of the hydrophobic H set, or the polar set P. Any given sequence of the same chain length can also be mapped to graph G by labeling of the appropriate nodes. We assign node v_i to be in set H if residue v_i is an H residue, to be in set P if it is a P residue. We also assign node $s \in H$ and $t \in P$. Pair node u_{ij} is assigned to be in set H if and only if both of the corresponding residues v_i and v_j are H residues. Pair node u_{ij} is assigned to set P if either v_i and v_j is P. Conversely, given a graph G representing the geometry of the molecular structure, we can

label the residue nodes v_i as either H or P, and label pair node u_{ij} as H if and only if both residues v_i and v_j are H residues. Node u_{ij} otherwise is labeled as P. This labeling provides a mapping from the labeled graph G to an H/P sequence. Pair node u_{ij} can be a member of set H if and only if both of the corresponding residue nodes v_i and v_j are in H. Clearly, there is one-to-one correspondence between an H/P sequence and a partition of the nodes in the directed graph that corresponds to a closed set H.

To ensure that H is a closed set so the labeled graph corresponds to a sequence, we cannot allow any cut on edges (u_{ij}, v_i) or (u_{ij}, v_j). Namely, pair node u_{ij} and residue nodes {v_i, v_j} must all have the same label. A cut on these edges will never occur because all such edges have been assigned with a large capacity B+1, here B= $\sum_{i<j-2} |\alpha| g(d_{ij})$, then B+1 is always greater than $\sum_{i<j-2} |\alpha| g(d_{ij})$: the trivial cut of H = {s} and P = {all other nodes} will already have a lower cost than B+1. Therefore, the minimum cut can only happen to the edges (s, u_{ij}) or (v_i, t).

2.3 Linear programming formulation for inverse folding problem

Kleinberg converted the protein inverse folding problem to the minimum cut problem in a flow network. In this study, we provide a novel simple fitness function for the inverse folding problem by reformulating Kleinberg's flow network algorithm into a linear programming problem, which provides a general model for studying a variety of problems in protein design. The key observation is that the max-flow and min-cut theorem can be applied and the s-t cut problem can

be reformulated as the dual max-flow linear programming problem.

2.3.1 Transformation to linear programming from flow network

Recall the ***max-flow, min-cut theorem***: the capacity of the smallest cut is exactly equal to the maximum flow that can be pushed from s to t. Given a graph G ($G = (V, E), s, t, c$) (here s is start node; t is sink node; c is the capacity on each edge, for example, the capacity on edge (u, v) is denoted as c(u, v)), the problem of finding the maximum flow in the network can be formulated as a linear program by simply writing down the definition of feasible flow. We have one variable $f(u,v)$ for every edge $(u,v) \in E$ of the network, and the problem is:

$$\text{Maximize} \quad \sum_{v:(s,v)\in E} f(s,v) \tag{2}$$

$$\text{Subject to} \quad \sum_{u:(u,v)\in E} f(u,v) = \sum_{w:(v,w)\in E} f(v,w) \qquad \forall v \in V - \{s,t\}$$

$$f(u,v) \le c(u,v) \qquad \forall (u,v) \in E$$

$$f(u,v) \ge 0 \qquad \forall (u,v) \in E$$

Let us see what the dual of (2) looks like. The dual of (2) has one variable for each vertex *v* (except *s* and *t*), which we shall call $\pi(v)$, corresponding to the conservation constraints, and one variable for each edge, which we shall call $\gamma(u,v)$, corresponding to the capacity constraints.

$$\text{Minimize} \quad \sum_{(u,v)\in E} c(u,v)\gamma(u,v) \tag{3}$$

$$\text{Subject to} \quad \pi(u) + \gamma(s,u) \geq 1 \qquad \forall v : (s,u) \in E$$

$$\pi(v) - \pi(u) + \gamma(u,v) \geq 0 \quad \forall (u,v) \in E, u \neq s, v \neq t$$

$$-\pi(v) + \gamma(v,t) \geq 0 \qquad \forall u : (v,t) \in E$$

This linear programming describes the min-cut problem. To see why, suppose that the $\pi(u)$ variable is meant to be 1 if u is in the cut with s, and 0 otherwise, and similarly for *v*. Each of the γ variables is to be 1 if the corresponding edge contributes to the cut capacity; and 0 otherwise. Then the constraints make sure that theses variables behave exactly as they should. For example, the second constraint states that if u is with s and v is not [this is the only case in which the sum $-\pi(u) + \pi(v)$ becomes -1], then (u,v) must contribute to the cut. Although the π and γ's are free to take values larger than one, they will be "slammed" by the minimization down to 1 or 0.

By the ***max-flow min-cut theorem***, the two Linear Programming's Primal, (2), and Dual, (3), above have the same optimum. In fact, this is true for general dual LP's. This is the ***duality theorem***, which can be stated as follows:

If an LP has a bounded optimum, then so does its dual, and the two optimal values coincide.

Until now, Kleinberg's flow network problem has been transformed to a simple linear programming problem.

2.3.2 Our proposed linear programming fitness function

The linear optimization problem (3) in Chapter 2.3.1 is min-cut problem as described. When $\gamma(u,v)$=1, the min-cut happens on the edge (*u, v*); when $\gamma(u,v)$=0, the min-cut does not happen on the edge (*u, v*). And when $\gamma(u,v)$=1, then $\pi(u)$=1 and $\pi(v)$=0, i.e. vertex u belongs to the set of s and vertex v belongs to the set of t; when $\gamma(u,v)$=0, then $\pi(u)$ and $\pi(v)$ both equal 1 or $\pi(u)$ and $\pi(v)$ both equl 0, i.e. vertex u and v are both in set of s or they are both in set of t. Besides, $\pi(s)=1$ and $\pi(t)=0$

Furthermore, we can apply this linear programming to our H/P sequence model. A minimum s-t cut can be found by minimizing the sum of cost functions of all edges in the graph, $\sum_{(u,v)\in E} c(u,v)\gamma(u,v)$, namely, by selecting an optimal subset of edges. The edges can be either residue edge, which corresponds to the solution term, or pair edge, which corresponds to contact energy. All edges have fixed capacity associated with them, whose values depend on the edge type. The solution of this linear programming leads to a closet set H and an optimal designed sequence is obtained. As the above discussion, $\gamma(u,v)$ is the "cut" indicator, so we have $\gamma(u,v)=\begin{cases}1, u\in H, and v\in P\\ 0, otherwise.\end{cases}$; $\pi(u)$ and $\pi(v)$ are the "label" indicator: $\pi(u)=\begin{cases}1, u\in H\\ 0, u\in P.\end{cases}$,

so does $\pi(v)$.

Combining the conclusion in Chapter 2.2.2, the minimum cut can only happen to the edges (s,u_{ij}) or (v_i,t), the min-cut linear programming problem (3) can be rewritten as

$$\text{Minimize} \quad \sum_{(s,u)\in E} c(s,u)\gamma(s,u) + \sum_{(v,t)\in E} c(v,t)\gamma(v,t) \tag{4}$$

$$\text{Subject to} \quad \pi(u) + \gamma(s,u) \geq 1 \qquad \forall u : (s,u) \in E$$

$$-\pi(v) + \gamma(v,t) \geq 0 \qquad \forall v : (v,t) \in E$$

We have $c(s,u) = |\alpha|$, and $c(v,t) = \beta$ in Chapter 2.2.2, then the linear optimization problem (4) becomes

$$\text{Minimize} \quad |\alpha| \sum_{(s,u)\in E} \gamma(s,u) + \beta \sum_{(v,t)\in E} \gamma(v,t) \tag{5}$$

$$\text{Subject to} \quad \pi(u) + \gamma(s,u) \geq 1 \qquad \forall u : (s,u) \in E$$

$$-\pi(v) + \gamma(v,t) \geq 0 \qquad \forall v : (v,t) \in E$$

$$\gamma(s,u), \gamma(v,t), \pi(u), \pi(v) = 0 or 1$$

The linear optimization problem (5) is our proposed fitness function for the protein inverse folding problem. The values of α and β is assigned -2 and 1/3 in Kleinberg's flow network algorithm. In our following experiments, we also use -2, 1/3 as their values.

3. EXPERIMENTS AND RESULTS

3.1 Protein structures for experiments

We implemented the above flow network algorithm and our network linear programming model on the 23 PDB structures drawn from the Protein Data Bank considered by Sun *et al.* (1995). See Table 1.

An advantage of testing the algorithm on real proteins is that we can compare the sequences we design to the true sequence of the proteins, as in Sun *et al.* (1995) and Kleinberg (1999); this is a way to assess the biological relevance of the GCSE model, Kleinberg's flow network algorithm, and our network linear programming model. For a protein structure from the PDB, let us define its natural H/P sequence to be the one obtained by translating the protein's true amino acid sequence into an H/P sequence, according to a designation of each of the 20 amino acids as either hydrophobic or polar. As Sun *et al.* classified hydrophobic (H) = A, C, I, L, M, F, W, Y and V; and polar (P) = R, N, D, E, Q, G, H, K, P, S, and T in the one-letter code of amino acids (Sun *et al.*, 1995). Since the fitness function Φ associated with the model is designed only to approximate the factors favoring the natural sequence, the natural sequence is likely to be sub-optimal when scored according to Φ with respect to its structure; correspondingly, the optimal sequence under Φ may differ non-trivially from the natural sequence.

Table 1. 23 Protein structures for experiments

Protein(PDBcode)	Sequence length	Number of Hydrophobic (H)	Number of Polar (P)	Ratio of H to P residues
1aaj	105	48	57	0.8421
1aba	87	35	52	0.6731
1aps	98	34	64	0.5313
1arr(mon)	53	20	33	0.6061
1arr(dim)	106	40	66	0.6061
1bba	36	15	21	0.7143
1bbl	37	14	23	0.6087
1bov	69	26	43	0.6047
1brq	174	74	100	0.7400
1cis	66	30	36	0.8333
1cmb(mon)	104	38	66	0.5758
1cmb(dim)	208	76	132	0.5758
1hel	129	54	75	0.7200
1ifb	131	50	81	0.6173
1kba(mon)	66	26	40	0.6500
1kba(dim)	132	52	80	0.6500
2gb1	56	21	35	0.6000
2hpr	87	37	50	0.7400
2il8	71	28	43	0.6512
256b	106	41	65	0.6308
3cln	143	52	91	0.5714
3rn3	124	47	77	0.6104
3trx	105	47	58	0.8103

Note. “mon” means monomer and “dim” means dimer. They refer to sequence design calculations performed using the monomer structure along as the target and the full dimmer structure as the target, respectively.

3.2 Simulation results of flow network algorithm

We simulated Kleinberg's flow network algorithm, repeated the sequence design process and reproduced the optimal sequence for the flow network optimization problem. The aim of this work is to verify and assess the ability of Kleinberg's flow network algorithm to design protein sequences from given protein structures. We tested the implementation of the Kleinberg's flow network algorithm on the above 23 PDB structures, in the part of searching the minimum cut / maximum flow in a directed graph, making use of the Cherkassky and Goldberg's push-relabel method for the maximum flow/minimum cut problems (Cherkassky and Goldberg, 1995). In the Table 2, we presented the simulation results of Kleinberg's flow network algorithm, computed the percentage agreement between the natural and optimal sequences for the 23 PDB structures.

Table 2. Simulation results of Kleinberg network flow algorithm for 23 PDB structures

Protein(PDBcode)	Sequence length	Flow network Alg.	Running time
1aaj	105	73.33	0.005000
1aba	87	74.71	0.003000
1aps	98	77.55	0.003000
1arr(mon)	53	62.26	0.001999
1arr(dim)	106	70.75	0.003000
1bba	36	58.33	0.000000
1bbl	37	56.76	0.001999
1bov	69	73.91	0.002000
1brq	174	74.14	0.010000
1cis	66	68.18	0.002000
1cmb(mon)	104	63.46	0.003000
1cmb(dim)	208	74.04	0.016997
1hel	129	79.07	0.007999
1ifb	131	68.7	0.006999
1kba(mon)	66	68.18	0.002000
1kba(dim)	132	77.27	0.005999
2gb1	56	78.57	0.001000
2hpr	87	78.16	0.003000
2il8	71	71.83	0.002997
256b	106	77.36	0.003000
3cln	143	72.03	0.007000
3rn3	124	69.35	0.006999
3trx	105	77.14	0.003999
Length-Weighted average		72.70	

Note. In the column under “Flow network algorithm”, the values are the percentage agreement between the optimal and natural sequences for the 23 structures from Sun *et al.* (1995).

The running time of the algorithm (in CPU seconds) on a cluster Intel(R) Core(TM)2 Duo CPU are depicted in Figure 2; for the structures of lengths 36—208 considered in Sun *et al.*(1995), the running time ranged from 0.00000001 (the shortest sequence 1bba in the length of 36) to 0.016997 (the longest sequence 1cmb(dim) in the length of 208) seconds. And from Figure 2, we can see that the running time is kind of related with the sequences' length.

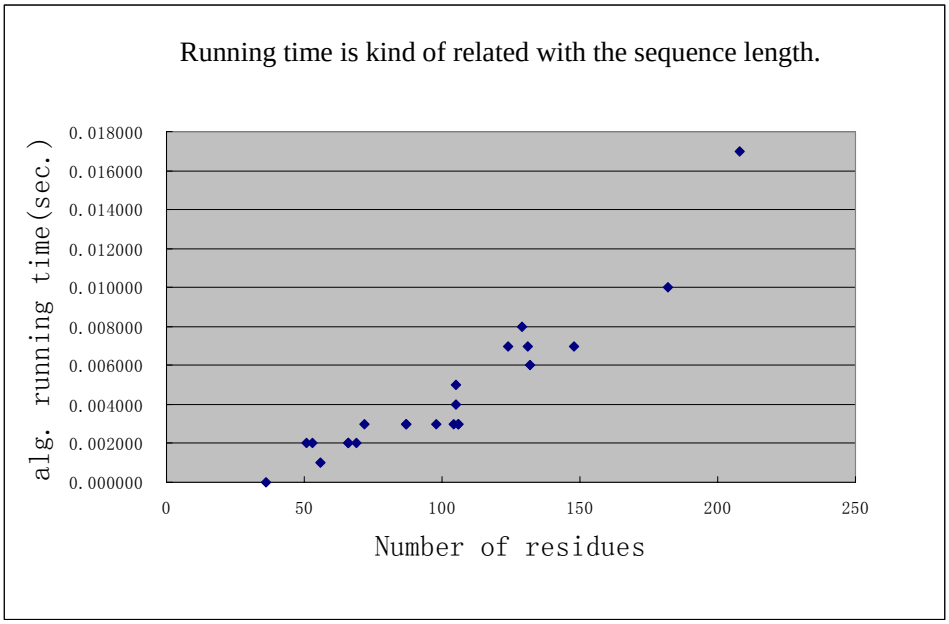

Figure 2. Kleinberg's network flow algorithm's running time. The running time is kind of related with the sequence length. Generally, with the increasing of sequence length, the running time becomes longer.

3.3 Results of linear programming model

Base on the statement in Chapter 2.3.2, we reformulate the protein inverse folding problem to a linear programming. Furthermore, we notice that the linear programming problem is also belonged to binary integer linear programming problem, because all the values of $\gamma(u,v)$, $\pi(u)$ and $\pi(v)$ are equal to 0 or 1. Then we solve this binary integer linear programming problem by the existing 0-1 linear programming algorithms (Wolsey, 1998; Nemhauser *et al*., 1988; Hillier *et al*., 2001), and realized it on Matlab R2011a. Results of the application on the above 23 PDB structures are presented in Table 3.

Table 3. Results of our network linear programming model for 23 PDB structures

Protein(PDBcode)	Sequence length	Linear prog.	Running time
1aaj	105	72.86	0.646189
1aba	87	74.51	0.527784
1aps	98	77.62	0.570127
1arr(mon)	53	62.31	0.353962
1arr(dim)	106	71.04	0.528745
1bba	36	58.33	0.160894
1bbl	37	56.92	0.214563
1bov	69	74.30	0.401179
1brq	174	72.59	1.796843
1cis	66	68.33	0.358479
1cmb(mon)	104	62.76	0.561255
1cmb(dim)	208	73.98	2.897239
1hel	129	78.91	1.100563
1ifb	131	68.87	1.367267
1kba(mon)	66	68.01	0.397716
1kba(dim)	132	77.29	1.474798
2gb1	56	79.02	0.313341
2hpr	87	77.69	0.358392
2il8	71	71.86	0.323597
256b	106	77.36	0.723121
3cln	143	69.59	2.089519
3rn3	124	68.52	1.012549
3trx	105	77.23	0.633199
Length-Weighted average		72.15	

Note. In the column under “Linear prog.”, the values are the percentage agreement between the optimal and natural sequences for the 23 structures from our network linear programming.

The running time of our linear programming algorithm (in CPU seconds) on a personal IBM laptop are depicted in Figure 3; for the structures of lengths 36—208 considered in Sun *et al.*(1995), the running time ranged from 0.160894 (the shortest sequence 1bba in the length of 36) to 2.897239 (the longest sequence 1cmb(dim) in the length of 208) seconds. And from Figure 3, we can see that the linear programming running time is also kind of related with the sequences' length.

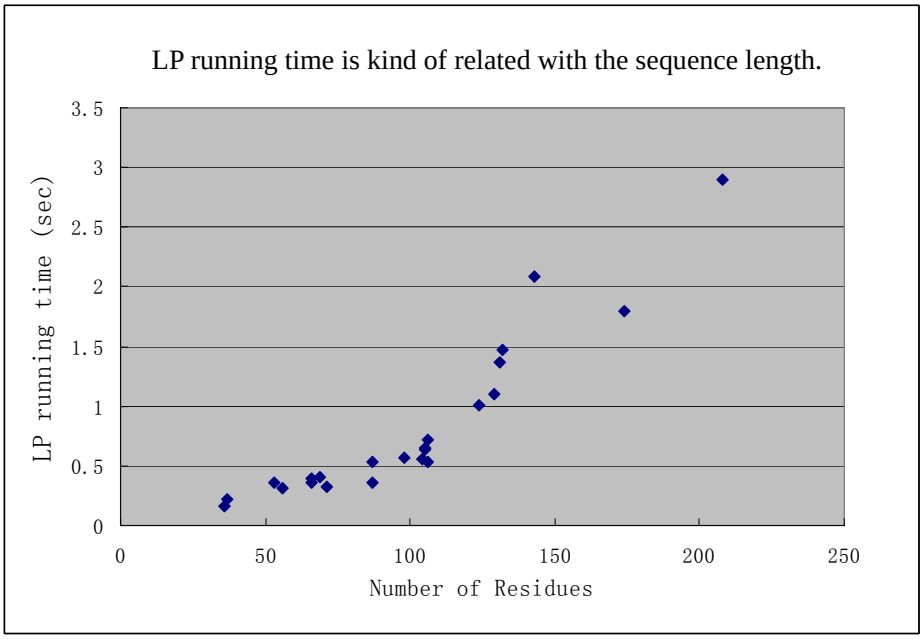

Figure 3. Our linear programming method running time. It is also kind of related with the sequence length. Generally, with the increasing of sequence length, the running time becomes longer

We also compare the results from Kleinberg's flow network algorithm and our network linear programming model, and quote the numbers of Sun *et al*. for the sake of comparison. From Table 4, it is interesting to note that the percentages agreement between optimal and natural sequences from the flow network algorithm and our network linear programming are almost equal. The agreement percentages change with the markedly different structures, as we move from Sun *et al.*'s designed sequences to flow network algorithm and our network linear programming. For certain protein, the agreement percentage jumps considerably when the sequences are designed by Kleinberg's flow network algorithm and our network linear programming model, instead of the Sun *et al.*'s model. For example, the agreement percentage for 1aaj increases from Sun *et al*.'s value of 66% to 73% for flow network and our linear optimal algorithms. On the other hand, certain structures, such as 1ifb, show significantly less agreement with the natural sequence when solved by flow network algorithm and our linear programming algorithm, from 76% to 69%. So does 3rn3, from 81% to 69%. The length-weighted average of percentage agreement between optimal and natural sequences is 72.1% for Sun *et al.*, 72.70% for Kleinberg's flow network algorithm, 72.15% for our network linear programming method, respectively. See Figure 4.

Table 4. Results of different algorithms for 23 PDB structures

Protein(PDBcode)	Sequence length	Sun *et al.*'s alg.	Flow network alg.	Our linear Prog.
1aaj	105	66	73.33	72.86
1aba	87	81	74.71	74.51
1aps	98	72	77.55	77.62
1arr(mon)	53	62	62.26	62.31
1arr(dim)	106	73	70.75	71.04
1bba	36	58	58.33	58.33
1bbl	37	68	56.76	56.92
1bov	69	74	73.91	74.30
1brq	174	68	74.14	72.59
1cis	66	64	68.18	68.33
1cmb(mon)	104	62	63.46	62.76
1cmb(dim)	208	70	74.04	73.98
1hel	129	74	79.07	78.91
1ifb	131	76	68.7	68.87
1kba(mon)	66	72	68.18	68.01
1kba(dim)	132	73	77.27	77.29
2gb1	56	80	78.57	79.02
2hpr	87	78	78.16	77.69
2il8	71	77	71.83	71.86
256b	106	81	77.36	77.36
3cln	143	62	72.03	69.59
3rn3	124	81	69.35	68.52
3trx	105	80	77.14	77.23
Length-Weighted average		72.1	72.70	72.15

Note. The agreement percentages from the flow network algorithm and our linear programming are almost equal. The markedly varied way in which the percentages of overlap changes, for different structures, as we move from Sun *et al.*'s designed sequences to flow network algorithm and our network linear programming.

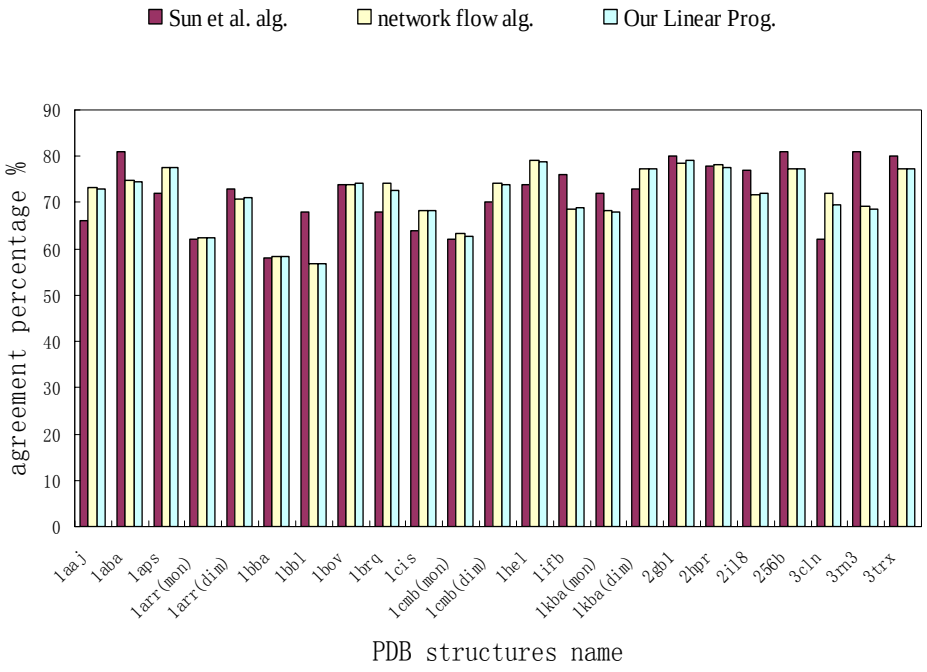

Figure 4. The agreement percentage comparison of three different methods for 23 PDB structures. The percentages agreement between optimal and natural sequences from the flow network algorithm and our network linear programming are almost equal. The agreement percentages change with the markedly varied way, for different structures, as we move from Sun *et al.*'s designed sequences to flow network algorithm and our network linear programming. For certain of the proteins, the agreement percentage jumps considerably when the sequences are designed by Kleinberg's flow network algorithm and our network linear programming model, instead of the Sun *et al.*'s model. For example, the agreement percentage for 1aaj increases from Sun *et al.*'s value of 66% to 73% for flow network and our linear optimal algorithms. On the other hand, certain structures, such as 1ifb, show significantly less agreement with the natural sequence when solved by flow network algorithm and our linear programming algorithm, from 76% to 69%. So does 3rn3, from 81% to 69%. The length-weighted average of percentage agreement between optimal and natural sequences is 72.1% for Sun *et al.*, 72.70% for Kleinberg's flow network algorithm, 72.15% for our network linear programming method, respectively.

3.4 Inverse folding with specified residue composition

Our linear programming formulation allows the study of additional problems in sequence design. An important consideration is that the designed sequence should have a large energy difference with random sequence. It is thought that random sequences are not correlated, and their energy depends only on the bulk property, namely, the residue composition, but not on the specific sequences. Designing optimal sequences with a pre-specified fixed composition will more likely result in sequences with large energy gaps. For a sequence with N residues, if there are n residues to be H in the sequence, we can design such a sequence with optimal energy by adding the following constraint to the linear programming problem:

$\sum_{v \in V_R} \pi(v) = n$, where $V_R = \{residuenodes\} \subset V$, i.e., V_R is the set of residue nodes.

Therefore, the network linear programming optimal function (5) in Chapter 2.3.2 could be

$$\text{Minimize} \quad |\alpha| \sum_{(s,u)\in E} \gamma(s,u) + \beta \sum_{(v,t)\in E} \gamma(v,t) \qquad (6)$$

$$\text{Subject to} \quad \pi(u) + \gamma(s,u) \geq 1 \qquad \forall u : (s,u) \in E$$

$$-\pi(v) + \gamma(v,t) \geq 0 \qquad \forall v : (v,t) \in E$$

$$\gamma(s,u), \gamma(v,t), \pi(u), \pi(v) = 0 or 1$$

$$\sum_{v \in V_R} \pi(v) = n, \text{ where } V_R = \{residuenodes\} \subset V$$

We now study the additional problems in sequence design—inverse folding problem with specified residue composition. Here we attempt to construct a sequence in which the number of residue H, *n*, is fixed. Following the existing research, the ratio of H and P residues in a sequence

is roughly 2/3, matching the relative frequencies of the corresponding amino acids in naturally occurring polypeptide sequences (Creighton, 1993). Given the length of a sequence, we can calculate the number of H residues in it. Based on the length of 23 PDB structures in Table 1, we obtain the values of n for each sequence presented under the column of "Calculated number of H" in Table 5. In Table 5 we compute the percentage agreement between the natural and optimal sequences for the 23 PDB structures with the optimal function (6), presented in the column under "LP with fixed HP composition". We also reproduce the numbers of network linear programming for the sake of comparison. It is interesting to find that for different PDB structures, the percentages of the agreement vary as we add $\sum_{v \in V_R} \pi(v) = n$ in the constraint expression. For some of the proteins, the agreement percentage jumps considerably when the sequences are designed by the method of network linear programming with the fixed HP composition. For example, the agreement percentage of 1bbl increased from linear programming's value of 56.92% to 64.86% for linear programming with the fixed HP composition constraint; the agreement percentage of 1kba monomer increases from linear programming's value of 68.01% to 75.76% for linear programming with the fixed HP composition constraint. On the other hand, certain structures show significantly less agreement with the natural sequence when solved to the linear programming's optimality with the fixed HP composition constraint, such as 1aba, the agreement percentage of 1aba decreases from linear programming's value of 74.51% to 70.12% for linear programming with the fixed HP composition constraint. Although in the above three cases, the calculated numbers of hydrophobic residues are equal to the practical numbers of hydrophobic residues in sequences, the linear programming with fixed HP composition constraint can not

make sure the increase of percentage agreement between optimal and natural sequences. And we also notice that for the sequences whose calculated numbers of hydrophobic residues are not equal to the practical numbers of hydrophobic residues, the agreement percentage has no remarkable increase, and furthermore, for most cases in this situation, the agreement percentage decreases, as we move from linear programming to linear programming with the fixed HP composition constraint. See Figure 5.

Table 5. Results of Linear Programming with fixed HP composition for 23 PDB structures.

Protein (PDBcode)	Sequence length	Real Number of H	Calculated number of H	LP	LP with fixed HP composition
1aaj	105	48	42	72.86	69.52
1aba	87	35	35	74.51	70.12
1aps	98	34	39	77.62	74.49
1arr(mon)	53	20	21	62.31	62.26
1arr(dim)	106	40	42	71.04	72.64
1bba	36	15	14	58.33	58.33
1bbl	37	14	15	56.92	64.86
1bov	69	26	28	74.30	71.01
1brq	174	74	70	72.59	71.84
1cis	66	30	26	68.33	62.12
1cmb(mon)	104	38	42	62.76	62.50
1cmb(dim)	208	76	83	73.98	72.60
1hel	129	54	52	78.91	77.52
1ifb	131	50	52	68.87	69.47
1kba(mon)	66	26	26	68.01	75.76
1kba(dim)	132	52	53	77.29	76.52
2gb1	56	21	22	79.02	75.00
2hpr	87	37	35	77.69	77.01
2il8	71	28	28	71.86	78.87
256b	106	41	42	77.36	76.42
3cln	143	52	57	69.59	70.63
3rn3	124	47	50	68.52	66.94
3trx	105	47	42	77.23	76.19
Length-Weighted average				72.15	71.74

Note. In the column under “Real number of H”, the values are the number of Hydrophobic residues in practice; the values in the column “Calculated number of H” are the number of Hydrophobic residues calculated by the sequence length multiplying 2/5. In the columns under “LP” and “LP with fixed HP composition”, the values are the percentage agreement between the optimal and natural sequences with the method network linear programming and network linear programming with the fixed HP composition for the 23 PDB structures, respectively.

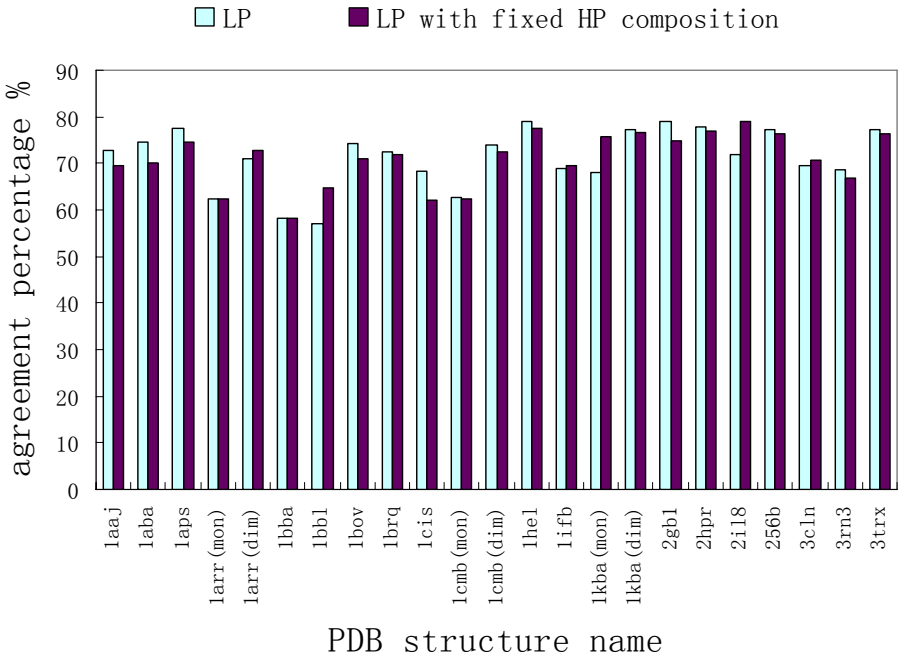

Figure 5. The agreement percentage comparison between network linear programming and network linear programming with the fixed HP composition. For some of the proteins, the agreement percentage jumps considerably when the sequence is designed by the method of network linear programming with the fixed HP composition. For example, the agreement percentage of 1bbl increases from linear programming's value of 56.92% to 64.86% for linear programming with the fixed HP composition constraint; the agreement percentage of 1kba monomer increases from linear programming's value of 68.01% to 75.76% for linear programming with the fixed HP composition constraint. On the other hand, certain structures show significantly less agreement with the natural sequence when solved to the linear programming's optimality with the fixed HP composition constraint, such as 1aba, the agreement percentage of 1aba decreases from linear programming's value of 74.51% to 70.12% for linear programming with the fixed HP composition constraint. Although in the above three cases, the calculated numbers of hydrophobic residues are equal to the practical numbers of hydrophobic residues in sequences, the linear programming with fixed HP composition constraint can not make sure the increase of agreement percentage. And we also notice that for the sequences whose calculated numbers of hydrophobic residues are not equal to the practical numbers of hydrophobic residues, the agreement percentage has no remarkable increase, and furthermore, for most cases in this situation, the agreement percentage decreases, as we move from linear programming to linear programming with the fixed HP composition constraint.

4. DISCUSSION AND CONCLUSION

In this study, we simulate and verify the flow network algorithm, and resolve the protein inverse folding problem by providing a novel linear programming method to construct optimal sequences. We illustrate the effectiveness of our method through the application on 23 structures drawn from the Protein Data Bank. Furthermore, we also consider the extensions of our network linear programming method with specified residue composition in a sequence, as a way to overcome the limitations that a sharp imbalance in the ratio of H to P residues prevents designed sequences from having a high degree of agreement with the natural sequence in most cases.

4.1 Some concern about our linear programming

For linear programming problem, geometrically, the linear constraints define the feasible region, which is a convex polyhedron. A linear function is a convex function, which implies that every local minimum is a global minimum; similarly, a linear function is a concave function, which implies that every local maximum is a global maximum. Optimal solution need not exist, for two reasons. First, if two constraints are inconsistent, then no feasible solution exists: For instance, the constraints $x \geq 2$ and $x \leq 1$ cannot be satisfied jointly; Second, when the polytope is unbounded in the direction of the gradient of the objective function (where the gradient of the

objective function is the vector of the coefficients of the objective function), then no optimal value is attained. Then will the infeasible situation happen to our network linear programming? To answer this question, we need to recall the process of our network linear programming's formulation.

Our network linear optimization function is strictly dual problem of the primal optimization problem (2) (discussed in Chapter 2.3.1) built up based on the physical description of maximum flow in a directed graph, which is also the minimum cut optimization problem in a directed graph. Its function is depicted in Chapter 2.3.1 as follows,

$$\text{Minimize} \quad \sum_{(u,v)\in E} c(u,v)\gamma(u,v) \tag{3}$$

$$\begin{aligned} \text{Subject to} \quad & \pi(u) + \gamma(s,u) \geq 1 && \forall v : (s,u) \in E \\ & \pi(v) - \pi(u) + \gamma(u,v) \geq 0 && \forall (u,v) \in E, u \neq s, v \neq t \\ & -\pi(v) + \gamma(v,t) \geq 0 && \forall u : (v,t) \in E \end{aligned}$$

As discussed in Chapter 2.3.2, we always have one of the following three situations,

$$\begin{cases} \gamma(u,v) = 1 \\ \pi(u) = 1 \\ \pi(v) = 0 \end{cases}, \text{ or } \begin{cases} \gamma(u,v) = 0 \\ \pi(u) = 1 \\ \pi(v) = 1 \end{cases}, \text{ or } \begin{cases} \gamma(u,v) = 0 \\ \pi(u) = 0 \\ \pi(v) = 0 \end{cases}$$

We can see that the constraint functions are satisfied in whichever situation of the above, so the constraints are totally satisfied jointly. And the polytope is bounded. Therefore, comparing with the above two reasons that no feasible solution exist, we conclude that our network linear programming is feasible.

On the other hand, simply understand that our network linear programming describes the minimum cut problem in a given directed graph with a capacity on each edge according to the Graph Theory. The minimum cut always exists in a given graph, then the optimal solution for the minimum cut optimization problem can always be found out, so our network linear programming is always feasible.

4.2 Future work

4.2.1 Parameter selection in our fitness function

Our linear programming fitness function is defied in Chapter 2.3.2 as follows,

$$\text{Minimize} \quad |\alpha| \sum_{(s,u)\in E} \gamma(s,u) + \beta \sum_{(v,t)\in E} \gamma(v,t) \tag{5}$$

$$\text{Subject to} \quad \pi(u) + \gamma(s,u) \geq 1 \qquad \forall u : (s,u) \in E$$

$$-\pi(v) + \gamma(v,t) \geq 0 \qquad \forall v : (v,t) \in E$$

$$\gamma(s,u), \gamma(v,t), \pi(u), \pi(v) = 0 or 1$$

In our experiment on 23 PDB structures in Chapter 3.3, the parameter α is set to -2, and β is set to 1/3. These two parameters α and β can be tracked from Sun *et al*.'s GCSE model. As we known that the relative values of α and β in the GCSE fitness function control the relative proportions of H and P residues in an optimal sequence. Qualitatively, if β is fixed, there is an increasing reward for hydrophobic contacts, with the decreasing of α's value; if α is fixed, there is an increasing penalty for solvent-exposed hydrophobic residues, with the

increasing of β's value. At more concrete level, the minimum number of H residues in an optimal sequence increases monotonically as β is held fixed and α is made increasingly negative (Kleinberg, 1999). Therefore, the adjustment of parameters α and β will bring the different ratio of H to P in a sequence. Consequently, this will affect the component of optimal sequences. Our linear programming's objective function will change with the values changing of α and β. Then the optimal solution for the linear programming could alter. In the process of observing the effect on optimal sequences with the changing of α and β's values, we can hold one of them as fixed, and wave the other, then look at the percentage of agreement between the natural and optimal sequences for each set of α and β. Through this method, we could find the set of α and β with the highest score in agreement percentage. Or according to the results of α and β's values, the structures could be divided into several groups. For different group, respectively, α and β are assigned different values to the optimal function. This will help us to improve the performance of our network linear programming model.

4.2.2 Inverse binding problem

Protein folding and protein binding are two intimately related problems. Similarly, protein inverse folding and protein inverse binding are closely related to each other. Inverse binding problem is to seek to design protein sequence for optimal binding. Once residues in a binding region are fixed, they will be withdrawn from the set of candidate residue nodes in the graph and their labels do not need to be assigned during optimization. Furthermore, if two

contacting residues *i* and *j* are both labeled as H, they will not contribute to the cost of pairwise contact energy. Edge node (s, u_{ij}) will have a label H and will not be subject for optimization. If either one of the two contacting residues is a P residue, the contact node u_{ij} will have a label P by definition, and edge (s, u_{ij}) will always contribute a constant $|\alpha|$ to the cost function, and is therefore removed from optimization. In this case, the formulation of the linear programming is exactly the same as our network linear programming (5) described in Chapter 2.3.2, the difference is that the set of edges E' is now smaller:

$$E' = E - \{(s,u_{ij}) \mid v_i \in Porv_j \in P\}$$

$$- \{(s,u_{ij}) \mid v_i \in Horv_j \in H\}$$

$$- \{(v_i,t) \mid v_i \in Porv_j \in H\}$$

Then what we need to do is to search the optimal solutions on the small set. The inverse binding problem has important implications in protein design, protein engineering, and in searching therapeutic agent.

4.3 Conclusion

In this study, we explore a new fitness function based on the Grand Canonical Sequence Evolution model of Sun *et al.* and Kleinberg's flow network transformation, and we have solved this protein inverse folding problem by providing a novel simple and efficient linear programming method to construct optimal sequences. Furthermore, we demonstrate our model's

effectiveness by implementation on 23 structures drawn from the Protein Data Bank.

We prove the correctness of the linear programming method theoretically in Chapter 2. And in Chapter 3 we apply our linear programming on 23 structures, and collate the designed sequences by linear programming method with the known native sequences of biological proteins, the agreement ranges from 58% to 79%. Furthermore, we compare the optimal sequences from linear programming method with those from Sun et al.'s GCSE model and Kleinberg's flow network algorithm. The percentage agreement between natural and optimal sequences changes with the markedly different structures, as we move among Sun et al.'s algorithm, flow network algorithm and our linear programming. The length-weighted average of percentage agreement has no big difference among these three methods, 72.1% for Sun et al.'s GCSE model, 72.70% for Kleinberg's flow network algorithm, and 72.15% for the linear programming, respectively. We can see that the performance of the linear programming is as good as other algorithms' performance. So we can conclude that the proposed linear programming method for solving protein inverse folding problem is reliable and efficient, and it may be useful tools for designing monomer sequences that will fold to specific target conformations of protein and other polymers.

In addition, the extension of the linear programming method—adding specified H/P composition as a constraint in the linear programming problem—varies the percentage of agreement between the natural and optimal sequences for different structures in different ways. If the specified ratio of H to P in a sequence is equal to the natural ration of H to P, then the network

linear programming with fixed H/P composition constraint will increase the agreement percentage in most cases; if the specified ratio of H to P is different from the natural ration of H to P in a sequence, then the agreement percentage will be decreased in most cases, as we move from the linear programming to the linear programming with fixed H/P composition constraint.

CITED LITERATURE

Ahuja, R., Magnanti, T., and Orlin, J. : Network Flows. Prentice-Hall, Englewood Cliffs, NJ, 1993.

Banavar, J., Cieplak, M., Maritan, A., Nadig, G., Seno, F., and Vishveshwara, S. : Structure-based design of model proteins. *Proteins: Struct. Funct. Genet.* 31: 10-20, 1998.

Banavar, J.R., and Maritan, A.M. : Geometrical approach to protein folding: a tube picture. *Reviews of Modern Physics* 75: 23-24, 2003.

Cherkassky, B., and Goldberg, A.V. : On implementing the push-relabel method for the maximum flow problem. *Proc. MPS Symp. On Int. Prog. And Comb. Optimization* : 157-171, 1995.

Chothia, C. : One thousand families for the molecular biologist. *Nature* 357: 543-544, 1992.

Cormen, J., Leiserson, C., and Rivest, R. : Introduction to Algorithms. McGraw-Hill, New York, 1990.

Creighton, T.E. : Proteins: Structure and Molecular Properties. Freeman, San Francisco, 1993.

Desjarlais, J.R., and Handel, T.M. : De novo design of the hydrophobic cores of proteins. *Protein Science* 4: 2006-2018, 1995.

Desmet, J., De Maeyer, M., Hazes, B., and Lasters, I. : The dead-end elimination theorem and its use in protein side-chain positioning. *Nature* 356: 539-542, 1992.

Deutsche, J.M., and Kurosky, T. : New algorithm for protein design. *Physical Review Letters* 76(2): 323-326, 1996.

Drexler, K.E. : Molecular engineering: An approach to the development of general capabilities for molecular manipulation. *Proc. Natl. Acad. Sci. U.S.A.* 78: 5275-5278, 1981.

Goldberg, A.V., and Tarjan, R.E. : Anew approach to the maximum flow problem. *J. ACM* 35: 921-940, 1988.

Gupta, Arvind, Manuch, Jan, and Stacho Ladislav : Structure-approximating inverse protein folding problem in the 2D HP model. *Journal of Computational Biology* 12: 1328-1345, 2005.

Gutin, A.M., Abkevich, V.I., and Shakhnovich, E.I. : Is Burst Hydrophobic Collapse Necessary for Protein Folding? *Biochemistry* 34 (9): 3066–3076, 1995.

Hart, W. : On the computational complexity of sequence design problems. *Proc. RECOMB Conf. Comput. Mol. Biol.* :128-136.

Hellinga, H.W., and Richards, F.M. : Optimal selection of sequences f proteins of known structure by simulated evolution. *Proceedings of the National Academy of Sciences* 91: 5803-5807, 1994.

Hillier, Frederick S. and Lieberman Gerald J. : Introduction to Operations Research, McGraw-Hill, 2001.

Jon M. Kleinberg : Efficient algorithms for protein sequence design and the analysis of certain evolutionary fitness landscapes. *Journal of computational biology* 6: 387-404, 1999.

Kamtekar, S., Schiffer, J.M., Xiong, H., Babik, J.M., and Hecht, M.H. : Protein design by binary patterning of polar and nonpolar amino acids. *Science* 262: 1680-1685, 1993.

Lee, C., and Levitt, M. : Accurate prediction of the stability and activity effects of site-directed mutagenesis on a protein core. *Nature* 352: 448-451, 1991.

Lippow, S.M., and Tidor, Bruce : Progress in computational protein design. *Biotechnology* 18: 305-311, 2007.

Merz, K., and LeGrand, S., eds. : The protein folding problem and tertiary structure prediction. Birkhauser, Boston, 1994.

Nemhauser, George L., and Wolsey, Laurence A. : Integer and Combinatorial Optimization, John Wiley & Sons, 1988.

Ponder, J., and Richards, F.M. : Tertiary templates for proteins. *J. Mol. Biol.* 193: 63-89, 1987.

Raha, Kaushik, Wollacott, A.M., Italia, Michael J., and Desjarlais, John R. : Prediction of amino acid sequence from structure. *Protein Science* 9: 1106-1119, 2000.

Richmond, T.J., and Richards, F.M. : Packing of -helices: Geometrical constraints and contact areas. *J. Mol. Biol.* 119: 537-555, 1978.

Sanjeev, B.S., Patra, S.M., and Vishveshwara, S. : Sequence design in lattice models by graph

theoretical methods. *Journal of Chemical Physics* 114(4): 1904-1914, 2001.

Saven, J.G., and Wolynes, P.G. : Statistical mechanics of the combinatorial synthesis and analysis of folding macro-molecules. *Journal of Physics and Chemistry B* 101: 8375-8389, 1997.

Shakhnovich, E.I. : Proteins with selected sequences fold into unique native conformation. *Physical Review Letters* 72: 3907-3910, 1994.

Shakhnovich, E.I., and Gutin, A.M. : Anew approach to the design of stable proteins. *Protein Eng.* 6:793-800, 1993.

Sleator, D., and Tarjan, R.E. : A data structure for dynamic trees. *J. Comput. Syst. Sci.* 26: 362-391, 1983.

Sun, B., Brem, R., Chan, H.S., and Dill, K.A. : Designing amino acid sequences to fold with good hydrophobic cores. *Protein Eng.* 8: 1205-1213, 1995.

Wolsey, Laurence A. : Integer Programming, John Wiley & Sons, 1998.

Yue, K., and Dill, K.A. : Inverse protein folding problem: Designing polymer sequences. *Proc. Natl. Acad. Sci. U.S.A.* 89: 4163-4167, 1992.

Zou, J., and Saven, J.G. : Statistical theory of combinatorial libraries of folding proteins: energetic discrimination of a target structure. *Journal of Molecular Biology* 296: 281-294, 2000.

YING WANG

EDUCATION

Aug.07-present Master Student in Bioinformatics, Department of Bioengineering,
University of Illinois at Chicago, IL, US
GPA:3.8/4.0

Sep.02-Jun.05 Master of Management, major in Technical Economics and Management,
School of Economics and Management, Beijing Jiaotong University, Beijing, China
Rank: 1/47, GPA(Overall): 3.87/4.0

Sep.97-Jul.01
Bachelor of Engineering, major in Mechanical Engineering and Automation,
School of Mechanical and Electronic Control Engineering,
Beijing Jiaotong University, Beijing, China
Rank: top5%, GPA(Overall): 3.54/4.0

Minor in Economics,
School of Economics and Management,
Beijing Jiaotong University, Beijing, China

CERTIFICATES

SAS® Certified Advanced Programmer for SAS®9 *2010*
SAS® Certified Base Programmer for SAS®9 *2009*

RESEARCH EXPERIENCE

- Molecular and Systems Computational Bioengineering Lab,
University of Illinois at Chicago, IL, US *Sep.07-May.08*
Constructed optimal hydrophobic and polar interaction models for protein inverse folding problem using graph theory and linear programming;

- Engineering Design and Optimization Laboratory,
Clemson University, SC, US *Aug.06-Jul.07*
Researched on optimization design and structure design; Developed and programmed for different optimal algorithms; Set the values of parameters with statistical methods to build mathematical

modeling and made simulations on Augmented Lagrangian Coordination for Decomposed Design Problems;

- School of Economics and Management,
  Beijing Jiaotong University, Beijing, China *Oct.04-Jun.05*
  Created statistical models to analyze competitive strengths and forecasted growth trends, using data acquisition, data mining and statistical inference with regression analysis and ANOVA in SAS.

- Laboratory of Rail Traffic Control and Safety,
  Beijing Jiaotong University, Beijing, China *Nov.00-Jun.01*
  Completed data collection, data cleansing; Built dynamic models to simulate the distribution of passenger flow; Executed risk analysis for railway stations.

- Laboratory of Mechanical and Design,
  Beijing Jiaotong University, Beijing, China *Sep.00-Nov.00*
  Participated in the project of the Traffic Safe Information Simulation System by microcontroller;

TEACHING EXPERIENCE

- Department of Bioengineering,
  University of Illinois at Chicago, IL, US *Aug.08-Jan.09*
  Instructed the course of BIOE339 Biostatistics I

- Department of Bioengineering,
  University of Illinois at Chicago, IL, US *Jan.08-May.08*
  Instructed the course of BIOE580 Principles of Bioinformatics

- Department of Bioengineering,
  University of Illinois at Chicago, IL, US *Sep.07-Dec.08*
  Instructed the course of BIOE480 Introduction to Bioinformatics

- Department of Economics and Management,
  Beijing Jiaotong University, Beijing, China *Feb.03-Jul.03*
  Instructed the course of Applied Statistics for undergraduates

- School of Mechanical and Electronic Control Engineering,
  Beijing Jiaotong University, Beijing, China *Sep.98-Jul.99*
  Instructed the course of Engineering Mechanics for undergraduates

WORKING EXPERIENCE

- Quantitative Analyst, Hedge Fund Research Inc., Chicago, IL, US *2010-present*
- Financial Analyst, Beijing Jianxin Assets Appraisal Co. Inc., Beijing, China *2005-2006*

COMPUTER ABILITY

SAS, SAS Enterprise Guide, Excel, SPSS, PowerPoint, Matlab, Perl, R, SQL (SQL Server, MySql, Access), Linux, Windows, C, Maple, Outlook, Photoshop, AutoCAD.