Inverse Protein Folding Problem via Quadratic Programming

Andrii Riazanov, Mikhail Karasikov, Sergei Grudinin

Inverse Protein Folding Problem via Quadratic Programming

Andrii Riazanov
Skolkovo Institute of Science and Technology
Moscow Institute of Physics and Technology
andrei.ryazanov@phystech.edu

Mikhail Karasikov
Moscow Institute of Physics and Technology
Skolkovo Institute of Science and Technology
karasikov@phystech.edu

Sergei Grudinin
University of Grenoble Alpes, LJK
CNRS, LJK
Inria
sergei.grudinin@inria.fr

Abstract

This paper presents a method of reconstruction a primary structure of a protein that folds into a given geometrical shape. This method predicts the primary structure of a protein and restores its linear sequence of amino acids in the polypeptide chain using the tertiary structure of a molecule. Unknown amino acids are determined according to the principle of energy minimization. This study represents inverse folding problem as a quadratic optimization problem and uses different relaxation techniques to reduce it to the problem of convex optimizations. Computational experiment compares the quality of these approaches on real protein structures.

1. Introduction

A protein is a sequence of basic molecules of amino acids. Proteins are essential compounds of all living organisms, they are involved in almost all structural, catalytic, sensory, and regulatory functions. Properties and functions of proteins mostly depend on their structure. Protein engineering is aimed at studying of protein structures and it finds applications in a number of areas. For example, the design of proteins with desired shapes and properties is required in such areas as medicine, biotechnology, synthetic biology, nanotechnologies, etc.

While *computational protein folding* predicts the structure of a given sequence, the goal of *computational inverse protein folding problem* is to find amino acid sequences that would fold best into a given 3D shape or scaffold. It can be referred as *computational protein design* (CPD), which targets at choosing a sequence of amino acids to perform a particular task. As there are 20 possible amino acids for each position in the protein chain, the variety of sequences for a specific 3D shape grows exponentially. In CPD, the inverse folding problem is formulated as an optimization problem, aimed at the minimization of the energy of the protein.

The inverse folding problem is usually solved including the fact that each residue can adopt multiple conformational states, which are most often grouped into a finite number of conformational isomers (rotamers). Two approximations are common for this problem. First, it is assumed that the protein's backbone is fixed. Second, the continuous domain of available conformations for the side-chains is approximated using a finite set of discrete conformations, defined by inner dihedral angles. The CPD is then formulated as the problem of determining the conformation corresponding to the minimum energy from that finite set.

A variety of algorithms was developed to solve the inverse folding problem. In [1], the problem is expressed as a Cost Function Network (CFN), also known as the Weighted Constraint Satisfaction Problem (WCSP). The interaction energies are represented as cost functions, and the unknown variables are the sets of pairs *(amino acid, conformation)*. Then, the problem is solved using the dead-end elimination/A^* algorithm (DEE/A^*). More precisely, first, the *DEE* algorithm reduces the computational cost of the problem, excluding from the consideration conformations that are unlikely to be optimal. Then, the A^* algorithm allows to expand a sequence-conformation tree, so that sequence-conformations are extracted and sorted on the basis of their energy values. The CFN prob-

lem is solved using the *toulbar2* solver. The investigation [1] also compares the CFN approach with several other optimization models. CPD is expressed as a 0/1 linear programming (01LP) problem, which is defined by a linear criterion to optimize over a set of Boolean variables under a conjunction of linear equalities and inequalities. It is also represented as a problem of 0/1 quadratic programming (01QP), which has a quadratic criterion to optimize with the same types of constraints as in 01LP. To solve these both models, the authors used the *cplex* solver and compared the results with the CFN approach.

In [6], the inverse folding problem is solved using genetic algorithm. This algorithm optimizes the sequence of amino acids in a protein chain with a stochastic search similar to natural selection in evolution. This approach was later improved in [8] by expressing the problem as a highly multi-modal optimization problem. This is achieved by using a diversity measure as the objective function through multi-objectivization. The common feature of all genetic algorithms is that they rely on evolutionary information of existing structures.

Investigation [7] studies the Answer Set Programming (ASP) approach for the inverse folding problem. It uses ***DEE*** in combination with branch and bound algorithms to eliminate high number of non-optimal pairs *(amino acid, conformation)*, as in [1]. Then the problem is represented in the ASP model and solved with solver *clingo4*.

All the aforementioned investigations determined both amino acid and conformation for all positions in the polypeptide chain. These approaches used energy functions which consider interactions between all possible pairs of rotamers, for example, energy function, implemented in the CPD dedicated tool *osprey 2.0* [17]. The mentioned works compare the efficiency of different algorithms, i.e. the time these algorithms need to find the minimum of energy for given instances. The CFN model demonstrated the best results overall, although even it could not solve all the instances in reasonable time. However, the quality of prediction of the primary structure (the coincidence between the found structure and the native one) is not investigated in these works. This quality depends on the choice of energy potential, and there is a challenging problem to find a practical scoring potential, which does not always match with the true potential function [18].

The main distinction of our investigation is that only the amino acids in the protein chain are predicted, and spatial conformations (rotamers) are not considered. This approach decrease the computational complexity of the problem significantly. It allows to receive basic information about the primary structure of a protein using much less resources, and allows to find the sequence of amino acids even for the most complicated instances. Thus, this study uses energy functions [14, 15], which describe only interactions between all possible pairs of amino acids, but not rotamers.

In this study the inverse protein folding problem is reduced to the quadratical programming problem, which can be written as

$$\begin{aligned} &\text{minimize} \quad f_0(\vec{x}) \\ &\text{subject to} \quad f_i(\vec{x}) \le 0, \quad i = 1,2,\ldots,N, \end{aligned} \tag{1}$$

where

$$f_i(x) = \vec{x}^{\mathsf{T}}\mathbf{A}_i\vec{x} + 2\vec{b}_i^{\mathsf{T}}\vec{x} + c_i, i = 0,1,\ldots,N.$$

The initial problem is not convex, because the matrices $\mathbf{A}_i$ are generally indefinite, therefore the problem is NP-hard and can not be solved efficiently. This paper studies semidefinite relaxation , Lagrangian relaxation [3] and continuous relaxation of the problem (1) to convex one, which can be solved in practical time. The solution of the relaxed problem provides a lower bound of the optimal value of (1). We use the rounding of the relaxed solution to find an upper bound and an approximate the optimal value. This paper also suggests Sequential Quadratic Programming (SQP) [4] and Simulated Annealing approaches to solve (1). Finally, we compare these relaxations and techniques and estimate the efficiency and the quality of primary structure prediction of using different approaches on real proteins structures from the SCWRL4 test set [5].

2. Problem statement

Let the protein chain consist of N amino acids. The set $\mathscr{C} = \{1,2,\ldots,20\}$ contains indexes that encode all 20 possible amino acids. Let $\vec{y} = (y_1,\ldots,y_N)$ be the sequence of residues, where $y_i \in \mathscr{C}$. Let functions $E_{kl} : \mathscr{C}^2 \to \mathbb{R}$ define symmetrical pairwise interaction energies between residues k and l. Then the problem of energy minimization is represented as

$$\sum_{j=1}^{N}\sum_{i=1}^{N} E_{ij}(y_i,y_j) \to \min_{y_1,y_2,\ldots,y_N \in \mathscr{C}}. \tag{2}$$

The problem can also be written in the following way:

$$\begin{aligned} &\underset{\vec{x}=[\vec{x}_1^{\mathsf{T}},\ldots,\vec{x}_N^{\mathsf{T}}]^{\mathsf{T}}}{\text{minimize}} \quad \vec{x}^{\mathsf{T}}\mathbf{Q}\vec{x} \\ &\text{subject to} \quad \vec{x}_k \in \{0,1\}^{20}, \qquad k = 1,\ldots,N, \\ &\qquad\qquad \|\vec{x}_k\|_0 = 1, \qquad k = 1,\ldots,N, \end{aligned} \tag{3}$$

where

$$\mathbf{Q} = \begin{bmatrix} [E_{11}] & [E_{12}] & \cdots & [E_{1N}] \\ [E_{21}] & [E_{22}] & \cdots & [E_{2N}] \\ \vdots & \vdots & \ddots & \vdots \\ [E_{N1}] & [E_{N2}] & \cdots & [E_{NN}] \end{bmatrix},$$

$$E_{ij} = \begin{bmatrix} E_{ij}(c_1,c_1) & E_{ij}(c_1,c_2) & \cdots & E_{ij}(c_1,c_{20}) \\ E_{ij}(c_2,c_1) & E_{ij}(c_2,c_2) & \cdots & E_{ij}(c_2,c_{20}) \\ \vdots & \vdots & \ddots & \vdots \\ E_{ij}(c_{20},c_1) & E_{ij}(c_{20},c_2) & \cdots & E_{ij}(c_{20},c_{20}) \end{bmatrix},$$

$$\mathbf{Q} \in \mathbb{R}^{20N \times 20N}, \quad E_{ij} \in \mathbb{R}^{20 \times 20}.$$

Finally, it is practical to re-write the problem as

$$\begin{aligned} \underset{\vec{x} \in \{0,1\}^{20N}}{\text{minimize}} \quad & \vec{x}^{\mathsf{T}} \mathbf{Q} \vec{x} \\ \text{subject to} \quad & \mathbf{A}\vec{x} = \mathbf{1}_N, \end{aligned} \tag{4}$$

where

$$\mathbf{A} = \begin{bmatrix} 1\cdots1 & 0\cdots0 & \cdots\cdots & 0\cdots0 \\ 0\cdots0 & 1\cdots1 & \cdots\cdots & 0\cdots0 \\ \vdots & \vdots & \ddots & \vdots \\ \underbrace{0\cdots0}_{20} & \underbrace{0\cdots0}_{20} & \cdots\cdots & \underbrace{1\cdots1}_{20} \end{bmatrix}, \quad \mathbf{A} \in \{0,1\}^{N \times 20N}.$$

To estimate the quality of primary structure prediction, we use the scoring matrix BLOSUM62 [9]. This matrix is a substitution matrix used for calculating the degree of matching between two protein sequences.

Let $\mathbf{B}$ represents the BLOSUM62 matrix. Then, the score $\mathbf{B}(y_i, \hat{y}_j)$ is the score for the substitution of residue y_i for $\hat{y}_j$ at j-position in the chain. If the score is positive, the residues are interchangeable (or equal). Otherwise, this substitution is less likely to occur, and the prediction is wrong. Let $\vec{y}_{\text{nat}}, \vec{y}_{\text{pred}} \in \mathscr{C}^N$ be native and predicted sequences of length N, respectively. Then, the quality function can be defined as

$$S(\vec{y}_{\text{nat}}, \vec{y}_{\text{pred}}) = \frac{\sum_{k=1}^{N} \mathbf{B}((\vec{y}_{\text{nat}})_k, (\vec{y}_{\text{pred}})_k)}{\sum_{k=1}^{N} \mathbf{B}((\vec{y}_{\text{nat}})_k, (\vec{y}_{\text{nat}})_k)} \tag{5}$$

A good quality prediction corresponds to $S > 0$, and the best predictions have values of S close to 1. Negative values of S mean poor quality predictions.

The aim of this study is to compare different optimization approaches to solve the problem (4). The value of S is considered as the quality of the primary structure prediction, while the optimization power of algorithms is estimated from the ability of the algorithm to predict the sequence with the lowest possible energy.

3. Algorithms and relaxations

3.1. Greedy algorithm

Given a problem (2)

$$\sum_{j=1}^{N} \sum_{i=1}^{N} E_{ij}(y_i, y_j) \to \min_{y_1, y_2, \ldots, y_N \in \mathscr{C}},$$

consider the greedy search algorithm to find an approximate solution:

Algorithm 1 Greedy algorithm

Require: $\vec{y}_{\text{start}} = (y_1^0, \ldots, y_N^0),\ y_k^0 \in \mathscr{C},\ k = \overline{1,N}$
Ensure: $\vec{y}_{\text{pred}} = (y_1, \ldots, y_N)$
  repeat
    $num\_changed \leftarrow 0$
    for all $k \in \{1, \ldots, N\}$ **do**
      $\hat{y}_k := \underset{y'_k \in \mathscr{C}}{arg\,min}\, f(y_1, \ldots, y_{k-1}, y'_k, y_{k+1}, \ldots, y_N)$
      if $\hat{y}_k \neq y_k$ **then**
        $y_k := \hat{y}_k$
        $num\_changed := num\_changed + 1$
  until $num\_changed > 0$

On each iteration, this algorithm searches for a protein conformation of a lower energy than the current one among conformations in the neighborhood of the current conformation. On each step it determines for each position of the chain whether there exists such an amino acid $\hat{y}_k$ that would decrease the energy of the protein by substitution y_k with $\hat{y}_k$. It stops when there is no position to change, so this algorithm returns at least the local minimum of the problem (2).

3.2. Continuous relaxation

The matrix $\mathbf{Q}$ in (4) could be relaxed to semidefinite positive by shifting its spectrum. Denote

$$\hat{\mathbf{Q}} = \mathbf{Q} - \lambda_{\min}(\mathbf{Q}) \cdot \mathbf{I}_{20N}, \quad c = \lambda_{\min}(\mathbf{Q}) \cdot N. \tag{6}$$

Then

$$\vec{x}^{\mathsf{T}} \hat{\mathbf{Q}} \vec{x} = \vec{x}^{\mathsf{T}} (\mathbf{Q} - \lambda_{\min} \mathbf{I}_{20N}) \vec{x} = \vec{x}^{\mathsf{T}} \mathbf{Q} \vec{x} - \lambda_{\min} \vec{x}^{\mathsf{T}} \vec{x}.$$

Considering $\vec{x} \in \{0,1\}^{20N}$ and $\mathbf{A}\vec{x} = \mathbf{1}_N$, we have $\vec{x}^{\mathsf{T}}\vec{x} = N$. Therefore, optimization problem (4) can be written as:

$$\begin{aligned} \underset{\vec{x} \in \{0,1\}^{20N}}{\text{minimize}} \quad & \vec{x}^{\mathsf{T}} \hat{\mathbf{Q}} \vec{x} + c \\ \text{subject to} \quad & \mathbf{A}\vec{x} = \mathbf{1}_N. \end{aligned} \tag{7}$$

Now the matrix $\hat{\mathbf{Q}}$ is semidefinite positive due to (6), because all its eigenvalues are nonnegative. Consider continuous relaxation of (7) by relaxing constraints $\vec{x} \in \{0,1\}^{20N}$ to $\vec{x} \geq \vec{0}_{20N}$:

$$\begin{aligned} \underset{\vec{x} \geq \vec{0}_{20N}}{\text{minimize}} \quad & \vec{x}^{\mathsf{T}} \hat{\mathbf{Q}} \vec{x} + c \\ \text{subject to} \quad & \mathbf{A}\vec{x} = \mathbf{1}_N. \end{aligned} \tag{8}$$

The problem (8) is convex now, and therefore can be solved efficiently. The solution of (8) also gives the lower bound for the optimum of the primary problem (4). An approximate solution of (4) and an approximate optimum are found from $\vec{x} = [\vec{x_1}^{\mathsf{T}}, \ldots, \vec{x_N}^{\mathsf{T}}]^{\mathsf{T}}$ as follows:

$$y_k = \underset{j=\overline{1,20}}{arg\,max}\ x_k^j, \quad k = 1, \ldots, N.$$

The solution could be also improved by using the greedy algorithm 1 with the solution of (8) as a starting position. The final approximation will be the upper bound for the optimum value of the initial problem (4).

3.3. Semidefinite relaxation (SDP)

SDP relaxation is described in [2]. Applying it to the problem (4) we have

$$\begin{aligned} \underset{\substack{\mathbf{X}\in\mathbb{R}^{20N\times 20N}\\ \vec{x}\geq\vec{0}_{20N}}}{\text{minimize}} \quad & \mathrm{Tr}\,(\mathbf{QX}) \\ \text{subject to} \quad & \mathbf{X}\succeq\vec{x}\vec{x}^{\mathrm{T}}, \\ & X_{ii}=x_i,\ i=\overline{1,20N}, \\ & \mathbf{A}\vec{x}=\vec{1}_N, \end{aligned}$$

where $\mathscr{S}_+^n=\{\mathbf{X}\in\mathbf{R}^n:\ \mathbf{X}=\mathbf{X}^{\mathrm{T}}\succeq 0\}$ is the set of symmetric semidefinite positive matrices of dimension N. By Schur's complement [10], the problem is equivalent to

$$\begin{aligned} \underset{\substack{\mathbf{X}\in\mathbb{R}^{20N\times 20N}\\ \vec{x}\geq\vec{0}_{20N}}}{\text{minimize}} \quad & \mathrm{Tr}\,(\mathbf{QX}) \\ \text{subject to} \quad & \begin{bmatrix}\mathbf{X} & \vec{x}\\ \vec{x}^{\mathrm{T}} & 1\end{bmatrix}\in\mathscr{S}_+^{20N+1}, \\ & X_{ii}=x_i,\ i=\overline{1,20N}, \\ & \mathbf{A}\vec{x}=\vec{1}_N. \end{aligned} \tag{9}$$

Again, the relaxed problem is convex and can be solved efficiently using the methods of convex programming. The optimum of (9) gives the lower bound for (4). The binary vector $\vec{x}$ and the upper bound for the initial problem (4) are recovered from the solution in the same way as in the previous case (continuous relaxation).

3.4. Lagrangian relaxation

The Lagrangian relaxation [3, 10] is a good method for getting a cheaply computable lower bound on the optimal value of a non-convex quadratic optimization problem. This method uses the fact that the dual of a problem is always convex, and the weak duality guarantees the optimum value of a dual problem to be the lower bound to the initial one.

The Lagrangian of (4) is

$$\begin{aligned} L(\vec{x},\vec{\lambda},\vec{u}) &= \vec{x}^{\mathrm{T}}\mathbf{Q}\vec{x}+\sum_{i=1}^{20N}\lambda_i(x_i^2-x_i)+\vec{u}^{\mathrm{T}}(\mathbf{A}\vec{x}-\vec{1}_N)= \\ &= \vec{x}^{\mathrm{T}}\left(\mathbf{Q}+\mathbf{D}(\vec{\lambda})\right)\vec{x}-\vec{\lambda}^{\mathrm{T}}\vec{x}+\vec{u}^{\mathrm{T}}(\mathbf{A}\vec{x}-\vec{1}_N)= \\ &= \vec{x}^{\mathrm{T}}\left(\underbrace{\mathbf{Q}+\mathbf{D}(\vec{\lambda})}_{\mathbf{P}(\vec{\lambda})}\right)\vec{x}-\left(\underbrace{\vec{\lambda}^{\mathrm{T}}-\vec{u}^{\mathrm{T}}\mathbf{A}}_{\vec{q}^{\mathrm{T}}(\vec{\lambda},\vec{u})}\right)\vec{x}-\underbrace{\vec{u}^{\mathrm{T}}\vec{1}_N}_{r(\vec{u})}. \end{aligned}$$

The dual function is then

$$g(\vec{\lambda},\vec{u})=\inf_{\vec{x}\in\mathbb{R}^{20N}}L(\vec{x},\vec{\lambda},\vec{u})=\inf_{\vec{x}\in\mathbb{R}^{20N}}\vec{x}^{\mathrm{T}}\mathbf{P}(\vec{\lambda})\vec{x}-\vec{q}^{\mathrm{T}}(\vec{\lambda},\vec{u})\vec{x}-r(\vec{u})=$$

$$=\begin{cases}-\frac{1}{4}\vec{q}^{\mathrm{T}}(\vec{\lambda},\vec{u})\mathbf{P}^{+}(\vec{\lambda})\vec{q}(\vec{\lambda},\vec{u})-r(\vec{u}), & \text{if } \mathbf{P}(\vec{\lambda})\succeq 0 \text{ and} \\ & \vec{q}(\vec{\lambda},\vec{u})\perp\mathrm{Ker}\,\mathbf{P}(\vec{\lambda}), \\ -\infty, & \text{otherwise.}\end{cases}$$

Finally, the Lagrangian relaxation of (4) is:

$$\begin{aligned} \underset{\vec{\lambda}\in\mathbb{R}^{20N},\,\vec{u}\in\mathbb{R}^N}{\text{maximize}} \quad & \gamma-r(\vec{u}) \\ \text{subject to} \quad & \gamma\leq 0, \\ & \begin{bmatrix}\mathbf{P}(\vec{\lambda}) & \frac{1}{2}\vec{q}(\vec{\lambda},\vec{u})\\ \frac{1}{2}\vec{q}^{\mathrm{T}}(\vec{\lambda},\vec{u}) & -\gamma\end{bmatrix}\in\mathscr{S}_+^{20N+1}. \end{aligned} \tag{10}$$

3.5. Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) [4, 11] is one of the most effective methods for constrained nonlinear optimization. The idea of SQP is to solve nonlinear problem using a sequence of quadratic programming (QP) subproblems. The objective function of the subproblem on each step is a convex quadratic approximation of the Lagrangian function.

First, we relax the constraints of the initial problem (4) in order to apply the SQP method. The constraints $x_i\in\{0,1\}$ should be substituted for $0\leq x_i\leq 1$:

$$\begin{aligned} \underset{\vec{x}\in\mathbb{R}^{20N}}{\text{minimize}} \quad & \vec{x}^{\mathrm{T}}\mathbf{Q}\vec{x} \\ \text{subject to} \quad & \mathbf{A}\vec{x}=\mathbf{1}_N, \\ & \mathbf{0}_{20N}\leq\vec{x}\leq\mathbf{1}_{20N} \end{aligned} \tag{11}$$

The Lagrangian of (11) is

$$L(\vec{x},\vec{\lambda},\vec{\mu},\vec{u})=\vec{x}^{\mathrm{T}}\mathbf{Q}\vec{x}-\vec{\lambda}^{\mathrm{T}}\vec{x}+\vec{\mu}^{\mathrm{T}}(\vec{x}-\mathbf{1}_{20N})+\vec{u}^{\mathrm{T}}(\mathbf{A}\vec{x}-\vec{1}_N). \tag{12}$$

Now let us formulate the QP subproblem for the k-th step of the algorithm:

$$\begin{aligned} \underset{\vec{d}\in\mathbb{R}^{20N}}{\text{minimize}} \quad & \frac{1}{2}\vec{d}^{\mathrm{T}}\mathbf{H}_k\vec{d}+2\vec{x}_k^{\mathrm{T}}\mathbf{Q}\vec{d} \\ \text{subject to} \quad & \mathbf{A}\left(x_k\vec{+}d\right)=\mathbf{1}_N, \\ & \mathbf{0}_{20N}\leq x_k\vec{+}d\leq\mathbf{1}_{20N}. \end{aligned} \tag{13}$$

Here $\mathbf{H}_k$ is a positive definite approximation of the Hessian matrix of the Lagrangian function (12), and $\vec{x}_k$ is a current iterate. The solution of (13) is used to form a new iterate $(\vec{x}_{k+1},\vec{\lambda}_{k+1},\vec{\mu}_{k+1},\vec{u}_{k+1})$.
Each iteration of the algorithm consists of three steps:

1. Updating the Hessian Matrix
2. Solving a Quadratic Program
3. Line Search and Merit Function

1. At each iteration a quasi-Newton approximation of the Hessian $\nabla^2_{xx}L(\vec{x}_{k+1},\vec{\lambda}_{k+1},\vec{\mu}_{k+1},\vec{u}_{k+1})$ is calculated using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [12] method. Denote

$$\vec{s}_k = \vec{x}_{k+1} - \vec{x}_k,$$

$$\vec{y}_k = \nabla_x L(\vec{x}_{k+1}\vec{\lambda}_{k+1},\vec{\mu}_{k+1},\vec{u}_{k+1}) - \nabla_x L(\vec{x}_k,\vec{\lambda}_k,\vec{\mu}_k,\vec{u}_k)$$

Then the BFGS approximation for the next iteration has the form

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\vec{y}_k\vec{y}_k^{\mathsf{T}}}{\vec{y}_k^{\mathsf{T}}\vec{s}_k} - \frac{\mathbf{H}_k\vec{s}_k\vec{s}_k^{\mathsf{T}}\mathbf{H}_k^{\mathsf{T}}}{\vec{s}_k^{\mathsf{T}}\mathbf{H}_k\vec{s}_k}. \tag{14}$$

This approximation keeps $\mathbf{H}_k$ positive definite during the iterations.

2. With iterate $(\vec{x}_k,\vec{\lambda}_k,\vec{\mu}_k,\vec{u}_k)$ and a quasi-Newton approximation of Hessian $\mathbf{H}_k$, the subproblem QP (13) is formulated. Considering the condition on $\mathbf{H}_k$ to be positive definite, QP is convex and can be solved efficiently. New iterates for dual variables $\vec{\lambda}_{k+1},\vec{\mu}_{k+1},\vec{u}_{k+1}$ are determined from the solution immediately as the values of the dual variables of (13) in the optimal point. The new iterate for the primal variable is set to $\vec{x}_{k+1} = \vec{x}_k + \alpha_k\vec{d}_k$, where $\vec{d}_k$ is the solution of QP (13), and the optimum length of the step α_k is determined at the next step.

3. The step length parameter α_k is determined in order to produce a sufficient decrease in a merit function. We use the following function:

$$\Psi(\vec{x}) = \vec{x}^{\mathsf{T}}\mathbf{Q}\vec{x} + \vec{w}^{\mathsf{T}}\vec{x}_+ + \vec{p}^{\mathsf{T}}\left(\mathbf{1}_{\mathbf{20N}}\vec{} - \tilde{x}\right)_+ + \vec{r}^{\mathsf{T}}(\mathbf{A}\vec{x} - \mathbf{1}_N), \tag{15}$$

Here, if $\vec{m} = [\vec{w}^{\mathsf{T}},\vec{p}^{\mathsf{T}},\vec{r}^{\mathsf{T}}]^{\mathsf{T}}$, then $\vec{m}$ is calculated as

$$\vec{m}_i = (\vec{m}_{k+1})_i = \max\left\{\pi_i, \frac{(\vec{m}_k)_i + \pi_i}{2}\right\}, \quad i = 1,\ldots,41N,$$

where vector $\vec{\pi} = \left[\vec{\lambda}^{\mathsf{T}},\vec{\mu}^{\mathsf{T}},\vec{u}^{\mathsf{T}}\right]^{\mathsf{T}}$ is the vector of dual variables of QP (13). The parameter α_k is calculated then from the minimization of the univariate function

$$\Psi(\vec{x}_k + \alpha_k\vec{d}_k) \to \min_{\alpha_k}.$$

The problem (11) is then reduced to a sequential solving of convex quadratical optimization problems.

3.6. Simulated Annealing

Simulated Annealing (SA) [13] is used to compare different approaches. It is a probabilistic method, and although it is unlikely to find the global optimum solution, it may often find a solution very close to it.

Let $T : \mathbb{Z}_+ \to (0;+\infty)$ be the exponentially decreasing function of the temperature, and $\mathbf{S}$ be a finite set of **states** – all manner of possible sequences. The set $N(x)$ is a set of **neighbors** to the state $x \in \mathbf{S}$. The energy function $F(x)$ is the objective function. $x_0 \in \mathbf{S}$ is the starting position. Then the iteration of SA algorithm is:

1. Randomly choose the state from the neighbors of a current state $x_{k+1} \in N(x_k)$
2. Assess the energy of the chosen state $F_{k+1} = F(x_{k+1})$
3. Compare F_{k+1} and F_k and decide, whether to accept selected move. If $F_{k+1} < F_k$, the move is accepted. Else, the move is accepted with the probability $\exp\left(-\frac{F_{k+1}-F_k}{T(k)}\right)$.

In this investigation problem, the set $\mathbf{S}$ is the set of all possible sequences of amino acids for a given length, neighbors $N(x)$ for the sequence x are the sequences, in which 5 or less residues differ from x. The energy function $F(x)$ is the energy of the protein structure.

4. Solution

This section describes the process of solving the problem (4) using methods, presented in the previous section. First, we build the energy matrix for a protein structure. In the current study, we use two coarse-grained distance-dependent potentials. One of them is described in [14]. Another is DFIRE-C_a potential [15], whose parameters are kindly provided by the authors. They both require only the distance between two C_α carbon atoms in a pair of amino acids to estimate the energy of interaction between these residues. So, by extracting coordinates of C_α atoms of the backbone of a given protein, we can build the energy matrix. That is, for each pair of positions i,j in the chain, we calculate the distance d_{ij} between C_α atoms at these positions. Then, the energy of the interaction between all 210 pairs of amino acids located at distance d_{ij} are estimated, and this is how matrices E_{ij} are obtained. By doing so for all $i,j \in \overline{1,N}$, we build the matrix $\mathbf{Q}$.

Then different relaxations are applied to the problem (4):

- Lagrangian relaxation (10) is used to receive the lower bound for the optimum value. It does not provide an approximate solution to the problem, because there is no strong duality between (4) and its dual.
- Greedy algorithm 1 is initialized with a random starting position.
- Continuous and SDP relaxations find both lower bound and approximate optimum of the problem. We improve received solutions with the greedy algorithm in the next step.

- SQP and Simulated Annealing give an approximate optimum of the energy.

Finally, we estimate the quality of optimization and prediction for each method at each test protein structure. The quality of prediction is obtained from (5) using the BLOSUM62 matrix.

5. Computational details

All the tested algorithms were implemented in *Python*, using the CVXPY package [16] to solve different convex problems that are obtained after the proposed relaxations. For the SQP algorithm, we used its MATLAB implementation from the Global Optimization Toolbox. The computation experiment was set on a quad-core Intel Core(TM) i7-4700HQ CPU @ 2.40GHz PC with 12 GB of RAM running on Ubuntu 14.04 LTS OS.

6. Results and Discussion

To compare the quality of different relaxation approaches in solving non-convex quadratic problems, we performed a series of computational experiments. More precisely, we used six algorithms, described in the section 3, to find the minimum of the protein energy. The computational experiments were set on a test set of protein structures extracted from the SCWRL4 benchmark. The average length of sequences for these proteins was 110 amino acids, so the dimension of $\vec{x}$ was about 2000.

Figure 1 presents the results of optimization the energy using the potential from [14].

From this figure we can see that among the compared methods, SQP and SA demonstrated the best power of optimization. They returned very proximate optimum energy values and predicted similar sequences of amino acids. The continuous relaxation approach minimizes the objective function worse than SA and SQP, but still returns the sequence with the energy, lower than the native. We should note that the SDP and Lagrangian relaxations methods cannot be applied to large optimization problems. This is because the number of their variables is quadratic with respect to the length of vector $\vec{x}$, and also because the computational requirements for these approaches exceed the available resources for the sequences of a practical length. For example, for $N = 100$ the number of variables for these methods is more than $2 \cdot 10^6$.

For all the techniques and all the test structures, we measured the time spent on finding the solution of the optimization problem. It was done in the following way – for different lengths of sequences from 5 to 155, we truncated the length of the structures of proteins this upper limit. Then, we applied all the tested optimization methods to solve the problem with cropped backbones, and the average time for each length was measured. Also, we computed the mean of quality of prediction for such sequences. The dependences of computational time and quality on the length of a protein chain are shown in Figs. 2 and 3, respectively.

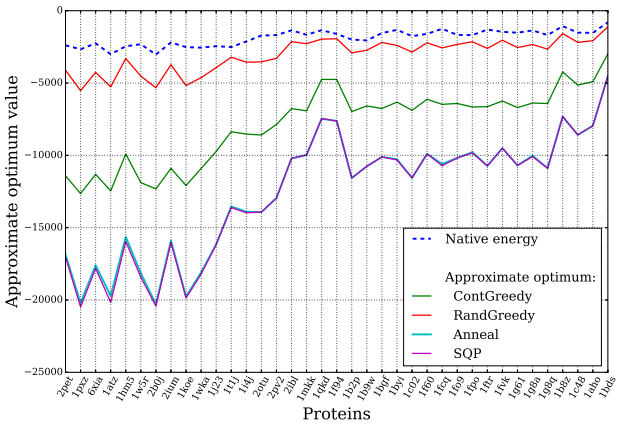

Figure 1. Approximate energy optimum for different relaxations computed on the test set.

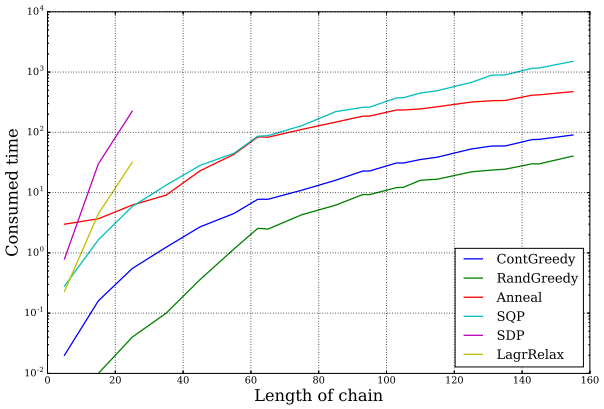

Figure 2. Computational time as a function of the length of a protein chain for all the algorithms in logarithmic scale.

Figure 1 clearly demonstrates that all the relaxations and all the tested methods found sequences of amino acids with the energy lower, than the one of the native state. The greedy algorithm with a native sequence of amino acids as the starting position also returned the sequence with a lower energy for all the tested potentials. This means that the native structures are not even the local minimum for the potential energy function used in this experiment. Therefore, the two tested potentials do not satisfy the crucial assumption, i.e., that the native conformation has the lowest

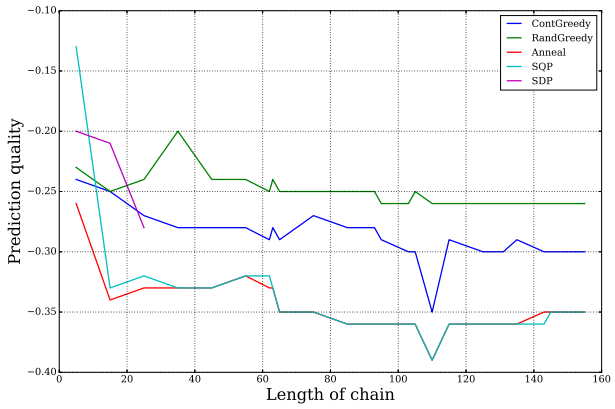

Figure 3. Dependence of prediction quality on the length of a protein chain for all the tested methods.

energy between all possible states for a given fixed backbone, and thus the quality of primary structure prediction is extremely poor. In searching for a minimum of energy, the algorithms select residue types with the lowest contact energies. This happens to be observed for pairs of cysteine-cysteine and tyrosine-tyrosine. As the result, the predicted structure often consist fully either of cysteine residues, or of tyrosine residues.

7. Conclusion

In this paper we proposed to formulate the Inverse Protein Folding Problem as a quadratic optimization problem, where the objective function is the energy of the protein, which is assumed to have the lowest value for the native conformation. The results shows that the SQP and SA approaches have the best optimization power on the test set of protein structures extracted from the SCWRL data set. These two methods returned very similar approximate optimums for nearly all the structures. The SDP approach turned out to increase the computational cost of the relaxed problem rapidly and required unreasonable time for the solution for any practical case. Continuous relaxation gave the worse approximation for all the test cases, compared to SQP and SA, but consumed less time to compute the answer. All materials needed for conducting an experiment can be found at `https://sourceforge.net/p/mlalgorithms/code/HEAD/tree/Group374/Ryazanov2016InverseFolding/code/`.

Although the tested algorithms demonstrated a good capacity in solving the optimization problem, the actual quality of the primary structure prediction was poor. This is mainly because the potential functions that we used for the energy estimation do not follow the assumption that the native sequence has the lowest energy for a given geometrical shape. Therefore, the structures that were found using the introduced techniques, had much lower energy than the native one, however the native and the predicted sequences coincided very weakly. We strongly believe that in order to achieve better prediction results, an improvement of the potential functions for the inverse folding problem is needed.

8. Acknowledgments

This work was supported by RFBR, grant 16-37-00111.

References

[1] David Allouche et. al. Computational protein design as an optimization problem. *Artificial Intelligence*, 2014.

[2] Stephen Boyd and Lieven Vandenberghe. Semidefinite Programming Relaxations of Non-Convex Problems in Control and Combinatorial Optimization. *Communications, Computation, Control, and Signal Processing*, pp. 279–287, 1997.

[3] Alexandre d'Aspremont and Stephen Boyd. Relaxations and Randomized Methods for Nonconvex QCQPs. *EE392o Class Notes*, 2003.

[4] Philip E Gill, Walter Murray, and Michael A Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, Jan. 2002.

[5] Georgii G. Krivov, Maxim V. Shapovalov, and Roland L. Dunbrack. Improved prediction of protein side-chain conformations with scwrl4. *Proteins: Structure, Function and Bioinformatics*, 77(4):778–795, 2009.

[6] David T. Jones. De novo protein design using pairwise potentials and a genetic algorithm. *Protein Science*, vol. 3, no. 4, pp. 567–574, Dec. 2008.

[7] Jacques Nicolas Hugo Bazille. Computational protein design: trying an answer set programming approach to solve the problem. *10th Workshop on Constraint-Based Methods for Bioinformatics*, 2014.

[8] Wiktor Jurkowski Sune S. Nielsen, Gregoire Danoy. A Novel Multi-objectivisation Approach for Optimising the Protein Inverse Folding Problem. *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, 2015.

[9] Henikoff JG. Henikoff S. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, vol. 89, no. 22, pp. 10915–10919, Nov. 1992.

[10] Lieven Vandenberghe Stephen Boyd. Convex Optimization. *Cambridge University Press*, 2004.

[11] J. Nocedal and S. J. Wright. Numerical Optimization, Second Edition. *Springer Series in Operations Research*, 2006. Chapter 18.

[12] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, vol. 13, no. 3, pp. 317–322, Mar. 1970.

[13] Dimitris Bertsimas, John Tsitsiklis, et. al. Simulated annealing. *Statistical science*, vol. 8, no. 1, pp. 10–15, Feb. 1993.

[14] C. A. Floudas R. Rajgaria, S. R. McAllister. A novel high resolution $C_\alpha - C_\alpha$ distance dependent force field based on a high quality decoy set. *Proteins: Structure, Function, and Bioinformatics*, vol. 65, no. 3, pp. 726–741, Nov. 2006.

[15] Liu S et al. Zhang C. An accurate, residue-level, pair potential of mean force for folding and binding based on the distance-scaled, ideal-gas reference state. *Protein Science*, vol. 13, no. 2, pp. 400–411, Feb. 2004.

[16] Steven Diamond and Stephen Boyd. CVXPY: A Python-Embedded Modeling Language for Convex Optimization. *Journal of Machine Learning Research*, 2016.

[17] P. Gainza, K.E. Roberts, I. Georgiev, R.H. Lilien, D.A. Keedy, C.Y. Chen, F. Reza, A.C. Anderson, D.C. Richardson, J.S. Richardson, et al. Osprey: protein design with ensembles, flexibility, and provable algorithms. *Methods Enzymol*, 2013.

[18] Ting Lan Chiu and Richard A.Goldstein. Optimizing potentials for the inverse protein folding problem. *Protein Engineering Design and Selection*, vol. 11, no. 9, pp. 749–752, Sep. 1998.