LINEAR PROGRAMMING BASED APPROACH TO THE DERIVATION OF A CONTACT POTENTIAL FOR PROTEIN THREADING

Tatsuya AKUTSU [a]
Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108, Japan

Hiroshi TASHIMO [b]
Department of Computer Science, Gunma University, 1-5-1 Tenjin, Kiryu, Gunma 376, Japan

This paper proposes a novel method of deriving a contact potential (a pair score function) for protein threading. In this method, the constraint that the score of the native threading is minimum over all possible threadings is expressed in a form of linear inequalities, and then parameters defining the contact potential are determined by applying a program package of linear programming. The most important advantage of this method over the previous methods is that this method can learn a score function from a small number of training data. The proposed method was evaluated using Lathrop and Smith's algorithm for finding optimal threadings and was shown to be effective for computing nearly correct threadings.

1 Introduction

The prediction of three-dimensional protein structure from amino acid sequence is one of the most important problem in computational biology. Although a lot of approaches were proposed, recent studies have been focused on an indirect approach in which, given an amino acid sequence and a set of structural models (structural templates), a structure into which the sequence is most likely to fold is computed. To test whether or not a sequence is likely to fold into a structure, an *alignment* between spatial positions in a protein structure (or a structural model) and amino acids in a sequence is computed using a suitable *score function* (equivalently, *potential function*). That is, an alignment which minimizes the total score (corresponding to potential energy) is computed. This minimization problem is called a *protein threading* problem, and an alignment between a sequence and a structure is called a *threading*. An alignment minimizing the total score is called an *optimal threading*.

To use protein threading in a predictive setting, there are two major problems: *how to compute an optimal threading*, and *how to derive a score function*.

[a] e-mail: takutsu@ims.u-tokyo.ac.jp
[b] Present address: Toyobo Co. Ltd., Tsuruga-city, Fukui 914, Japan.

For the former problem, a lot of methods have been proposed. Most methods use DP (dynamic programming) [3,7,15] or its variant such as double DP [9]. However, using DP based approach, it may fail to find an optimal threading when a pair score function is used. Although recent Asilomar meeting (CASP2) [6] revealed that protein threading was useful for protein structure prediction even if such DP-based methods were used and computed threadings were incorrect, computing optimal threadings seems important for futher improvement of protein threading. Recently, Lathrop and Smith proposed a novel branch-and-bound algorithm (LS-algorithm, in short) which always finds an optimal threading [11]. Although there are some limitations (gaps within a core segment are not allowed), their method has great advantage over the other methods since it always finds an optimal threading. They examined several score functions, but the results were not satisfactory in the sense of accuracy of the obtained threadings. They listed some problems to be solved for improving the accuracy, in which the improvement of score functions was included. Thus, the latter problem (i.e., developing a good score function) is very important.

A lot of methods have also been proposed for the latter problem. Usually, they are empirically derived from known tertiary structures. In most of them, score functions are derived using Boltzmann-like statistical methods based on the concept of *quasichemical approximation* [14]. In a simplest form of the quasichemical approximation, contact potential $\mu_{x,y}$ between residues (amino acid types) x and y is derived by $\mu_{x,y} = -T \ln f_{x,y}$, where $f_{x,y}$ is a normalized frequency of contacts between x and y extracted from the known structures, and T (*temperature*) denotes the energy scale [13,14]. Based on the quasichemical approximation, a lot of variants of score functions have been proposed. For example, Sippl developed distance-dependent potentials [16], Godzik *et al.* developed a potential with multi-body interactions [7], and Nishikawa and Matsuo developed a potential considering dihedral angles [15]. Although the quasichemical approximation is reasonable under the assumption that protein sequences are random [13], real protein sequences are not considered to be random. Thus, quasichemical approximation does not necessarily suit for real proteins.

By the way, a *real score function* should distinguish the native structure (native conformation) by making its energy minimum (or minimal) among all other conformations. Although it may be impossible to develop such a real score function, it is important to develop a score function which approximately satisfies the above requirement. There have been several such studies. Goldstein *et al.* maximized energy gap between the native structure and alternative structures (decoys), using an optimization procedure developed in the theory of neural networks [8]. However, their optimization procedure could be applied to only one protein, and thus they obtained a score function by averaging over

all proteins. Mirny and Shakhnovich used a Monte-Carlo method in order to simultaneously maximize energy gaps for all proteins [13]. However, it is not guaranteed that energy gaps are truly maximized because their optimization procedure is a Monte-Carlo type. Maiorov and Crippen used an iterative procedure to solve the constraints that energy of the native structure should be less than those of alternative conformations, where alternative conformations were generated by incorrect threadings through several tens of structures [12]. However, in their method, only threadings without gaps (insertions and deletions) were used and examined. Thus, it is not guaranteed that the optimal threading exactly matches with the *native threading* (i.e., the threading defined from the native conformation) if gaps between core segments are allowed.

In this paper, we focus on the problem of deriving a pair score function such that the optimal threading coincides (or approximately coincides) with the native threading where gaps between core segments are allowed. Note that we could not select a correct structure from template structures if we could not find correct threadings, where a *correct threading* means an optimal threading which exactly coincides with the native threading. To derive such a structure, we randomly generate alternative threadings and make a constraint:

$$score(\text{native threading}) + gap \ < \ score(\text{alternative threading})$$

for each alternative threading. Then, we find a score function which simultaneously satisfies these inequalities. Since each constraint is *linear* (if we assume a usual contact potential), we can use a *linear programming* (LP, in short) solver [18]. The above method is similar to the method of Maiorov and Crippen [12]. But, there are some important differences.

(1) Our score function was developed for computing a correct threading, while their score function was developed for selecting a correct conformation among candidate conformations and they did not consider gaps between core segments. Since energy gap between the native threading and a slightly perturbed threading is considered to be not large, their score function does not seem to be suitable for computing correct threadings. Moreover, in order to examine *self-threading* (i.e., threading of a sequence through its own structural model), we have implemented LS-algorithm [11].

(2) We use an LP solver to solve linear constraints directly, while they use an iterative procedure. Previously, there had been no LP solver which simultaneously could solve large number of constraints. However, owing to recent progress on the *interior point method* [10,18], recently developed LP solvers can solve a million of constraints. Therefore, we used such an LP solver and thus we could directly solve a large number of constraints.

In order to evaluate the proposed method, we examined self-threading

using the derived score function. The results show that the score function derived by the proposed method is as good as existing score functions. Since the most important advantage of the proposed method over statistical methods is that it requires a small number of training data, score functions can be specialized for some structural families. In order to confirm this property, we applied the proposed method to some structural families, and we obtained better threadings using specialized score functions.

Although we assumed a simple contact potential as a score function, the proposed method can be generalized for more complex score functions. Moreover, the proposed method may be modified in order to derive score functions for identification of transmembrane domains, sequence alignment and prediction of RNA secondary structures [1].

2 Method

2.1 Score Function

In this paper, we use a simple score function. Our score function consists of three kinds of functions: α_x, β_x, and $\mu_{x,y}$. α_x and β_x denote the scores of a single residue x when x is aligned into a core segment of α-helix and a core segment of β-strand, respectively. $\mu_{x,y}$ denotes the score (the score of a contact) between residues x and y, and it corresponds to a usual *pairwise contact potential*. We assume that the score between x and y is 0 if the Euclidean distance between residue positions (aligned positions) is more than D where we use $D = 7.0\mathring{A}$, otherwise the score between x and y is $\mu_{x,y}$. Since there are 20 kinds of amino acids, our score function consists of 250 parameters (20 for α_x, 20 for β_x, and 210 for $\mu_{x,y}$ where we assume $\mu_{x,y} = \mu_{y,x}$).

2.2 Model of Protein Threading

The model of protein threading used in this paper follows Jones *et al.* [9], Bryant and Lawrence [4], and Lathrop and Smith [11] (see Fig. 1). The structural model (structural template) corresponds to an annotated backbone trace of the secondary structure segments in the conserved core fold, where each core is either α-helix or β-strand. Core segments are connected by variable loop (or coil) regions. Loops are not considered part of the conserved fold. Therefore, a structural model is represented by a sequence of residue positions in core segments. We use positions of C_β atoms for denoting residue positions.

When an amino acid sequence is threaded through the structural model, gaps within a core segment are not allowed. That is, successive elements in a sequence must correspond to successive positions in a core. Alignment gaps

are confined to the connecting non-core loop regions, and thus the loop lengths are variable. This threading model is termed *gapped alignment*[4].

Here, we define the threading problem in a formal way. Let $P = \langle p_1, \ldots, p_n \rangle$ be a sequence of positions of residues (e.g., positions of C_β atoms) in core regions of a template structure. Note that residues in loop regions are ignored since the scores of these residues are always set to 0 in this paper. Let $p_{l_1} \ldots p_{l_2-1}$, $p_{l_2} \ldots p_{l_3-1}$, $\cdots$, $p_{l_k} \ldots p_n$ be core segments, where $p_{l_i} \ldots p_{l_{i+1}-1}$ corresponds to a core segment, and $l_1 = 1$. Let $A = a_1 a_2 \ldots a_m$ $(m \geq n)$ be an amino acid sequence, which is to be threaded. Then, a threading t is a function from $\{1, 2, \ldots, n\}$ to $\{1, 2, \ldots, m\}$ such that $t(i) > t(i-1)$ if $i = l_j$ for some j, otherwise (i.e., both p_i and p_{i-1} are included in the same core) $t(i) = t(i-1) + 1$. Note that this definition means that residue $a_{t(i)}$ is assigned to position p_i, the order of residues must be preserved, and successive elements in a sequence must correspond to successive positions in a core.

Then, the total score $score(t)$ of threading t is defined by

$$score(t) = \sum_{i=1}^{n} \gamma_{a_{t(i)}} + \sum_{i<j, d(i,j) \leq D} \mu_{a_{t(i)}, a_{t(j)}} ,$$

where $d(i,j)$ denotes the Euclidean distance between positions p_i and p_j, and

$$\gamma_x = \begin{cases} \alpha_x, & \text{if } x \text{ is aligned into } \alpha\text{-helix,} \\ \beta_x, & \text{if } x \text{ is aligned into } \beta\text{-strand.} \end{cases}$$

A threading t is called an *optimal threading* if $score(t) \leq score(t')$ holds for any threading $t' \neq t$.

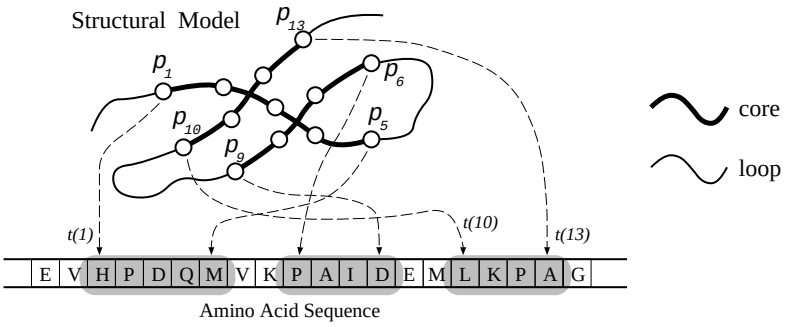

Figure 1: Model of protein threading. In this model, gaps between core segments are allowed, whereas gaps within a core segment are not allowed.

2.3 Making and Solving Constraints

Recall that our requirement for a score function is that the optimal threading coincides with the native threading. For that purpose, we should find a score function for which the score of the native threading is minimized over all possible threadings. Of course, if multiple proteins are given, we should find a score function for which the scores of all native threadings are simultaneously minimized. Although several similar studies have been done [8,12,13]. finding such a score function is very difficult [1]. Therefore, we develop a method which approximately satisfies the above requirement.

For simplicity, we consider a case that only one protein data (a sequence and its structural model) is given. But, it can be extended for a case of multiple protein data by merging the generated constraints.

The proposed method is conceptually very simple. Lathrop and Smith give a method to generate all possible threadings at uniformly random when a sequence and its structural model are given [11]. Based on their method, we randomly generate N alternative threadings $t_1, \cdots, t_N$ such that $t_i \neq t$ for all i, where t denotes the native threading. Then, we make the following LP instance (denoted by LP-A):

Maximize:	gap,	
Subject to:	$score(t) + gap < score(t_i)$,	(for all t_i)
	$-B \leq \alpha_x \leq B$, $-B \leq \beta_x \leq B$,	(for all x)
	$-B \leq \mu_{x,y} \leq B$,	(for all x, y)

where B is a constant in order to limit the ranges of parameters (currently, we use $B = 10.0$ because most of previous score functions use values between -10.0 to 10.0). Note that $score(t)$ is expressed by a linear combination of parameters (α_x,β_x,$\mu_{x,y}$) and thus all the above inequalities are linear. Then, we apply an LP solver [18] to the above instance. If there exists a set of values satisfying the above constraints, the LP solver outputs one which maximizes gap. Note that if we could generate all possible threadings, a score function that minimized the native threading would be derived. However, since generating all possible threadings takes too long time, we use randomly generated threadings.

In LP-A, every constraints of $score(t) + gap < score(t_i)$ must be satisfied and the LP solver fails to find a set of values if at least one of the constraints can not be satisfied. Therefore, we do not use LP-A. Instead we use the following LP instance (denoted by LP-B):

$$
\begin{array}{lll}
\textbf{Minimize:} & \sum_{i=1}^{N} e_i, & \\
\textbf{Subject to:} & score(t) + gap < score(t_i) + e_i, & (\text{for all } t_i) \\
& -B \leq \alpha_x \leq B,\ -B \leq \beta_x \leq B, & (\text{for all } x) \\
& -B \leq \mu_{x,y} \leq B, & (\text{for all } x, y) \\
& e_i \geq 0, & (\text{for all } i)
\end{array}
$$

where gap is a constant (currently we use $gap = 100.0$, which was determined from several trials). In this case, $score(t) + gap < score(t_i) + e_i$ can always be satisfied by letting $e_i > score(t) + gap - score(t_i)$. Since the LP solver tries to minimize $\sum e_i$, it is expected that most constraints are satisfied (by making $e_i = 0$). Moreover, if all the constraints can be satisfied, the LP solver always satisfies them by letting $\sum e_i = 0$.

Although LP-B seems good, the number of parameters becomes very large because there are as many parameters (e_i's) as constraints. Usually, CPU time for LP increases rapidly as the number of parameters increases. Therefore, we should keep the number of parameters as small as possible. Thus, instead of using distinct e_i's, we use the same parameter for e_i, e_{i+E}, e_{i+2E}, $\cdots$, where we currently use $E = 10$ (with no special reason). Note that, even in this case, the LP solver always satisfies all the constraints if it is possible.

3 Results

3.1 Hardware and Software

Computational experiments have been done on a SUN ULTRA-1 workstation (200MHz). All programs were written in C language. We used 'LOQO' [17] as an LP solver, where 'LOQO' is based on the interior point method and can solve very large LP instances [18]. In order to evaluate the quality of derived score functions, we implemented LS-algorithm in C language [11].

3.2 A Case of the Same Training Data Set and the Same Test Data Set

First we examined a fundamental case: the test data set is the same as the training data set. In particular, we examined two cases: (a) using a score function derived only from its own protein data; and (b) using a score function derived from all protein data in the data set. In each case, 500 or 1000 alternative threadings were generated per protein data. We use the following data set (23 protein data) obtained from Protein Data Bank [2]: 256b(A), 2end, 1rcb, 2mhr, 351c, 1bgc, 1ubq, 1mbd, 1lis, 1aep, 1hoe, 2hpr, 5cyt, 1bp2, 5cpv, 2mcm, 1plc, 1yat, 7rsa, 3fxn, 9rnt, 2sns, 2lzm. The results are summarized in Table 1. In this table, the following data are shown: the number of alternative

threadings generated (per protein) for deriving a score function; the number of proteins for which correct threadings were computed; the frequency of correct core segments (i.e., the number of correct segments / the number of core segments). Note that we call a core segment *correct* if the position in the optimal threading exactly coincides with the position in the native threading. You can see that a correct threading is computed in most cases when we use a score function obtained from its own protein data. Moreover, you can see that the number of correct threadings increases as the number of generated (alternative) threadings increases. However, if we use a score function obtained from multiple protein data, we can obtain only a few correct threadings although we can obtain many correct core segments. Moreover, the number of correct threadings does not necessarily increase even if the number of generated threadings increases.

Table 1: Relationship between the number of generated threadings and the number of correct threadings.

number of generated threadings	(a)		(b)	
	500	1000	500	1000
number of correct threadings	18	21	5	4
frequency of correct segments	0.97	0.99	0.53	0.55

3.3 A General Case

In order to evaluate the quality of the score function, we derive a score function from training data and computed self-threadings for test data. We use the following protein data set A as test data: 256b(A), 2end, 1rcb, 2mhr, 351c, 1bgc, 1ubq, 1mbd, 1lis, 1aep, 1hoe, 2hpr, 5cyt, 1bp2, 5cpv, 2mcm, 1plc, 1yat, 7rsa, 3fxn, 9rnt, 2sns, 2lzm, and we use the following protein data set B as training data: 3chy, 1pkp, 1aak, 8dfr, 1cde, 2cpl, 2cyp, 1f3g, 4fgf, 2act, 1dhr, 1mat, 1tie, 3est, 2ca2, 1byh, 1apa, 5tmn, 1lec, 1nar, 5cpa, 9api(A), 2had, 2cpp. Note that A and B are disjoint and all data are non-homologous. These protein data sets are subsets of one used by Lathrop and Smith [11]. It took 42 minutes (26 minutes for generating constraints and 16 minutes for solving LP) to derive the score function, where 500 alternative threadings were generated per protein (i.e., 12000 threadings were generated in total), and all generated constraints were satisfied (i.e., $\sum e_i = 0$). The average CPU time and the maximum CPU time for computing an optimal threading were 9.6 minutes and 88.5 minutes respectively. Since our implementation of LS-algorithm was not elaborated as their implementation, it took longer time and thus we used

small protein data as test data.

As in LS-paper [11], we measured the displacement between the optimal and the native threadings for each core segment across all self-threading trials. Error distributions from 89 α-helix and 77 β-strand core segment threadings are shown in Fig. 2. Note that in Fig. 2(a), frequencies of error $3, 4, -4$ are higher than those of neighbors. This reflects the amphipathic periodicity of α-helices. Fig. 2(b) also reflects the periodicity of β-strands. Similar phenomena were observed in LS-paper [11].

For α-helices, the sum of frequencies with errors between -4 and 4 (which are considered to be near miss) is 0.798. For β-strands, the sum of the same frequencies is 0.766. Although our test data set is a subset, these sums are as good as those shown in LS-paper in which results across 5 score functions are shown.

From these results, we can see that our score function is at least as good as existing score functions. Moreover, our score function is derived only from 24 protein data, while most score functions are derived from hundreds of protein data. From this, it is confirmed that the proposed method requires a small number of training data.

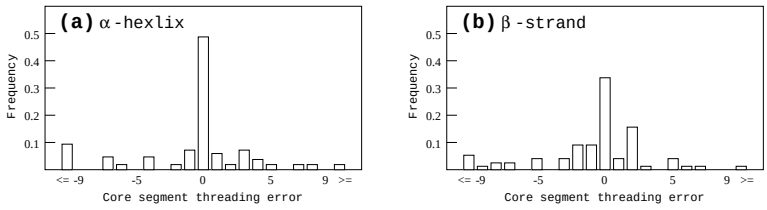

Figure 2: Frequencies of alignment errors between the optimal and the native threadings for each core segment across 23 protein data (89 α-helices and 77 β-strands). Error is computed as the optimal threading sequence index minus the native threading sequence index.

3.4 Score Functions Specialized for Structural Families

As noted in the above, the most important advantage of the proposed method over other methods is that it requires a small number of protein data. Using this, we may specialize a score function for each of structural families.

To ensure this property, we compare the score function obtained in the previous subsection with specialized score functions. We used the following data set A (α-up-down fold family) and B (Lipocalins fold family): $A = \{$ 2hmq(A), 256b(A), 1lpe, 2tmv(P), 2gmf(A), 2lig(A), 1bbh(A), 2asr, 2mhr $\}$ and $B = \{$ 1bbp(A), 1mup, 2hmb, 1mdc, 1opa(A), 1rbp, 1alb, 2ifb $\}$.

For each protein data $x \in A$ (resp. $x \in B$), we derived a (specialized) score function from $A - x$ (resp. $B - x$), and we examined self-threading for x using this score function. As in the previous subsection, 500 alternative threadings were generated per each protein data.

The results are summarized in Fig. 3 (α-up-down fold family) and Fig. 4 (Lipocalins fold family), where error distributions from α-helices (resp. β-strands) are shown in Fig. 3 (resp. Fig. 4) because most core segments are α-helices (resp. β-strands). In each figure, Fig. (a) shows the results obtained by using a general score function derived in the previous subsection, and Fig. (b) shows the results obtained by using specialized score functions. Note that in Fig. 4, only results for 1mdc, 1opa(A), 1rbp, 1alb, 2ifb are shown because the CPU time for searching an optimal threading for each of 1bbp(A), 1mup, 2hmb exceeds the time limit (4 hours).

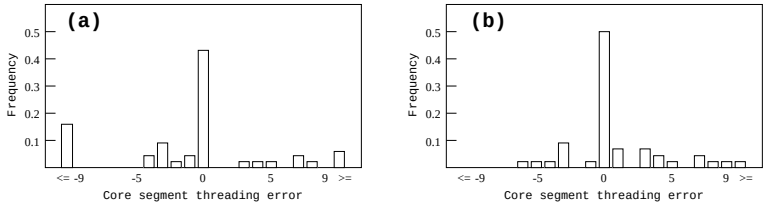

Figure 3: Frequencies of alignment errors across 9 proteins from the α-up-down fold family, where a general score function is used in (a) and specialized score functions are used in (b).

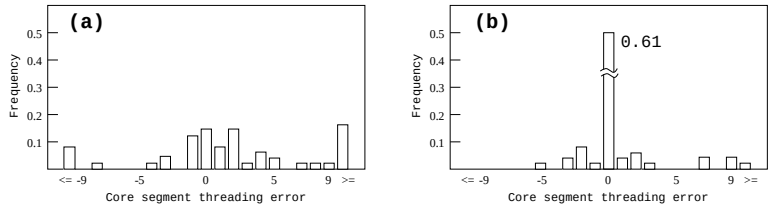

Figure 4: Frequencies of alignment errors for each core segment across 5 proteins from the Lipocalins fold family.

You can see that the results using specialized score functions are better than those using the general score function in both cases. You can also see that the difference is small in Fig. 3, whereas the difference is large in Fig. 4. The reason why there exists such a gap is probably that the sequences in data set B are more homologous than those in A.

Although we do not yet examine many families (because of the slow searching speed), the results suggest that the proposed method may be useful for deriving score functions specialized for structural families.

4 Conclusion

In this work, we proposed and evaluated a novel method of deriving a score function (a contact potential). In particular, we focused on a score function for computing nearly correct threadings.

As shown in the above, the proposed method has advantage over Boltzmann-like statistical methods [7,14,15,16] because the proposed method can derive a reasonable score function from fewer number of proteins. However, we have made computational experiments using small number of data, and the form (e.g., threshold, classification of environments, effect of loop regions) of a score function was not optimized. Thus, we are planning to make rigorous experiments using a large number of protein data and optimize the form of a score function.

Although similar methods have been already proposed, our method is different from them as described in Introduction. The most important difference is that all the generated constraints are always satisfied in our method if constraints can be satisfied, because we use an elaborated LP solver. However, for any method, it is not guaranteed that the derived score function is fully consistent with the training data (i.e., minimizing the scores of the native conformations or the native threadings). Indeed, in a context of threading with a pairwise contact potential, we have proved that deciding the existence of such score function was computationally hard [1]. However, errors produced by our method is one-sided. That is, if the LP solver outputs "there is no feasible solution satisfying all constraints", then this conclusion is always correct. Thus, if the LP solver outputs "no feasible solution", we can conclude that there does not exist a score function minimizing the scores of given native folds under the same condition, where condition consists of threshold of contact distance, classification of environment (α,β,exposed,...), etc. Our method might be useful to seek such conditions because the LP solver will output "no feasible solution" even for small number of input data if the condition is not appropriate.

Of course, it may be more important to discuss about score functions that approximately satisfy the constraints. Although we have proposed such a method (LP-B) in this paper, the results were not satisfactory. Indeed, when there were non-satisfied constraints, the derived score function was not good even if we used LP-B. Therefore, we should improve the method.

In this paper, we only examined self-threadings. In order to apply the

score function to the protein structure prediction, optimal threadings for at least 1000 templates [5] should be computed for each amino acid sequence with unknown structure. However, even if we use the elaborated search program [11], it will take very long time to compute 1000 optimal threadings. Therefore, in order to apply our score function to the protein structure prediction, we should develop much faster search algorithms.

Acknowledgments

This work was partially supported by the Grant-in-Aid for Scientific Research on Priority Areas, "Genome Science", of the Ministry of Education, Science, Sports and Culture in Japan.

References

1. T. Akutsu and M. Yagiura, *Technical Report of Information Processing Society of Japan* **AL-57**, 53 (1997).
2. F.C. Bernstein *et. al.*, *J. Mol. Biol.* **112**, 535 (1977).
3. J.U. Bowie, R. Lüthy and D. Eisenberg, *Science* **253**, 164 (1991).
4. S.H. Bryant and C.E. Lawrence, *PROTEINS: Structure, Function, and Genetics* **16**, 92 (1993).
5. C. Chothia, *Nature* **357**, 543 (1992).
6. D. Eisenberg, *Nature Structural Biology* **4**, 95 (1997)
(CASP2 homepage: http://predictioncenter.llnl.gov).
7. A. Godzik and J. Skolnick, *Proc. Natl. Acad. Sci. USA* **89**, 12098 (1992).
8. R. Goldstein, Z.A. Luthey-Schulten and P. Wholynes, *Proc. Natl. Acad. Sci. USA* **89**, 4918 (1992).
9. D.T. Jones, W.R. Taylor and J.M. Thornton, *Nature* **358**, 86 (1992).
10. N.K. Karmarkar, *Combinatorica* **4**, 373 (1984).
11. R.H. Lathrop and T.F. Smith, *J. Mol. Biol.* **255**, 641 (1996).
12. V.N. Maiorov and G.M. Crippen, *J. Mol. Biol.* **227**, 876 (1992).
13. L.A. Mirny and E.I. Shakhnovich, *J. Mol. Biol.* **264**, 1164 (1996).
14. S. Miyazawa and R.L. Jernigan, *Macromolecules* **18**, 538 (1985).
15. K. Nishikawa and Y. Matsuo, *Protein Engineering* **6**, 811 (1993).
16. M.J. Sippl, *J. Mol. Biol.* **213**, 859 (1990).
17. R.J. Vanderbei, *LOQO User's Manual*, (Princeton University, 1992).
18. R.J. Vanderbei, *Linear Programming. Foundations and Extensions* (Kluwer Academic Publishers, Boston, 1996).