Flexible Alignments for Protein Threading

Guillaume Collet, Rumen Andonov, Nicola Yanev, Jean-François Gibrat

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Flexible Alignments for Protein Threading

Guillaume Collet — Rumen Andonov — Nikola Yanev — Jean-François Gibrat

N° 6808

Janvier 2009

Thème BIO

ISSN 0249-6399 ISRN INRIA/RR--6808--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
RENNES - BRETAGNE ATLANTIQUE

Flexible Alignments for Protein Threading

Guillaume Collet , Rumen Andonov , Nikola Yanev , Jean-François Gibrat

Thème BIO — Systèmes biologiques
Équipes-Projets Symbiose

Rapport de recherche n° 6808 — Janvier 2009 — 12 pages

Abstract: We present a new local alignment method for the protein threading problem. Local sequence-sequence alignments are widely used to find functionally important regions in families of proteins. However, to the best of our knowledge, no local sequence-structure alignment algorithm has been described in the literature. Here we model local alignments as Mixed Integer Programming (MIP) models. These models permit to align a part of a protein structure onto a protein sequence in order to detect local similarities. The paper describes two MIP models, compares and analyzes their performance by using ILOG CPLEX 10 solver.

Key-words: Integer programming, combinatorial optimization, protein threading problem, protein structure alignment

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Alignements flexibles pour la reconnaissance de repliements

Résumé : Nous proposons une méthode d'alignement local pour la reconnaissance de repliement des protéines. Les alignements locaux de séquences sont largement utilisés afin de découvrir des régions conservées au sein des familles protéines. Néanmoins, nous n'avons pas connaissance d'algorithmes d'alignement local entre des séquences et des structures de protéine. Nous modélisons les alignements locaux sous la forme de modèle MIP (Mixed Integer Programming). Ces modèles permettent d'aligner une portion d'une structure de protéine avec une séquence de protéine afin de détecter des similarités locales. Dans ce rapport, nous proposons deux modèles MIP dont les performances sont comparées et analysées en utilisant le solveur ILOG CPLEX 10.

Mots-clés : Programmation en nombres entiers, optimisation combinatoire, reconnaissance de repliements, alignement de protines

1 Introduction

The amount of genomic data available, thanks to the steady improvement of DNA sequencing techniques, is exponentially increasing. Genomic and metagenomic projects will, in the near future, generate millions of protein sequences for which nothing but the amino acid sequence is known. One of the goals of these projects is to determine the function of the corresponding proteins. To exploit this avalanche of new data, biologists, to a large extent, have to rely on *in silico* methods.

The most important *in silico* methods are based on the concept of homology. Two proteins are homologous if they descend from a common ancestor. The ancestor protein was characterized by a given sequence, a three-dimensional (3D) structure and a function. Descendant proteins have inherited these three features. However, the sequences can have undergone a variable amount of mutations, insertions and deletions and may or may not still be recognizably similar. The 3D structures, on the other hand, show a better conservation than the sequences and, except for local differences, have remained globally similar. Finally, the ancestor function may have been conserved by its descendants. The principle of homology-based analysis method is to identify a homology relationship between a new protein and a protein whose function is known. This permits, with due care, to suggest a function for the new protein.

Detecting homology relationships when the sequences are sufficiently similar is relatively straightforward. To show that two proteins are evolutionary related one uses sequence alignment methods based on dynamic programming techniques. Suitable modifications of the fundamental algorithm enable the user to perform global, semi-global and local alignments (see Fig.1).

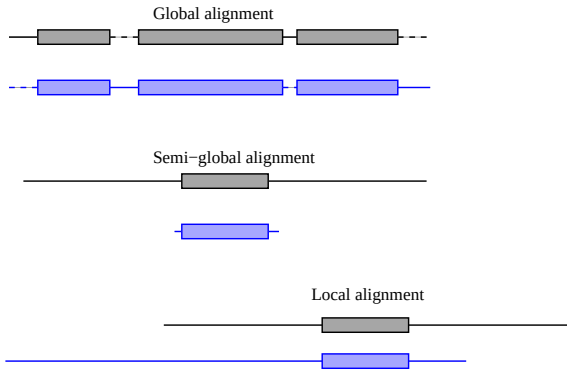

Figure 1: When aligning protein sequences it is important to be able to carry out different types of alignments according to the situation encountered. When aligning proteins belonging to the same family global alignments are used, i.e., gaps at both ends of the alignments are penalized. On the other hand, when one looks for a protein domain in a longer sequence, semi-global alignments are required. When performing semi-global alignments gaps at both ends of the alignment are not penalized. The most general type of alignment is the local alignment where only substrings of the sequences are aligned (this might correspond to a common domain found in two proteins that are otherwise different).

For remote homologs, i.e., homologous proteins for which the sequences are no longer similar, sequence alignment methods fail. One, then, must resort to another property of homologous protein, namely the good conservation of the 3D structures. In such a case, to figure out a homology relationship between 2 proteins, one aligns the sequence of the new protein with the 3D structures of known proteins. Such methods are called fold recognition methods or threading methods.

For a long time, threading methods *using non local parameters* (see the next section) suffered from the lack of a rigorous method capable of determining the sequence – structure alignment with the optimal score. They relied on heuristic techniques, e.g. stochastic techniques such as a Gibbs Monte Carlo. Lathrop & Smith [1] were the first to propose an algorithm based on a branch & bound technique providing the global alignment with the optimal score, although their algorithm had a limited efficiency. Since then, other methods have been developed that improved the efficiency of the sequence – structure alignment algorithm [2, 3, 4] and provided semi-global alignment algorithm ([5]).

This paper describes a new algorithm that expands upon algorithms proposed in previous works [3, 4, 5] to allow implementation of *local* sequence – structure alignments. This allows threading methods to cover the whole spectrum of alignment types needed to analyze homologous proteins. This participates to our effort to make the threading methods as versatile and easy to use as sequence alignment methods.

2 Outline of the Protein Threading Problem

2.1 Definition of alignments

Our definition of alignments is based on the definition of the Protein Threading Problem (PTP) given in [1].

Query Sequence A query sequence is a string of length N over the 20-letter amino acid alphabet. This is the amino acid sequence of a protein with unknown structure.

Structure Template All current threading methods replace the 3D coordinates of the known structure by an abstract template description in terms of blocks, neighbor relationships, distances, environments, etc (see Fig 2). We consider that a structure template is an ordered set M of m blocks. Block k has a fixed length of L_k amino acids.

Blocks correspond to the most conserved parts of the structure, usually the secondary structure elements (SSEs : α-helices and β-strands). Let $I \subseteq \{(k,l) \mid 1 \leq k < l \leq m\}$ be the set of blocks interactions. The graph with set of vertices M and set of edges I is called the generalized contact map graph.

Alignments The alignment of a sequence with a 3D structure can be described as positioning the blocks along the sequence. Such an alignment is called feasible if the blocks preserve their original order and do not overlap (see Fig 3).

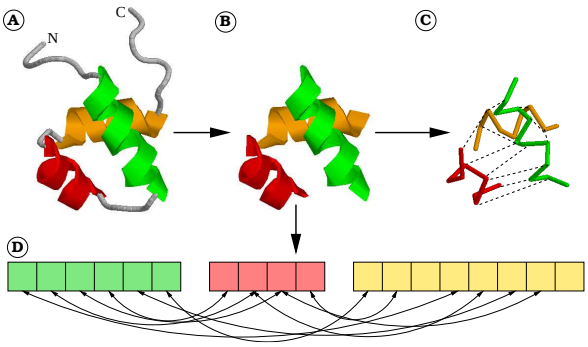

Figure 2: A) "ribbon" representation of a protein with 3 α helices, B) only the secondary structure elements, here the 3 helices, are taken into consideration, C) dotted lines represent interactions between positions in the helices, i.e., residues in contact. It must be noted that these interactions involve pairs of positions in the structure. This property prevents the use of dynamic programming techniques to align the sequence with the 3D structure, D) block representation of the protein, double headed arrows correspond to the dotted line in C. In the green, red and orange blocks respectively 6, 4 and 8 amino acids can be aligned. The score of a given alignment is the sum, over all the defined interactions, of the corresponding amino acid pair parameters.

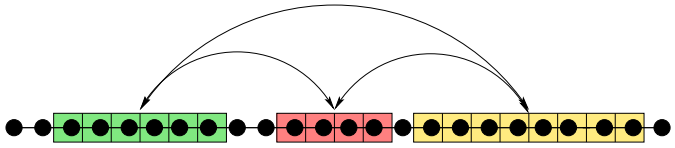

Figure 3: Example of alignment of query sequence of length $N = 24$ and template containing $m = 3$ blocks of lengths $L_1 = 6$, $L_2 = 4$ and $L_3 = 8$. An arrow is drawn between 2 blocks if there is at least one position in each block which are in interaction in the 3D structure. Thus the corresponding generalized contact map is: $I = \{(1,2),(1,3),(2,3)\}$

An alignment is completely determined by the starting positions of all blocks along the sequence. In fact, in the classical PTP, it is more convenient to use relative positions instead of absolute positions. It can be shown that all blocks have the same set of relative positions that vary between 1 and $n = N - \sum_{k \in M}(L_k) + 1$.

The number of possible alignments (the search space size of PTP) is given by $|\mathcal{T}| = \binom{m+n-1}{m}$ [1], which is a huge number even for small instances (for example, if $m = 20$ and $n = 100$ then $|\mathcal{T}| \approx 2.5 \times 10^{22}$).

With this definition of the PTP, gaps are not allowed within blocks. They are confined to loops joining the blocks. The biological justification for this feature is that blocks, being conserved regions of the 3D structure, the probability of insertion or deletion within them is very small.

2.2 Network flow formulation

This section is a summary of [3, 4]. In order to develop appropriate mathematical models, PTP is reformulated as a network optimization problem. Let $G(V, A)$ be a digraph with vertex set V and arc set A. The vertex set V is organized in columns as shown in Fig 4. In each column, each vertex corresponds to a relative position of the corresponding block along the sequence. A cost C_{ik} is associated to each vertex (i, k) as defined by the scoring function. C_{ik} is a local score that depends only on the location of the block along the sequence. To each arc $((i, k), (j, l))$ is associated a cost D_{ikjl} as defined by the scoring function. D_{ikjl} is a non local score that depends on the location of blocks k and l along the sequence and the set of interacting positions in these 2 blocks.

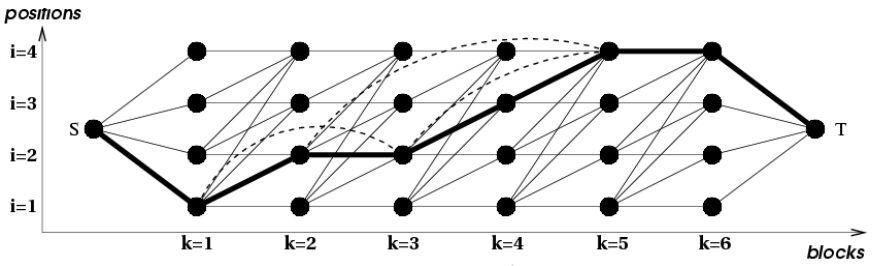

Figure 4: Example of alignment graph. The thick line path corresponds to the threading in which the relative positions of blocks are 1,2,2,3,4,4. Dashed line edges represent non local interactions where the set of block interactions is $I = \{(1, 3), (2, 5), (3, 5)\}$.

Adding two extra vertices S and T and the corresponding arcss $(S, (i, 1))$, $i = 1, ..., n$ and $((j, m), T)$, $j = 1, ..., n$, illustrates the one-to-one correspondence between the set of feasible threadings and the set of S-T paths in G. We say that a S-T path *activates* its vertices. An arc is *activated* by a S-T path if both ends are on the path. We call the subgraph induced by the activated arcs of an S-T path an *augmented path*. Then PTP is equivalent to finding the longest augmented path in G.

2.3 Integer programming formulation

Let y_{ik} be binary variables associated with vertices in the previous network. Then y_{ik} is 1 if block k is at position i and 0 otherwise (vertex (i, k) is activated or not). Let Y be the polytope defined by the following constraints:

$$\sum_{i=1}^{n} y_{ik} = 1 \qquad k \in [1, m] \tag{1}$$

$$\sum_{j=1}^{i} y_{j(k+1)} - \sum_{j=1}^{i} y_{jk} \leq 0 \qquad k \in [1, m-1],\ i \in [1, n-1] \tag{2}$$

$$y_{ik} \in \{0, 1\} \qquad k \in [1, m],\ i \in [1, n] \tag{3}$$

To take into account the interaction costs, we introduce a second set of variables $0 \leq z_{ikjl} \leq 1$, with $(k, l) \in I$ and $1 \leq i \leq j \leq n$. The variable z_{ikjl}

is set to 1 if the corresponding arc is activated. Then, we define the following constraints:

$$y_{jl} = \sum_{i=1}^{j} z_{ikjl} \qquad (k,l) \in I,\ j \in [1,n] \tag{4}$$

$$y_{ik} = \sum_{j=i}^{n} z_{ikjl} \qquad (k,l) \in I,\ i \in [1,n] \tag{5}$$

$$0 \leq z_a \leq 1 \qquad a \in A \tag{6}$$

Finding the longest augmented path in graph G (i.e. solving PTP) is then equivalent to optimizing the following function subject to the above constraints:

$$\max \sum_{k=1}^{m} \sum_{i=1}^{n} C_{ik} y_{ik} + \sum_{a \in A} D_a z_a \tag{7}$$

This model, known as MYZ, was first introduced in [3] where it was shown to outperform the MIP model used in the RAPTOR package [2] for all large PTP instances. Both models (MYZ and RAPTOR) are solved by first relaxing constraints (3), i.e., letting y_{ik} be real variables such that $0 \leq y_{ik} \leq 1$. The relaxed problem is solved by a linear programming (LP) technique. The LP solution is then used as a lower bound in a subsequent branch & bound algorithm that finds the integer solution.

3 Flexible alignments : towards better PTP models

3.1 Definition

In the previous section, we assumed that SSEs correspond to conserved regions of the 3D structure. Constraints (1) force each block to be aligned with the query sequence. However, sometimes it happens that not all members of a protein family share exactly the same number of SSEs. Forcing all blocks to align with the sequence leads to spurious alignments with bad scores that sometimes prevent the method to detect remote homologs. To tackle this issue, we developed a new alignment method, called flexible, which permits to omit blocks from the alignment if such omissions improve the score. Extended abstract of our approach was first presented in [6]. In this paper we describe in details the mathematical models and we compare them computationally.

Because an unspecified number of blocks can be disregarded when a flexible alignment is carried out, each block can, potentially, be aligned with the whole sequence. It follows that relative positions are no longer meaningful. With flexible alignments, block k takes $n_k = N - L_k + 1$ absolute positions along the sequence, where L_k denotes the length of block k. This leads to columns having different heights in the network flow and the need to make use of an "offset" to move from a column to the next. (this offset is illustrated in Fig 5).

Definition 1 *A **feasible path** is an arbitrary sequence $(i_1,k_1),(i_2,k_2),\ldots,(i_t,k_t)$ Publier such that $i_j \leq i_{j+1} - L_{k_j}$ and $k_j < k_{j+1}$ for $j \in [1,m-1]$.*

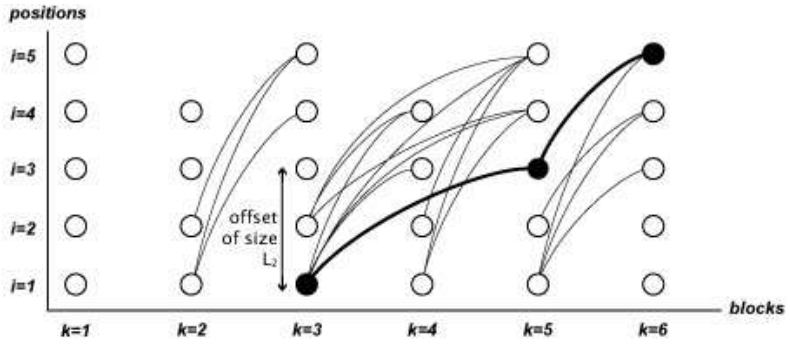

Figure 5: Example of flexible alignment graph with five blocks. Blocks' sizes are respectively : 2,3,2,3,2,2. The path in thick lines corresponds to the threading in which: blocks 1, 2, and 4 are omitted, and blocks 3, 5, and 6 are on positions 1, 3 and 5. All arcs from A are represented. Because of absolute positions, notice that a offset of size L_k is needed to go from column k to another.

In order to implement the mechanism for omitting blocks during an alignment, two concurrent schemes have been proposed:

1. we modify constraints (1) in order to allow for block omission.
2. N dummy vertices are added in every column. If a dummy vertex is activated in a column, the corresponding block is omitted.

In the second scheme, two sorts of vertices exist : real (R) and dummy (D). This generates 4 types of subpaths: R to R , R to D, D to R, and D to D (as shown in fig 6).

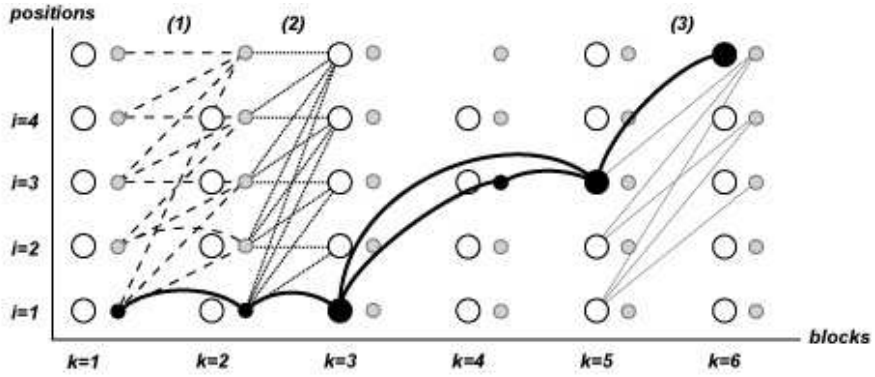

Figure 6: Example of a flexible alignment graph with dummy vertices (small gray circles). This illustrates the same alignment as in figure 5 but not all arcs are represented for clarity. (1) dashed lines represent D to D paths, (2) dotted lines represent D to R paths, and (3) gray lines represent R to D paths. Notice that an offset of size L_k is needed to go from a real vertex ik to the next (be it dummy or real). No offset is needed when moving from a dummy vertex.

3.2 Mathematical models

According to the above two schemes, we implemented 2 models : Compact Model (CM) for the first one and Extended Model (EM) that uses dummy nodes.

3.3 Compact Model

The objective function is given by (7). This model uses constraints (3) and (6) together with the additional constraints:

$$\sum_{i=1}^{n_k} y_{ik} \leq 1, \qquad k \in [1,m] \tag{8}$$

$$\sum_{j=i+L_k}^{n_l} z_{ikjl} - y_{ik} \leq 0, \qquad \begin{cases} (k,l) \in I, \\ i \in [1,n_k] \end{cases} \tag{9}$$

$$\sum_{i=1}^{\min(j-L_k,n_k)} z_{ikjl} - y_{jl} \leq 0, \qquad \begin{cases} (k,l) \in I, \\ j \in [1,n_l] \end{cases} \tag{10}$$

$$y_{ik} + \sum_{j=1}^{min(n_k,i+L_k-1)} y_{jl} \leq 1, \qquad \begin{cases} 1 \leq k \leq l \leq m, \\ i \in [1,n_k] \end{cases} \tag{11}$$

$$\sum_{i=1}^{n_k} y_{ik} + \sum_{i=1}^{n_l} y_{il} - \sum_{j=L_k+1}^{n_l} \sum_{i=1}^{j-L_k} z_{ikjl} \leq 1, \qquad (k,l) \in I \tag{12}$$

Constraints (8), (9) and (10) correspond to constraints (1), (4) and (5). They have the same effect with the difference that (8), (9) and (10) are inequalities (there is one or zero vertex activated in a column; there is one or zero arc leaving an activated vertex; there is one or zero arc entering an activated vertex). Constraints (11) have the same effect as (2). Finaly, constraints (12), coerce the activation of an arc if its vertices are activated.

3.4 Extended Model

Denote by $d_{ik}, i \in [1,N], k \in [1,m]$ the so called dummy variables. The objective function is again given by (7). This model uses constraints (3), (6) from MYZ model and it also uses constraints (9), (10) and (12) from the Compact Model. Additional constraints are the following:

$$\sum_{i=1}^{n_k} y_{ik} + \sum_{i=1}^{n_k} d_{ik} = 1 \qquad k \in M \tag{13}$$

$$\sum_{i=1}^{j} d_{ik} + \sum_{i=1}^{min(j,n_k)} y_{ik} - \sum_{i=1}^{j} d_{ik-1} - \sum_{i=1}^{j-L_{k-1}} y_{i(k-1)} \leq 0 \quad \begin{cases} k \in [2,m] \\ j \in N \end{cases} \tag{14}$$

$$d_{ik} \in \{0,1\} \quad \begin{cases} k \in M \\ i \in [1,n_k] \end{cases} \tag{15}$$

Constraints (13) state that exactly one vertex (either real or dummy), must be activated in a column (these constraints are similar to (1) in MYZ model).

Constraints (14) replace constraints (2) from MYZ. They preserve the order of the blocks.

4 Computational Results

4.1 Benchmark

To compare the performances of our mathematical models, we created a benchmark based on the ASTRAL and SCOP databases [7, 8]. We selected 3 classes (all-alpha, all-beta and alpha-beta) from the top of SCOP hierarchy. Then in each class, we selected two folds. Here is a summary of our benchmark :

- All-α class
  1. *DNA/RNA binding 3 helical bundle* (SCOP : 46,688, 81 families)
  2. 4 *helical up and down bundle* (SCOP : 47,161, 12 families)
- All-β class
  1. *Immunoglobuline like beta sandwich* (SCOP : 48,725, 49 families)
  2. *Double stranded β helix* (SCOP : 51,181, 29 families)
- α-β class
  1. *TIM $\alpha\beta$ barrel* (SCOP : 51,350, 84 families)
  2. *NADP binding Rosmann* (SCOP : 51,734, 29 families)

We used AEROSPACIE scores [7] to select the best protein domains representative of the 284 families from the above 6 fold classes. We only kept structures having a size between 50 and 300 residues without prosthetic groups and no disulfide bridge. This led to a template database of 217 protein domains.

We selected 30 query proteins from this database making sure that each query had at least one similar domain in the template database. We performed the alignment of these 30 query proteins with the template database.

4.2 Running times

In this section, we compare the results obtained by the models defined in section 3.2. These models are solved with the CPLEX 10.0 package from the Ilog company on a cluster of computers each one possessing two AMD Opteron 64a processors and 4 Go of RAM.

Figure 7 shows that EM is faster than CM for 94% of the instances. If we consider computation time greater than 1 sec, EM is faster for 99% of the cases.

4.3 Relative gap between LP solution and the integer solution

As explained in section 2.3 the models are solved by i) letting the y_{ik} variables assume real values between 0 and 1, ii) solving the corresponding relaxed problem with a linear programming technique then iii) employing the solution just

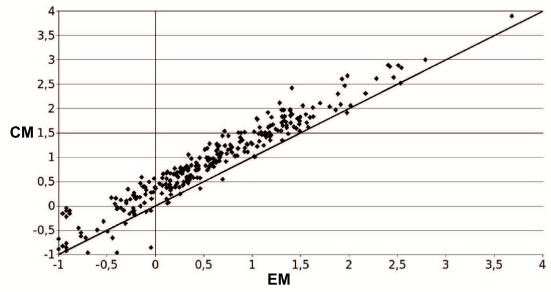

Figure 7: Comparison of the computation times obtained by EM (on the x axis) and CM (on the y axis). Each point represents an alignment. Times are expressed in seconds and are plotted using a base 10 logarithm scale.

found as a lower bound for a branch & bound algorithm whose purpose is to find the optimal integer solution. Linear programming is a time polynomial technique therefore the time intensive part of this algorithm is the branch & bound technique. The relative gap (RG) between the solution of the relaxed problem (LP) and the optimal solution (OPT) is defined as: $RG = \frac{LP-OPT}{OPT}$. It is a good indicator of the efficiency of the model since the smaller RG, the easier for the branch & bound algorithm to find the solution. The optimal case is obtained when $RG = 0$, i.e., when the LP algorithm ends up at a vertex of the polytope and thus provides directly the integer solution.

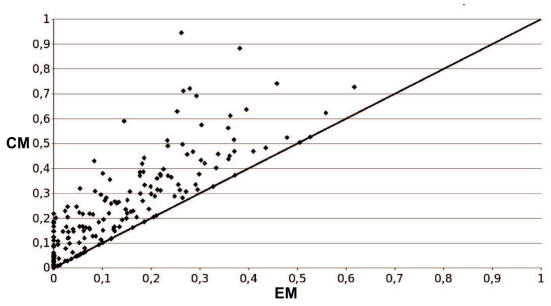

Figure 8: Comparisons of relative gaps ($\frac{LP-OPT}{OPT}$) between models EM and CM. Each point is a sequence – structure alignment. The EM model always results in a smaller relative gap than the CM model.

Figure 8 shows that EM always gives a smaller RG than CM. It must be noted that the LP method find directly the integer solution in 41% of the cases for the CM model and 52% of the cases for the EM model.

Three factors exert an influence on the efficiency of a model : the number of variables, the number of constraints and the type of constraints, e.g., problems with equality constraints are easier to solve than problems with inequality constraints using the LP algorithm. EM presents two differences with CM: it has a smaller number of constraints and more constraints are of the "equality" type. This is a likely explanation for its better efficiency.

5 Conclusion

This paper describes a new method for aligning protein sequences and structures. We propose flexible alignments that give the possibility of omitting parts of the 3D structure that might not be conserved in remote homolog proteins. These alignments have been modelled in terms of integer programming and tested with the CPLEX 10 solver. The performed experiments prove that our algorithm is indeed capable of omitting blocks when necessary to improve the objective function. However, the accuracy of the obtained alignments strongly depends on the quality of the score functions. Our results show that the score functions must be conveniently adapted to the specificity of the flexible algorithm. This a subject of our current research. We are also working on a dedicated algorithm in order to improve the performance obtained by CPLEX 10 solver.

References

[1] R.H. Lathrop. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Engineering*, 255:1059–1068, 1994.

[2] J. Xu, M. Li, G. Lin, D. Kim, and Y. Xu. Raptor: optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–118, 2003.

[3] R. Andonov, S. Balev, and N. Yanev. Protein threading problem: From mathematical models to parallel implementations. *INFORMS Journal on Computing*, 16(4):393–405, 2004. Special Issue on Computational Molecular Biology/Bioinformatics, Eds. H. Greenberg, D. Gusfield, Y. Xu, W. Hart, M. Vingro.

[4] N. Yanev and R. Andonov. Parallel divide&conquer approach for the protein threading problem. *Concurrency and Computation: Practice and Experience*, 16:961–974, 2004.

[5] R. Andonov, G. Collet, J-F. Gibrat, A. Marin, V. Poirriez, and N. Yanev. Recent advances in solving the protein threading problem. In E-G. Talbi and A. Zomaya, editors, *Grids for Bioinformatics and Computational Biology*, chapter 14, pages 325–356. Wiley-Interscience, 2007.

[6] G. Collet, N. Yanev, A. Marin, R. Andonov, and J-F. Gibrat. A flexible model for protein fold recognition. In A. Denise, P. Durrens, S. Robin, E. Rocha, A. de Daruvar, and A. Groppi, editors, *Septièmes Journées Ouvertes de Biologie, Informatique et Mathématiques (JOBIM)*, pages 215–216, 2006.

[7] J.M. Chandonia, M. Levitt, and S. Brenner. Astral compendium in 2004. 32:189–192, 2004.

[8] A. Andreeva, D. Howorth, S.E. Brenner, T.J.P. Hubbard, C. Chothia, and A.G. Murzin. Scop database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Research*, 32:226–229, 2004.

Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399