Continuous Optimization Methods for Structure Alignments *

R. Andreani † J. M. Martínez ‡ L. Martínez § F. Yano ¶

January 30, 2006

Abstract

Structural Alignment is an important tool for fold identification of proteins, structural screening on ligand databases, pharmacophore identification and other applications. In the general case, the optimization problem of superimposing two structures is nonsmooth and nonconvex, so that most popular methods are heuristic and do not employ derivative information. Usually, these methods do not admit convergence theories of practical significance. In this work it is shown that the optimization of the superposition of two structures may be addressed using continuous smooth minimization. It is proved that, using a Low Order-Value Optimization approach, the nonsmoothness may be essentially ignored and classical optimization algorithms may be used. Within this context, a Gauss-Newton method is introduced for structural alignments incorporating (or not) transformations (as flexibility) on the structures. Convergence theorems are provided and practical aspects of implementation are described. Numerical experiments suggest that the Gauss-Newton methodology is competitive with state-of-the-art algorithms for protein alignment both in terms of quality and speed. Additional experiments on binding site identification, ligand and cofactor alignments illustrate the generality of this approach.

Key words: Structural alignment, Protein alignment, Gauss-Newton method, Continuous optimization, Order-Value Optimization.

1 Introduction

We are concerned with the comparison of three-dimensional shapes, particularly biomolecules and related chemical structures. These problems arise in Chemistry and Biology as far as an increasing number of molecular and biomolecular structures are discovered, claiming for their classification and comparison with respect to their three-dimensional folds.

*Technical Report MCDO 060124, Department of Applied Mathematics, State University of Campinas, Brazil, October 2005. This work was supported by PRONEX-Optimization 76.79.1008-00, FAPESP (Grants 01-04597-4 and 02-14203-6) and CNPq.

†Department of Applied Mathematics, IMECC-UNICAMP, State University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil. E-mail: andreani@ime.unicamp.br

‡Department of Applied Mathematics, IMECC-UNICAMP, State University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil. E-mail: martinez@ime.unicamp.br

§Institute of Chemistry, State University of Campinas, E-mail: lmartinez@iqm.unicamp.br.

¶Department of Applied Mathematics, IMECC-UNICAMP, State University of Campinas. E-mail: yano@ime.unicamp.br

The most natural way to compare structures is to superimpose them in some optimal manner, looking for their similarities and discrepancies after superposition.

Rigid structural alignments can be classified in three main groups according to the knowledge of the relationship between the points (from now on called "atoms") that form the structure:

1. The atoms of one structure have a known correspondence with the atoms of the second structure.

2. The correspondence is not known, but there are restrictions on the postulated bijection.

3. The correspondence is not known and there are no restrictions to its determination.

The first case has a well known analytical solution and, therefore, represents no challenge [11, 12]. This is the alignment that is usually required when two molecules sharing the same atoms but different structures are aligned and occurs when one compares two structures of the same protein determined experimentally or two very similar proteins, for which a correspondence between the amino acids can be deduced from the comparison of their sequences.

The second case is much more challenging and is very relevant for comparison of protein structures. Proteins are chains of α-amino acids bound by peptide bonds. The structure of a protein "starts" at one amino acid (defined, by convention, as the N-terminal amino acid) and "ends" at the amino acid which is farther from the first one in terms of its position in the chain (C-terminal amino acid). Therefore, there exists a natural sequence for the amino acid chain and, consequently, the alignment must preserve order and be free of gaps as much as possible. Since there is an analytical solution for any known correspondence, the restrictions on the possible bijections simplify the problem. Several methods for protein structural comparison rely on these restrictions, or penalize its lack of fulfillment. The exploitation of constraint information usually provides these methods with improved speed. Restricting the space of correspondences, however, is a simplification that may prevent the algorithms from finding novel structural similarities, particularly those which cannot be identified by the simple comparison of the sequence of amino acids. Moreover, algorithms strongly based on these constraints cannot be used for general structural comparisons (even of proteins) when they include cofactors (other molecules that do not belong to the amino acid chain but are bound to it), or for the comparison of non-proteic structures (i.e. ligand databases) in which sequentiality does not appear at all. Finally, the algorithms that rely on the atom sequence are not robust in terms of the input received. This means that the algorithms may fail to find the alignment if the atoms are scrambled.

The third and more general case, in which no information is known on the correspondence between atoms, is the general problem we deal in this work. This is the most challenging problem in structural alignment, because the number of possible correspondences is too large for every one to be tested. This general problem applies to the comparison of general molecular structures, including proteins as a particular case, ligands, or for other problems such as image comparison in 2D or 3D. Our focus in this work is on the comparison of molecular structures, being Protein Alignment the best studied particular case.

The complexity of Structural Alignment problems increases if the structures have internal degrees of freedom or may suffer non-rigid transformations. In these cases, not even the simplest alignment (where correspondence between atoms is known) has an analytical solution. Some

algorithms deal with non-rigidity, but most of them are ultimately based on the analytical solution of the fixed correspondence rigid alignment. The algorithms proposed in this paper, on the other hand, naturally incorporate flexibility or any other transformation on the structures, being the complexity of the problem increased only by the larger number of local minima and practical aspects of implementation.

1.1 Protein Alignment

Molecular structures are represented by the 3D coordinates of its atoms. The structure of proteins, in particular, may be represented in a simplified way, by the 3D coordinates of the Cα atom of each amino acid. This representation captures the main features of the three-dimensional arrangements of amino acids in the protein structure.

The Protein Data Bank (http://www.rcsb.org/pdb/) [4] contains information for the structure of more than 30 thousand proteins. An alignment between two proteins $A = (A_1, \ldots, A_{n_A})$ and $B = (B_1, \ldots, B_{n_B})$ is a one-to-one correspondence Φ between a subset of atoms of A and a subset of atoms of B such that:

1. The domain and the range of Φ must be *similar* and should be as large as possible.
2. The *Order-Preserving property* OPP ($i < j \Rightarrow \Phi(i) < \Phi(j)$) is generally (but not always) desirable.
3. The number of *gaps* (cases in which $k = \Phi(i)$ and the first $j > i$ belonging to the domain is such that $j > i + 1$ or $\Phi(j) \neq \Phi(i) + 1$) should be, in general, small.

Structural similarity between two proteins may indicate that they are involved with similar functions. Moreover, since structural similarity is conserved more than sequence similarity, Protein Alignment gives powerful clues for looking back to evolutionary history [10, 15].

The first requirement for Protein Alignment (similarity and large bijection) must be always preserved but the next two should be relaxed. Frequently, when a good alignment between proteins is found satisfying the first requirement, the second and third requirements are (almost) fulfilled for free. However, some alignments between proteins may satisfy the first requirement (especially *similarity*) but not the remaining two.

1.2 Other Alignments of Biomolecular Relevance

In some Protein Alignment problems the OPP and the few-gaps requirement may be relaxed. Moreover, in different important alignment problems in Chemistry, those two requirements do not need to be satisfied at all. Some examples are given below.

1. Binding site identification: A pharmacophore is a group of amino acids that are involved in direct interactions with some protein ligand (a drug, for example). The pharmacophore is usually composed by amino acids from very different positions in the protein and, therefore, no information on the sequence may be used for identification and alignment. Furthermore, it may be interesting to use the full amino acid structure (not only the Cα atoms) and, perhaps, allow for the rotation of the amino acids' side chains.

2. Ligand screening: Ligands are small molecules (usually of 30 to 100 atoms) that bind proteins specifically and are potential drug candidates. One possible screening of ligand databases consists in searching for other molecular structures that are similar to a known drug or natural ligand, in such a way that new molecules with pharmacological value may be identified. Ligand structures contain no sequence information and are usually highly flexible.

3. Alignment of protein structures including cofactor: Proteins are frequently bound to other structures, such as drugs, DNA or other proteins. The comparison of macromolecular structures could use some information on the sequentiality of the internal protein components, but there is no sequentiality associated to the whole arrangement (which may contain, for example, other protein, ligand, DNA, etc.).

1.3 Measures of Similarity and Protein-Alignment Methods

For measuring similarity between two structures one needs to translate and rotate one of them in such a way that, in some sense, both are *superimposed*. Given two superimposed proteins, the best alignment relative to a separable scoring function can be found analyzing paths in the matrix of Protein-to-Protein distances [19]. On the other hand, given the bijection Φ, the best superposition associated to a rigid movement can be found analytically solving the Procrustes problem [7, 11, 12].

Some methods for Protein Alignment (DALI [9, 8], SAP [29, 28], CE [26], FAST [33] and the URMS-based algorithm introduced in [13]) seek directly a good correspondence based on internal characteristics of the proteins (for instance, internal distance matrices) and, thus, compute the optimal superposition only at the end of the analysis. MAMMOTH [23, 18] uses URMS to find a superposition and, then, uses heuristics, scores and enlargements to find a sufficiently large correspondence. Other methods (STRUCTAL [27], LSQMAN [14], SSM [17]) try to find an adequate correspondence through successive superposition trials. Given a tentative bijection Φ, a superposition that optimizes the matching distances is computed using Procrustes. Using the superposition so far obtained, a new bijection is computed and the process continues iteratively until a convergence criterion is satisfied. These methods differ in the way in which the one-to-one correspondence Φ is computed at every iteration: STRUCTAL maximizes a score function using dynamic programming, SSM uses the secondary structure of the proteins for finding a suitable bijection and LSQMAN uses an heuristic based on matching small-distance residuals and mapping enlargement. A comprehensive evaluation of these methods may be found in [15]. Given a required precision $\varepsilon > 0$, Kolodny and Linial [16] introduced a method for solving the Protein Alignment problem whose complexity is polynomial in the number of atoms. The idea is to use an exhaustive ε-search in the space of rotations and to exploit the polynomiality of the dynamic programming procedure to optimize a score function. Although this is not a practical method, it sheds light on complexity issues.

None of the practical methods mentioned above include the possibility of internal rotations. (The method of Kolodny and Linial may be trivially extended to the case with internal rotations preserving polynomiality.) In other words, the movement that allows one to superimpose the proteins is assumed to be rigid in all the cases.

Our approach is to rely on the evaluation of superpositions (translations and rotations) by means of continuous optimization tools. A superposition is given by a translation and several angles. Three angles correspond to a rigid movement and the remaining ones correspond to internal rotations. A natural idea is to attribute a functional value to the superposition and to optimize the corresponding function by means of a suitable smooth minimization algorithm. The difficulty is that natural similarity functions may be nonsmooth (perhaps non-continuous) and very difficult to evaluate. Here we claim that, using the proper *order-value based* merit definition, the employment of smooth unconstrained optimization for structural alignment is quite reliable. Nonsmoothness of the objective function remains but, in this case, lack of differentiability is *benign*, in the sense that we may define smooth methods that preserve well-definiteness and global convergence.

In Section 2 we describe the mathematical model used for addressing the similarity problem and we prove that the merit function may be viewed as the minimum of a set of smooth functions. In Section 3 we define a Gauss-Newton-like algorithm for minimizing the similarity function defined in the previous section. Also in this section, we prove global convergence to critical points. In Section 4 we show how to find initial approximations and we give algorithmic details. In Section 5 we provide numerical experiments comparing the Gauss-Newton approach to the popular DALI alignment algorithm. Conclusions are given in Section 6.

Notation.

$\#C$ denotes the cardinality of the set C.
$\|\cdot\| = \|\cdot\|_2$
$I\!N = \{0, 1, 2, \ldots\}$

2 Mathematical Formulation

Structures A and B are identified here by the sets of points (atoms) $A_1, \ldots, A_{n_A}$ and $B_1, \ldots, B_{n_B}$ in the three-dimensional Euclidian space $I\!R^3$. Associated to Structure A we have n_{ax} rotation axes ($n_{ax} \leq n_A - 1$), determined by n_{ax} different pairs of consecutive atoms

$$(A_{iax(1)}, A_{iax(1)+1}), \ldots, (A_{iax(n_{ax})}, A_{iax(n_{ax})+1}).$$

A *movement* T is defined by $(3+n_{ax})$ angles $\alpha_1, \ldots, \alpha_3, \alpha_4, \ldots, \alpha_{3+n_{ax}}$ and three displacement parameters $a = (a_1, a_2, a_3)$. The set of points $T(A_1), \ldots, T(A_{n_A})$ is denoted $T(A)$. For obtaining $T(A)$ we perform, successively, the following transformations on the atoms of A:

1. $T_1(A)$ is the result of applying a rotation with angle α_1 around the x-axis to all the atoms of A. Analogously, $T_2(A)$ comes from applying a rotation with angle α_2 around the y-axis to all the points of $T_1(A)$ and $T_3(A)$ is the α_3-rotation of all the atoms of $T_2(A)$ around the z-axis.

2. For $j = 1, \ldots, na$, $T_{3+j}(A)$ is defined as the rotation of the points

   $$T_{3+j-1}(A_{iax(j)+2}), \ldots, T_{3+j-1}(A_{n_A})$$

   around the axis determined by $T_{3+j-1}(A_{iax(j)})$ and $T_{3+j-1}(A_{iax(j)+1})$.

3. $T(A) \equiv \{T(A_1), \ldots, T(A_{n_A})\}$ is the set of translated atoms $\{a + T_{3+n_{ax}}(A_1), \ldots, a + T_{3+n_{ax}}(A_{n_A})\}$.

A fixed orientation for rotations is preserved throughout all the calculations. Rotations are computed using the algorithm described in [31].

Clearly, $T(A)$ depends on $\alpha_i, i = 1, \ldots, (3 + n_{ax})$ and a_1, a_2, a_3. We will write $T = T(\alpha, a)$ when we want to emphasize this dependence.

Our goal is to find a movement that, in some sense, maximizes the similarity between $T(A)$ and B. We address this objective trying to match n_{guess} atoms of $T(A)$ ($n_{guess} \leq \min\{n_A, n_B\}$) with n_{guess} atoms of B. Typically, $n_{guess} \in [0.85 \times \min\{n_A, n_B\}, 0.95 \times \min\{n_A, n_B\}]$. The similarity function $f(\alpha, a)$ is computed by the following algorithm.

Algorithm FUN

Step 1. For all $i = 1, \ldots, n_A$, compute $T(A_i)$ and $pair(i) \in \{1, \ldots, n_B\}$ such that

$$\|T(A_i) - B_{pair(i)}\| = \min\{\|T(A_i) - B_j\|, j = 1, \ldots, n_B\}.$$

Step 2. Compute $I(T) \subset \{1, \ldots, n_A\}$, the set of indices that define the n_{guess} smaller distances $\|T(A_i) - B_{pair(i)}\|$. In other words, $\#I(T) = n_{guess}$ and for all $i \in I(T)$, $j \in \{1, \ldots, n_A\} - I(T)$, $\|T(A_i) - B_{pair(i)}\| \leq \|T(A_j) - B_{pair(j)}\|$.

Step 3. Compute

$$f(\alpha, a) = \sum_{i \in I(T)} \|T(A_i) - B_{pair(i)}\|^2. \tag{1}$$

The function f is nonsmooth and nonconvex. In spite of this, we will see that it is reliable to minimize it using, essentially, smooth optimization algorithms. If there exist n_{guess} atoms in A that fit exactly n_{guess} atoms of B then, a transformation $T(\alpha, a)$ exists such that $f(\alpha, a) = 0$. Since f is nonnegative this transformation will define a global minimizer of f. Reciprocally, if we find a global minimizer of f there are good chances of having found a good matching between A and B.

The following characterization of $f(\alpha, a)$ will be useful. We will prove that the merit function f computed by Algorithm **FUN** is the minimum of a set of smooth functions which, in turn, do not need to be computed at all.

Theorem 1. *Let I_A be the set of subsets of $\{1, \ldots, n_A\}$ with n_{guess} elements. Let J_B be the set of n_{guess}-uples of $\{1, \ldots, n_B\}$ Then,*

$$f(\alpha, a) = \min\left\{ \sum_{i=1}^{n_{guess}} \|T(A_{k_i}) - B_{\ell_j}\|^2 \mid \{k_1, \ldots, k_{n_{guess}}\} \in I_A, (\ell_1, \ldots, \ell_{n_{guess}}) \in J_B \right\}.$$

Proof. Let us emphasize that, when we write $\{k_1, \ldots, k_{n_{guess}}\} \in I_A$ we mean that $k_i \neq k_j$ if $i \neq j$. However, if $(\ell_1, \ldots, \ell_{n_{guess}}) \in J_B$ we may have $\ell_i = \ell_j$ for some $i \neq j$.

Let us write $I(T) = \{\bar{k}_1, \ldots, \bar{k}_{n_{guess}}\}$. Clearly, $(pair(\bar{k}_1), \ldots, pair(\bar{k}_{n_{guess}}))$ is an n_{guess}-uple whose elements are in $\{1, \ldots, n_B\}$. Therefore,

$$\min\left\{ \sum_{i=1}^{n_{guess}} \|T(A_{k_i}) - B_{\ell_j}\|^2 \mid \{k_1, \ldots, k_{n_{guess}}\} \in I_A, (\ell_1, \ldots, \ell_{n_{guess}}) \in J_B \right\}$$

$$\leq \sum_{i=1}^{n_{guess}} \|T(A_{\bar{k}_i}) - B_{pair(\bar{k}_i)}\|^2 = \sum_{i \in I(T)} \|T(A_i) - B_{pair(i)}\|^2 = f(\alpha, a).$$

Reciprocally, assume that $\{\bar{k}_1, \ldots, \bar{k}_{n_{guess}}\}$ and $(\bar{\ell}_1, \ldots, \bar{\ell}_{n_{guess}})$ are such that

$$\sum_{i=1}^{n_{guess}} \|T(A_{\bar{k}_i}) - B_{\bar{\ell}_i}\|^2 = \min \left\{ \sum_{i=1}^{n_{guess}} \|T(A_{k_i}) - B_{\ell_j}\|^2 \mid \{k_1, \ldots, k_{n_{guess}}\} \in I_A, (\ell_1, \ldots, \ell_{n_{guess}}) \in J_B \right\}. \tag{2}$$

By the definition of $pair(i)$, for all $i = 1, \ldots, n_{guess}$ we have:

$$\|T(A_{\bar{k}_i}) - B_{pair(\bar{k}_i)}\| \leq \|T(A_{\bar{k}_i}) - B_{\bar{\ell}_i}\|.$$

Therefore, by the definition of $I(T)$,

$$f(\alpha, a) = \sum_{i \in I(T)} \|T(A_i) - B_{pair(i)}\|^2 \leq \sum_{i=1}^{n_{guess}} \|T(A_{\bar{k}_i}) - B_{pair(\bar{k}_i)}\|^2 \leq \sum_{i=1}^{n_{guess}} \|T(A_{\bar{k}_i}) - B_{\bar{\ell}_i}\|^2.$$

By (2), this completes the proof. □

Let us call m the number of elements of the Cartesian product $I_A \times J_B$. Then, we can write $I_A \times J_B = \{\nu(1), \ldots, \nu(m)\}$. Suppose that $\nu(i) = (\{i_1, \ldots, i_{n_{guess}}\}, (j_1, \ldots, j_{n_{guess}}))$. Then, we define

$$f_i(\alpha, a) = \sum_{k=1}^{n_{guess}} \|T(A_{i_k}) - B_{j_k}\|^2. \tag{3}$$

By Theorem 1 we have that

$$f(\alpha, a) = \min\{f_i(\alpha, a), i = 1, \ldots, m\}. \tag{4}$$

Observe that the function $f(\alpha, a)$ is not differentiable although all the functions f_i have continuous partial derivatives for all α, a.

The characterization (4) allows one to minimize the nonsmooth function f using, essentially, smooth minimization algorithms. The idea is simple. Given the approximation (α^k, a^k) to the minimizer of f, one computes, using Algorithm **FUN**, an index i such that $f_i(\alpha^k, a^k) = f(\alpha^k, a^k)$. This index does not need to be unique, but this is irrelevant for our calculations. Since f_i is smooth, it is easy to compute a new approximation $(\alpha^k_{trial}, a^k_{trial})$ such that $f_i(\alpha^k_{trial}, a^k_{trial})$ is sufficiently smaller than $f_i(\alpha^k, a^k)$. But $f(\alpha^k_{trial}, a^k_{trial})$ is not greater than $f_i(x^k_{trial}, a^k_{trial})$, therefore $f(\alpha^k_{trial}, a^k_{trial})$ must be sufficiently smaller than $f(\alpha^k, a^k)$. This argument supports the generalization of any unconstrained method to the minimization of f. In the next section we will see that, due to the structure of the functions f_i, a generalization of the Gauss-Newton method is quite adequate for our problem.

3 Gauss-Newton-like Algorithm

For simplicity, let us write $x = (\alpha, a)$, $n = 6 + n_{ax}$. Then, $f_i : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable for all $x \in \mathbb{R}^n$, $i = 1, \ldots, m$. The first derivatives of f_i are hard to compute when $n_{ax} > 0$. Second derivatives also exist but their practical computation is prohibitive. The problem of minimizing f is an Order-Value Optimization problem in the sense of [1, 2, 3]. As in [2] we say that x^* is a critical point of the problem if there exists $i \in \{1, \ldots, m\}$ such that $f_i(x^*) = f(x^*)$ and $\nabla f_i(x^*) = 0$. According to [2], local minimizers of f must be critical points. The sum-of-squares structure (3) suggests the use of a variation of Gauss-Newton method for minimizing f. Observe that, by (3),

$$\nabla f_i(x) = 2J_i(x)^T F_i(x),$$

where $J_i(x)$ is a Jacobian $n_{guess} \times n$ matrix and $F_i(x) \in \mathbb{R}^{n_{guess}}$ is the corresponding residual vector. The Gauss-Newton direction $d_i(x)$ comes from solving:

$$G_i(x)d_i(x) = -J_i(x)^T F_i(x),$$

with $G(x) = J_i(x)^T J_i(x)$. If $J_i(x)$ is not full-rank or is severely ill-conditioned we replace $G_i(x)$ by $0.995G_i(x) + 0.005I$. If, after this change, $G_i(x)$ remains ill-conditioned we repeat this regularization. Eventually, $G_i(x)$ is suitable conditioned and $d_i(x)$ can be used as search direction.

Therefore, the direction $d_i(x)$ is given by

$$\Big[(1-t)J_i(x)^T J_i(x) + tI\Big]d_i(x) = -J_i(x)^T F_i(x),$$

for some $t \in [0, 1)$. Obviously, $d_i(x)$ is continuous as a function of t and tends to $-J_i(x)^T F_i(x)$ as t tends to 1. Assuming that $J_i(x)^T F_i(x) \neq 0$, it follows that:

$$\lim_{t \to 1} \frac{d_i(x)^T \nabla f_i(x)}{\|d_i(x)\| \|\nabla f_i(x)\|} = -1.$$

Therefore, given $c > 0$, after a finite number of replacements $G_i(x) \leftarrow 0.995G_i(x) + 0.005I$, one obtains that

$$d_i(x)^T \nabla f_i(x) \leq -c\|d_i(x)\| \|\nabla f_i(x)\|. \tag{5}$$

The formal description of the algorithm is the following.

Algorithm GN

Given $x^0 = (\alpha_1^0, \ldots, \alpha_{3+n_{ax}}^0, a_1^0, a_2^0, a_3^0) \in \mathbb{R}^n$, $\alpha_j^0 \in [0, 2\pi]$ for all $j = 1, \ldots, 3 + n_{ax}$, $\gamma \in (0, 1/2)$, $\theta \in (0, 1)$, $t_{inic} \geq 1$, $\varepsilon \geq 0$, perform the following steps:

Step 1. Initialize $k \leftarrow 0$.
Step 2. Compute $f(x^k)$. Let $i = i(x^k)$ be such that $f_i(x^k) = f(x^k)$.
Step 3. Compute $J_i(x^k)$ and $\nabla f_i(x^k)$. If $\|\nabla f_i(x^k)\|_\infty \leq \varepsilon$, terminate.

Step 4. Compute $d^k = d_i(x^k)$ such that

$$(d^k)^T \nabla f_i(x^k) \leq -\theta \|d^k\| \|\nabla f_i(x^k)\|. \tag{6}$$

using the Gauss-Newton procedure described above.
Step 5. Define t_k by one of the following rules:
Rule 5.a: The step t_k is the first element of the sequence $\{t_{inic} \times 2^{-k}, k = 0, 1, 2, \ldots\}$ such that

$$f(x^k + t_k d^k) \leq f(x^k) + \gamma t_k (d^k)^T \nabla f_i(x^k). \tag{7}$$

Rule 5.b: The step t_k is the first element of the sequence $\{t_{inic} \times 2^{-k}, k = 0, 1, 2, \ldots\}$ such that

$$f_i(x^k + t_k d^k) \leq f(x^k) + \gamma t_k (d^k)^T \nabla f_i(x^k). \tag{8}$$

Step 6. Compute $x^{k+1} = (\alpha_2^{k+1}, \ldots, \alpha_{3+n_{ax}}^{k+1}, a_1^{k+1}, a_2^{k+1}, a_3^{k+1})$ such that

$$\alpha_j^{k+1} \in [0, 2\pi),\ j = 1, \ldots, 3 + n_{ax} \tag{9}$$

and

$$f(x^{k+1}) \leq f_i(x^k + t_k d^k). \tag{10}$$

Set $k \leftarrow k + 1$ and go to Step 2.

Remark 1. There is a a subtle difference between the stepsize rules **5.a** and **5.b**. Since $f(x^k + t_k d^k) \leq f_i(x^k + t_k d^k)$ it is obvious that (8) implies (7). Therefore the step obtained by Rule **5.a** cannot be smaller than the one obtained by Rule **5.b**. For this reason, in our tests we use Rule **5.a** and $f(x^{k+1}) = f(x^k + t_k d^k)$. However, let us suppose that we are able to obtain w^k, the global minimizer of the function f_i. In this case, taking $x^{k+1} = w^k$, we have:

$$f(x^{k+1}) \leq f_i(x^{k+1}) = f_i(w^k) \leq f_i(x^k + t_k d^k).$$

This means that w^k may be taken as new iterate and that the computation of t_k may be avoided. In the case $n_{ax} = 0$, as we mentioned before, a global minimizer of f_i may be analytically obtained as the solution of the Procrustes problem therefore, in this case, the algorithm can be described as the iterated computation of $i(x^k)$ and Procrustes resolutions.

The computer work per iteration is dominated by $O(n_{guess} n^2)$. It can be reduced to $O(n_{guess} n)$ if one uses a quasi-Newton formula instead of Gauss-Newton. However, since $i(x^k)$ changes from one iteration to another, it is not clear how a quasi-Newton update should be implemented.

Theorem 2. *If Algorithm* **GN** *does not terminate at x^k, then x^{k+1} is well defined and $f(x^{k+1}) < f(x^k)$.*

Proof. Let $i = i(x^k)$. Since $\varepsilon \geq 0$, if the algorithm does not terminate at x^k, we have that $\nabla f_i(x^k) \neq 0$. By the reasoning that leads to (5), the direction d^k satisfies (6). Therefore, $(d^k)^T \nabla f_i(x^k) < 0$, so:

$$\lim_{t \to 0} \frac{f_i(x^k + t d^k) - f_i(x^k)}{t (d^k)^T \nabla f_i(x^k)} = 1.$$

Thus, for $t > 0$ small enough,

$$\frac{f_i(x^k + td^k) - f_i(x^k)}{t(d^k)^T \nabla f_i(x^k)} \geq \gamma$$

and, so,

$$f_i(x^k + td^k) \leq f_i(x^k) + \gamma t (d^k)^T \nabla f_i(x^k).$$

But $f(x^k) = f_i(x^k)$ and $f(x^k + td^k) \leq f_i(x^k + td^k)$. Therefore, for $t > 0$ small enough,

$$f(x^k + td^k) \leq f(x^k) + \gamma t (d^k)^T \nabla f_i(x^k).$$

Therefore, t_k is well defined at Step 5 of Algorithm **GN**.

Let us write

$$x^k + t_k d^k = (\alpha_1, \ldots, \alpha_{3+n_{ax}}, a_1, a_2, a_3).$$

Let $\beta_j \in [0, 2\pi)$, $j = 1, \ldots, 3+n_{ax}$ be such that $(\alpha_j - \beta_j)/(2\pi)$ is integer for all $j = 1, \ldots, 3+n_{ax}$. Clearly, $f(\beta_1, \ldots, \beta_{3+n_{ax}}, a_1, a_2, a_3) = f(\alpha_1, \ldots, \alpha_{3+n_{ax}}, a_1, a_2, a_3)$. Therefore, we may complete the iteration defining

$$x^{k+1} = (\beta_1, \ldots, \beta_{3+n_{ax}}, a_1, a_2, a_3).$$

□

Theorem 3. *The sequence generated by Algorithm* **GN** *lies in a compact set.*

Proof. By periodicity, we may think that $T_{3+n_{ax}}$ is generated by rotations with angles $\alpha_j \in [0, 2\pi)$. By the continuity of $T_{3+n_{ax}}$ and the compacity of $[0, 2\pi]$, there exists $c_1 > 0$ such that $\|T_{3+n_{ax}}(A_i)\| \leq c_1$ for all $i = 1, \ldots, n_A$, and all possible choices of $\alpha_1, \ldots, \alpha_{3+n_A}$.

Clearly, there exists $c_2 > 0$ such that $\|B_\ell\| \leq c_2$ for all $\ell = 1, \ldots, n_B$.

Let $M > \sqrt{f(x^0)}$. Let $\|a\| > M + c_1 + c_2$, $\alpha_j \in [0, 2\pi)$ for all $j = 1, \ldots, 3 + n_A$, $T = T(\alpha, a)$, $i \in \{1, \ldots, n_A\}$, $\ell \in \{1, \ldots, n_B\}$. Then,

$$\|T(A_i) - B_\ell\| \geq \|T(A_i)\| - \|B_\ell\| \geq \|T(A_i)\| - c_2.$$

But

$$\|T(A_i)\| \geq \|a\| - \|T(A_i) - a\| = \|a\| - \|T_{3+n_{ax}}(A_i)\| \geq \|a\| - c_1,$$

so,

$$\|T(A_i) - B_\ell\| \geq \|a\| - c_1 - c_2 > M > \sqrt{f(x^0)}.$$

Therefore, $\|T(A_i) - B_\ell\|^2 > f(x^0)$ for all $i = 1, \ldots, n_A$, $\ell = 1, \ldots, n_B$. This implies that $f(\alpha, a) > f(x^0)$. So,

$$f(x) \leq f(x^0) \Rightarrow \|a\| \leq M + c_1 + c_2.$$

Since $f(x^k) \leq f(x^0)$ for all k, we have that

$$\|a^k\| \leq M + c_1 + c_2$$

for all k. Since $\alpha_j^k \in [0, 2\pi)$ for all $j = 1, \ldots, 3 + n_{ax}$, the thesis is proved. □

Theorem 4. *Let $\{x^k\}$ be an infinite sequence generated by Algorithm* **GN**. *Then, there exists $c > 0$ such that*

$$t_k \geq \min\left\{\frac{1}{\|d^k\|}, \frac{(1-\gamma)\theta\|\nabla f_{i(x^k)}(x^k)\|}{2c\|d^k\|}, t_{inic}\right\}$$

for all $k \in I\!N$.

Proof. The functions f_i have continuous second derivatives for all $x \in I\!R^n$. By Taylor's formula, since $\{x^k\}$ is bounded, there exists $c > 0$ such that, whenever $\|td^k\| \leq 1$,

$$f_i(x^k + td^k) \leq f_i(x^k) + t(d^k)^T\nabla f_i(x^k) + c\|td^k\|^2 \tag{11}$$

for all $i = 1, \ldots, m$.

Since $f(x^k + td^k) \leq f_i(x^k)$ for all $i = 1, \ldots, m$ and $f_{i(x^k)}(x^k) = f(x^k)$, we deduce that, if $\|td^k\| \leq 1$,

$$f(x^k + td^k) \leq f(x^k) + t(d^k)^T\nabla f_{i(x^k)}(x^k) + c\|td^k\|^2$$

for all $k \in I\!N$. By (6), $(d^k)^T\nabla f_{i(x^k)}(x^k) < 0$ and, by the definition of the algorithm, since $\nabla f_{i(x^k)}(x^k) \neq 0$, we also have that $d^k \neq 0$. By (11) and an elementary calculation shows that, if

$$t \in \left(0, \frac{(\gamma-1)(d^k)^T\nabla f_{i(x^k)}(x^k)}{c\|d^k\|^2}\right)$$

and $\|td^k\| \leq 1$, then

$$f_i(x^k + td^k) \leq f(x^k) + \gamma t(d^k)^T\nabla f_{i(x^k)}(x^k).$$

and, so,

$$f(x^k + td^k) \leq f(x^k) + \gamma t(d^k)^T\nabla f_{i(x^k)}(x^k).$$

Therefore,

$$t_k \geq \min\left\{\frac{1}{\|d^k\|}, \frac{(\gamma-1)(d^k)^T\nabla f_{i(x^k)}(x^k)}{2c\|d^k\|^2}, t_{inic}\right\}.$$

Thus, by (6),

$$t_k \geq \min\left\{\frac{1}{\|d^k\|}, \frac{(1-\gamma)\theta\|\nabla f_{i(x^k)}(x^k)\|}{2c\|d^k\|}, t_{inic}\right\},$$

as we wanted to prove. □

Theorem 5. *Let $\{x^k\}$ be an infinite sequence generated by Algorithm* **GN**. *Then,*

$$\lim_{k\to\infty} \|\nabla f_{i(x^k)}(x^k)\| = 0.$$

Proof. By the definition of Algorithm **GN** and (6) we have that

$$f(x^{k+1}) \leq f(x^k) - \gamma\theta t_k\|d^k\|\|\nabla f_{i(x^k)}(x^k)\|$$

for all $k \in I\!N$.

Therefore, by Theorem 4,

$$f(x^{k+1}) \leq f(x^k) - \min\left\{\gamma\theta\|\nabla f_{i(x^k)}(x^k)\|, \frac{\gamma(1-\gamma)\theta^2\|\nabla f_{i(x^k)}(x^k)\|^2}{2c}, \gamma\theta t_{inic}\|d^k\|\|\nabla f_{i(x^k)}(x^k)\|\right\}. \tag{12}$$

But $G(x^k)d^k = -\nabla f_{i(x^k)}(x^k)$, so $\|\nabla f_{i(x^k)}(x^k)\| \leq \|G(x^k)\|\|d^k\|$. Since $\{x^k\}$ is bounded, $\|G(x^k)\|$ is bounded as well. Therefore, there exists $c_1 > 0$ such that $\|\nabla f_{i(x^k)}(x^k)\| \leq c_1\|d^k\|$ for all $k \in I\!N$. Therefore, by (12),

$$f(x^{k+1}) \leq f(x^k) - \min\left\{\gamma\theta\|\nabla f_{i(x^k)}(x^k)\|, \frac{\gamma(1-\gamma)\theta^2\|\nabla f_{i(x^k)}(x^k)\|^2}{2c}, \frac{\gamma\theta t_{inic}\|\nabla f_{i(x^k)}(x^k)\|^2}{c_1}\right\}$$

for all $k \in I\!N$. Since $f(x^k) \geq 0$ for all $k \in I\!N$, this inequality implies that $\|\nabla f_{i(x^k)}(x^k)\| \to 0$, as we wanted to prove. □

Theorem 4 shows that $\nabla f_{i(x^k)}(x^k)$ plays the same role as $\nabla f(x^k)$ does in smooth unconstrained optimization. As a consequence, the algorithm always terminates in a finite number of iterations if $\varepsilon > 0$. In **Remark 1** we observed that the choice $x^{k+1} = \arg\min f_{i(x^k)}(x)$ is a particular case of the iteration of Algorithm **GN**. When this choice is adopted, the algorithm terminates in a finite number k_{tot} of iterations, even when $\varepsilon = 0$, at a global minimizer of $f_{i(x^{k_{tot}})}$. This follows from the fact that, in this case, $k \neq \ell$ implies that $i(x^k) \neq i(x^\ell)$.

4 Implementation Features

4.1 Initial Approximations for Sequential Alignments

In practical terms, Theorem 5 says that Algorithm **GN** tends to find local minimizers of the function given by (1). In order to increase the chance of finding *global* minimizers we need to use suitable initial angles and displacements.

Many times (but not always!) there exists a natural correspondence between chains of consecutive atoms of A and B. In order to exploit this fact, we define $nsmall = \lfloor 0.45 \times n_{guess} \rfloor$ (with $n_{guess} \approx \lfloor 0.90 \times \min\{n_A, n_B\} \rfloor$) and we seek a chain of $nsmall$ atoms of A that is *analogous* to a chain of $nsmall$ atoms of B, without internal rotations. For each atom A_i $(i = 4, \ldots, n_A)$, we define

$$\Delta(A_i) = (\|A_i - A_{i-1}\|^2, \|A_i - A_{i-2}\|^2, \|A_i - A_{i-3}\|^2).$$

Analogously, for all $i = 4, \ldots, n_B$,

$$\Delta(B_i) = (\|B_i - B_{i-1}\|^2, \|B_i - B_{i-2}\|^2, \|B_i - B_{i-3}\|^2).$$

If T is a movement *without internal rotations* we have that

$$\Delta(T(A_i)) = \Delta(A_i) \text{ for all } i = 4, \ldots, n_A.$$

Therefore, if the chain $(T(A_i), T(A_{i+1}), \ldots, T(A_{i+nsmall-1}))$ fits the chain $(B_j, B_{j+1}, \ldots, B_{j+nsmall-1})$ $(4 \leq i \leq n_A - nsmall + 1, 4 \leq j \leq n_B - nsmall + 1)$ we have that

$$\Delta(A_i) \approx \Delta(B_j), \ldots, \Delta(A_{i+nsmall-1}) \approx \Delta(B_{j+nsmall-1}).$$

The observations above led us to define the following algorithm for finding small paired chains.

Algorithm CHAIN
Step 1. Set $big \leftarrow \infty$.
Step 2. For $i = 4, \ldots, n_A - nsmall + 1, j = 4, \ldots, n_B - nsmall + 1$, execute Step 3.
Step 3. If

$$\sum_{k=1}^{nsmall} \|\Delta(A_{i+k-1}) - \Delta(B_{j+k-1})\|_1 \leq big$$

then, set

$$big \leftarrow \sum_{k=1}^{nsmall} \|\Delta(A_{i+k-1}) - \Delta(B_{j+k-1})\|_1$$

and save (i, j).

Each saved small pair of chains is used to find initial approximations for angles and displacements. If the pair (i, j) was saved by Algorithm **CHAIN**, we compute the movement associated to (i, j) solving

$$\text{Minimize} \sum_{k=1}^{nsmall} \|T(A_{i+k-1}) - B_{j+k-1}\|^2. \tag{13}$$

Problem (13) is a smooth nonlinear least-squares problems whose variables are $\alpha_1, \ldots, \alpha_{3+n_{ax}}$, a_1, a_2, a_3. Usually, it is quite easy to solve (13) by means of a simplification of Algorithm **GN**. Moreover, when there are no internal axes, a global minimizer is available as a solution of the Procrustes problem.

This procedure is used to find initial points for minimizing (1), as described in the following algorithm.

Algorithm INITIAL
Step 1. Set $big \leftarrow \infty$.
Step 2. For each (i, j) saved by Algorithm **CHAIN**, execute Steps 3 and 4.
Step 3. Solve the easy auxiliary problem (13). Let f_{easy} be the objective function value at the solution of this problem.
Step 4. If $f_{easy} \leq big$, set $big \leftarrow f_{easy}$ and save the solution $(\alpha_1, \ldots, \alpha_{3+n_{ax}}, a_1, a_2, a_3)$ of (13).

As a result of the application of Algorithm **INITIAL**, we have a few initial points (angles and displacements) that may be used by **GN** for minimizing (1). Algorithm **GN** uses these initial points for finding minimizers of (1) in the cases in which the Order-Preserving Property must be fulfilled.

4.2 Computing the Function *pair*

At Step 1 of Algorithm **FUN** we need to compute, for all $i = 1, \ldots, n_A$, the atom of B which is closest to $T(A_i)$. This computation can be costly, if n_B is large. For reducing the cost as much as possible, we store, for each atom B_i, the distances to the atoms B_j in increasing order. The algorithm used for this purpose is FLASHSORT [20]. This sorting process is rather expensive, even using this fast subroutine, but it is performed only once, as a preprocessing phase of the alignment. The availability of the distances to each B_i in increasing order is essential for abbreviating the computation of $pair(i)$ at each call of FUN. We proceed as follows: Assume that $i \in \{1, \ldots, n_B\}$. Let $B_{j_1}, B_{j_2}, B_{j_3}, \ldots$ be the atoms of B with $\|B_{j_{k+1}} - B_{j_1}\| \geq \|B_{j_k} - B_{j_1}\|$ for all k. B_{j_1} may be chosen arbitrarily although it is desirable that j_1 should be a good guess for $pair(i)$. In our implementation we choose $j_1 = 1$ if $i = 1$ and $j_1 = pair(i-1)$ if $i > 1$. Then, $pair(i)$ is computed by the following algorithm:

Algorithm PAIR
Step 1. Let $kmax$ be the largest k such that

$$\|B_{j_k} - B_{j_1}\| \leq 2\|B_{j_1} - T(A_i)\|. \tag{14}$$

Step 2. Compute $kmin \in \{1, \ldots, kmax\}$ such that

$$\|B_{j_{kmin}} - T(A_i)\| = \min\{\|B_{j_1} - T(A_i)\|, \ldots, \|B_{j_{kmax}} - T(A_i)\|\}.$$

Define $pair(i) = j_{kmin}$.

Observe that, by (14), if $k > kmax$,

$$\|B_{j_k} - T(A_i)\| \geq \|B_{j_k} - B_{j_1}\| - \|B_{j_1} - T(A_i)\| > \|B_{j_1} - T(A_i)\|.$$

Therefore, the distances $\|B_{j_k} - T(A_i)\|$ do not need to be computed for $k > kmax$.

4.3 Algorithmic Parameters

4.3.1 Angle and Sufficient Descent

The parameter θ is a lower bound for the cosine of the angle between the (negative) gradient and the search direction. Since $G_i(x)$ is positive semidefinite this cosine is always nonnegative but it may be close to zero if $G_i(x)$ is severely ill-conditioned. We use $\theta = 10^{-8}$.

In smooth unconstrained optimization, the sufficient decrease (or Armijo) parameter is generally equal to 10^{-4}. We follow the same tradition here.

4.3.2 Stopping Criteria

We use $\varepsilon = 10^{-4}$. Sometimes, the algorithm uses a rather large number of unfruitful iterations to arrive to $\|\nabla f_i(x^k)\| \leq \varepsilon$, so we use a second stopping criterion:

$$f(x^{k-1}) - f(x^k) \leq \varepsilon_{rel} f(x^{k-1})$$

with $\varepsilon_{rel} = 10^{-8}$.

Usually, **GN** terminates in less than 45 iterations. However, we also are prepared to stop the process if the number of iterations is greater than 100.

4.3.3 Initial Step

The initial step t_{inic} is a sensitive parameter for this class of problems. In the classical Gauss-Newton method for ordinary unconstrained smooth nonlinear least-squares problems, one uses $t_{inic} = 1$, with the expectancy of obtaining quadratic convergence, if the residual at the solution is close to zero [21] (Chapter 10). In our problem, one of whose characteristics is the presence of many local minimizers, it is better to use $t_{inic} > 1$. Many times, this choice allows the algorithm to jump over local-nonglobal minimizers and to reach lower functional values. In our experiments we used $t_{inic} = 1.75$. A remarkable experimental fact is that, in the vast majority of iterations, only one function evaluation is needed (sufficient descent was obtained with $t_k = t_{inic}$).

4.4 Greedy Bijection

The objective of alignment is to fit the largest number of atoms of A to the corresponding atoms of B with the smallest possible average sum of squared distances. So, one has two goals: maximize the number of matched pairs and minimize the distances. Obviously, these two objectives are conflictant. When the number of matched atoms increases, the average sum of squared distances tends to increase and vice-versa. Our approach to the problem uses a tentative number of matches n_{guess} and finds the movement that minimizes the sum of squared distances of n_{guess} atoms of Structure A to n_{guess} atoms of Structure B, allowing repetitions in B. We sacrifice bijection in order to produce a suitable optimization problem. It seems plausible that the optimal movement so far obtained will also produce good fittings when we force a one-to-one correspondence between points of $T(A)$ and atoms of B, but this hypothesis needs to be tested in practice.

Assume that our optimal movement is represented by the transformation T, so that the rotated and displaced Structure A is $T(A) = \{T(A_1), \ldots, T(A_{n_A})\}$. Without loss of generality, assume $n_A \leq n_B$. For each $n_{bij} \in \{1, \ldots, n_A\}$ we wish to consider all the bijections between n_{bij} points of $T(A)$ and n_{bij} points of B, selecting the one for which the sum of squared distances is the smallest. This exhaustive procedure would be extremely expensive, therefore we use cheap *greedy* approximation. The algorithm that computes the greedy one-to-one correspondences is given below.

Algorithm GREEDY

Define

$$d_{i,j} = \|T(A_i) - B_j\|^2, i = 1, \ldots, n_A,\ j = 1, \ldots, n_B.$$

For $n_{bij} = 1, \ldots, n_A$, execute Steps 1–2.

Step 1. Compute $i_{n_{bij}}, j_{n_{bij}}$ such that

$$d_{i_{n_{bij}}, j_{n_{bij}}} = \min\{d_{i,j}, i = 1, \ldots, n_A, j = 1, \ldots, n_B\}.$$

Step 2. For all $i = 1, \ldots, n_A, j = 1, \ldots, n_B$, replace

$$d_{i_{n_{bij}},j} \leftarrow \infty,\ d_{i,j_{n_{bij}}} \leftarrow \infty.$$

Algorithm **GREEDY** computes a bijection between the points $\{T(A_1), \ldots, T(A_{n_A})\}$ and a subset of n_A points of $\{B_1, \ldots, B_{n_B}\}$. This bijection exhibits the increasing order property:

$$\|T(A_{i_1}) - B_{j_1}\| \leq \|T(A_{i_2}) - B_{j_2}\| \leq \ldots \|T(A_{i_{n_A}}) - B_{j_{n_A}}\|.$$

The first n_{bij} pairs of this bijection define the optimal greedy matching between $T(A)$ and B with n_{bij} elements, under the movement computed by the Gauss-Newton approach. Associated to each $n_{bij} \in \{1, \ldots, n_A\}$, we have a root mean-squared deviation $rmsd(n_{bij})$, defined by

$$rmsd(n_{bij}) = \sqrt{\frac{1}{n_{bij}} \sum_{k=1}^{n_{bij}} \|T(A_{i_k}) - B_{j_k}\|^2}.$$

These quantities provide means for comparison of this approach with methods for Protein Alignment.

Several warnings regarding the greedy algorithm are necessary:

1. Algorithm **GREEDY** is not optimal at all. Given n_{bij}, better one-to-one correspondences associated to the superposition computed by **GN** probably exist.

2. Given n_{bij} and the bijection Φ computed by **GREEDY**, a smaller value for the root mean squared deviation *rmsd* may be obtained matching the domain with the range of Φ (using Procrustes if $n_{ax} = 0$). The reported $rmsd(n_{bij})$ corresponds to the movement computed by **GN** and no additional effort is made trying to reduce it.

3. The bijection computed by **GREEDY** does not necessarily fulfill the Order-Preserving Property and there is no limitation in the number of gaps it produces.

5 Numerical Experiments

5.1 An Illustrative Example

In order to visualize the concepts displayed above, let us consider the comparison of two protein-like structures with 115 and 123 atoms each. We applied Algorithm **GN** with the initial points provided by Algorithm **INITIAL**. We used $n_{guess} = \lfloor 0.85 \times n_A \rfloor = 97$ and $nsmall = \lfloor 0.45 \times n_{guess} \rfloor = 43$.

5.1.1 Without Internal Rotations

Let us consider first the case in which $n_{ax} = 0$. In this case, Algorithm **INITIAL** saved only two points, therefore Algorithm **GN** needed to be executed twice. The best of these two runs gave $f(x^*) = 428.66$. The best translation obtained was

$$(a_1^*, a_2^*, a_3^*) = (0.284, -3.25, 1.11)$$

n_{bij}	*rmsd* (rigid)	MD (rigid)	*rmsd* (flexible)	MD (flexible)
10	0.727	0.880	0.794	1.019
20	0.880	1.118	0.947	1.132
30	1.029	1.368	1.061	1.439
40	1.177	1.690	1.211	1.674
50	1.369	2.107	1.349	1.923
60	1.539	2.337	1.490	2.140
70	1.720	2.721	1.636	2.509
78	1.900	3.323	1.793	3.090
80	1.960	3.632	1.841	3.223
90	2.430	5.402	2.191	4.711
100	3.143	8.832	2.894	7.428
110	5.514	21.987	4.584	17.966
115	8.029	35.113	6.687	25.693

Table 1: Results for the illustrative example. MD is the Maximal Distance obtained for rigid and flexible (with one internal rotation) alignments. The internal rotation reduces both *rmsd* and MD for significant (> 50) values of n_{bij}.

and the best rotation angles were:

$$(\alpha_1^*, \alpha_2^*, \alpha_3^*) = (3.881, 6.055, 4.586).$$

5.1.2 With One Internal Rotation

In this example we consider $n_{ax} = 1$. The internal rotation is around the axis determined by atoms 50 and 51 of Protein A. Again, Algorithm **GN** needed to be executed only twice (two points saved by **INITIAL**). The best objective function was $f(x^*) = 422.20$. The best translation obtained was

$$(a_1^*, a_2^*, a_3^*) = (0.0936, -2.84, 0.575).$$

The best rotation angles were:

$$(\alpha_1^*, \alpha_2^*, \alpha_3^*) = (3.857, 6.042, 4.640).$$

The best internal rotation angle was:

$$\alpha_4^* = 0.109.$$

Using the Greedy Alignment (Algorithm **GREEDY**) we obtained the results given in Table 1.

5.2 A Comparison against DALI

In order to test the reliability of the **GN** approach we selected, at random, 5 proteins of the Protein Data Bank. We obtained from the DALI webpage the 20 proteins of the PDB that are more similar to the selected proteins, according to [10] (see Appendix). The Protein Data Bank identifier (PDBId) and names of the five selected proteins are:

1. 1LBD: *Retinoid-X-Receptor* (238 Cα atoms)
2. 1NPD: *Hypothetical Shikimate 5-Dehydrogenase* (277 Cα atoms)
3. 1KTB: *Alpha-N-Acetylgalactosaminidase* (388 Cα atoms)
4. 1UKX: *GCN2 EIF2α kinase* (2740 Cα atoms)
5. 1M4U: *Bone Morphogenic Protein-7* (112 Cα atoms)

As usually in Protein Alignment, the comparison involves the structure of Cα atoms of the different proteins.

For each pair of proteins under consideration, we consider the values "lali" and "rmsd" reported by DALI's tables. These values refer to the best alignment computed by DALI. "lali" is the number of aligned pairs and "rmsd" is the root mean-squared deviation related to the alignment. Recall that the optimal alignment computed by DALI is obtained from the analysis of the internal distance matrices. Once the bijection Φ is determined, range and domain of Φ are superimposed using Procrustes and "rmsd" is computed for this superposition. We applied our method to the same comparison, including the **GREEDY** algorithm, and we selected n_{bij} ="lali". We report $rmsd(n_{bij})$, as defined in Section 5. We *do not* intend to superimpose the bijection that corresponds to n_{bij}. Instead, $rmsd(n_{bij})$ is completely determined by Algorithm **GN**.

In the comparison between our algorithm and DALI, we consider that **GN** "wins" if $rmsd(n_{bij}) <$ "rmsd" (given by DALI) and vice-versa. This analysis provides a rough basis of comparison. It must be observed that neither DALI nor **GN** are designed with the purpose of minimizing the root mean-squared deviation for "lali" atoms. DALI maximizes a score function, on the basis of which the value of "lali" is decided, and our algorithm minimizes the similarity function f, for $n_{guess} = 0.9n_A$. However, the root mean-squared deviation related to a "lali"-bijection is the only parameter that may be used to assess the reliability of our approach.

With these warnings in mind, we report the results in Table 2. In this table, the first 20 best alignments obtained by DALI [10] are reported together with the $rmsd$ of **GN** (between parentheses) for n_{bij} ="lali". The names of the proteins corresponding to these alignments are given in the Appendix.

As can be observed in Table 2, the **GN** algorithm obtained smaller rmsd's for 88 of the 100 alignment tests. Five alignments were ties, since the rmsd's obtained by both methods were zero when comparing the proteins to themselves (the first alignment of each set). Finally, the rmsd obtained by DALI was smaller than the one obtained by **GN** in 7 alignments. Unfortunately, the comparison of alignment algorithms is a very difficult task, because each algorithm is based on a different score [15]. The purpose of this comparison is not to state that the quality of the alignments obtained by **GN** is better than the ones given by DALI, but to provide convincing evidence that the **GN** approach is reliable, particularly if the natural rmsd score is used.

In general, there is a clear correlation between the rmsd obtained by DALI and the rmsd obtained by the **GN** algorithm. An exception is the 1UKX example, in which the rmsd's obtained by **GN** were much smaller than the rmsd's obtained by DALI. The proteins being compared in this example are relatively large (2740 Cα atoms) and, as can be observed, the

	1LBD		1NPD		1KTB		1UKX		1M4U	
Protein	lali	rmsd	lali	rmsd	lali	rmsd	lali	rmsd	lali	rmsd
1	238	0.0(0.00)	288	0.0(0.00)	388	0.0(0.00)	137	0.0 (0.00)	112	0.0(0.00)
2	182	1.1(0.78)	276	**1.9(2.12)**	383	1.1(1.07)	65	3.2 (1.68)	106	0.9(0.81)
3	193	2.4(1.21)	253	**2.6(2.92)**	361	**1.6(1.70)**	42	2.5 (1.45)	103	3.2(2.77)
4	191	2.5(1.90)	263	2.1(2.24)	357	1.8(1.48)	59	3.0 (1.42)	87	2.5(1.50)
5	196	2.5(1.90)	252	2.7(2.28)	317	3.6(2.41)	43	4.7 (1.40)	103	4.0(3.34)
6	189	2.2(1.86)	267	**3.5(4.72)**	301	3.7(2.55)	59	4.1 (1.81)	59	1.5(1.26)
7	190	2.2(1.86)	229	3.7(2.82)	316	3.7(2.72)	55	4.2 (0.43)	81	6.2(4.17)
8	202	3.1(2.42)	220	3.1(2.23)	309	**3.1(3.15)**	46	3.5 (1.57)	77	3.4(2.61)
9	191	2.5(2.00)	221	3.6(2.45)	315	**3.3(3.52)**	47	5.8 (1.83)	72	3.9(3.30)
10	189	2.4(2.03)	171	3.7(2.61)	318	3.6(2.61)	60	12.1(0.62)	68	4.9(3.08)
11	197	2.6(2.15)	172	4.1(2.34)	320	3.8(3.03)	55	4.5 (1.50)	74	6.2(2.45)
12	194	2.9(1.92)	223	3.4(2.51)	319	3.8(2.52)	62	8.3 (0.24)	65	3.8(2.37)
13	192	2.6(1.89)	168	4.2(2.83)	276	3.6(3.13)	63	5.3 (1.51)	54	3.4(0.85)
14	188	2.5(2.05)	208	3.6(2.33)	292	3.8(3.16)	50	3.3 (0.64)	35	2.8(1.75)
15	109	5.9(2.71)	183	4.9(2.70)	320	3.9(2.52)	52	4.7 (1.72)	45	3.9(2.00)
16	92	3.6(2.26)	219	3.3(2.39)	318	3.9(2.54)	56	8.4 (1.46)	57	6.2(2.25)
17	93	5.2(1.99)	224	**3.2(3.86)**	315	3.7(2.69)	51	5.1 (0.79)	46	4.6(1.75)
18	84	4.3(1.99)	217	3.4(2.49)	292	3.8(3.24)	54	7.5 (0.26)	48	5.6(2.12)
19	94	4.3(3.04)	205	3.3(2.30)	276	3.6(3.02)	46	6.3 (1.34)	40	2.9(1.78)
20	83	5.3(1.94)	220	3.5(2.57)	297	3.7(3.58)	58	5.3 (1.31)	52	6.3(2.05)

Table 2: A comparison of Gauss-Newton with DALI alignments. The lali value is the one reported by DALI. The rmsd values reported are the ones available for DALI alignments and, between parentheses, the value *rmsd(lali)* provided by the Gauss-Newton method. Cases were the rmsd obtained by Gauss-Newton is larger than the rmsd obtained by DALI are highlighted in bold-face fonts.

Method	Average time / s
SSAP	3.79
STRUCTAL	0.80
DALI	1.89
LSQMAN	0.75
CE	1.11
SSM	0.26
GAUSS-NEWTON	1.39

Table 3: Average time required to perform an alignment

values of lali reported by DALI are small (up to 65). Therefore, in this case, the DALI method was unable to find good alignments and, for the same small values of n_{bij} =lali, the **GN** algorithm finds much better rmsd's. This occurs independently of the large variations of n_{guess} (from 43 to 2466, depending on the size of the second protein). For example, we observed that in the comparison of 1UKX with 1OV2 (3198 Cα atoms), the **GN** solution induces a bijective alignment of 1471 atoms with the same rmsd reported by DALI (3.3 Å) for lali=50.

5.2.1 Timing Analysis

Protein Alignment methods are frequently used to compare one or more proteins with all the structures of a large database. This requires, not only quality of the solutions found, but also efficiency in terms of computer time.

The comparison of times required for the alignment of proteins in a large database was reported in [15]. The results presented were obtained on a Intel Xeon 2.8 GHz. In order to roughly estimate the speed of the **GN** approach, we performed a comparison of the Retinoic Acid Receptor (pdb id. 1LBD, 238 atoms) to all the structures in Protein Data Bank (31948 structures). Although the number of comparisons of our test is smaller, it provides a tentative measure of the average time required for the alignment of two structures in the PDB database. Our results were obtained in an Opteron 242 with 1.2 GHz, which is a CPU similar in speed as the one used in [15]. The code was written in Fortran77 (with double precision) and was compiled with the GNU Fortran compiler version 3.4.4 with the options -O3 -ffast-math. The results are presented in Table 3.

This comparison seems to show that the Gauss-Newton method is competitive with current state-of-the-art algorithms for protein alignment. The most popular algorithm continues to be DALI and the time required by each alignment of the Gauss-Newton method is similar to the DALI average time.

The distribution of the time required by each task in the alignment performed by the Gauss-Newton algorithm is given in Table 4. We observe that distance matrix computations (calculation and ordering) takes about 20% of the total time of the alignment. This is step is not required in a large database comparison, since these matrices can be computed and stored previously. In our run, the matrices for both proteins were computed every time a new alignment was performed.

The most costly task was the **INITIAL** heuristic. No algorithm that ignores the fact that

Task	Average time required / s
Reading files	0.015
INITIAL Heuristic	0.600
Distance matrix computations	0.312
Gauss-Newton	0.460

Table 4: Time required by each task of the Gauss-Newton program.

most protein alignments are indeed sequential can compete both in quality and speed with an algorithm that takes advantage of this property. The **INITIAL** heuristic *is not* a fundamental part of our method and, in principle, may be replaced by different initial-point techniques (as multistart or Monte Carlo).

Fast algorithms using the secondary structure (as SSM does) could be used to provide initial approximations for Gauss-Newton alignments. In this case, the time increase required by the use **GN** does not appear to be critical. The fact that the **GN** trials take 0.460 s per alignment is encouraging. We observed that each trial uses an average of 42.9 GN iterations and decreases the function value obtained by the INITIAL procedure by an average of 65%. Therefore, the **GN** algorithm significantly improves the quality of the alignments while being relatively fast.

5.3 General alignments

5.3.1 Binding site identification

In this section we illustrate the application of Algorithm **GN** to two cases in which binding sites of proteins need to be identified.

The first case involves the identification of the binding site of DNA in Human Topoisomerase. Two structures of the Human Topoisomerase bound to DNA strands [24, 6] were taken from the Protein Data Bank. The structures are similar, but not identical. The DNA binding site in the first structure was defined as the set of Cα atoms whose distance to the DNA strand is less than 6Å. The full structure and the selected Cα atoms (black balls) are shown in Figure 1(a). The second Topoisomerase is represented in Figure 1(b). The goal was to identify the binding site in this second structure. The complete binding site contains 238 Cα atoms and preserves some sequentiality, but it exhibits 28 gaps. The complete protein structures contain 471 Cα atoms.

We performed alignments both *with* and *without* the **INITIAL** heuristic, using $n_{guess} = n_A$. In both cases the binding site in the second structure was correctly identified. The heuristic, however, achieved a best function value of 18340.49, while, at the solution, the function value $f(x^*)$ was 80.99. Therefore, the **GN** execution was crucial for obtaining the global solution, and decreased the merit function value by 96.6%. The heuristic is not clearly successful in this case due to the presence of many gaps in the solution.

We also performed 100 **GN** trials starting from random initial points. The global solution was found in 75 of these 100 trials. Therefore, in this case, the **GN** algorithm is successful even without using sequential information. The best alignment can be seen in Figure 1(c).

The second case involves the identification of the binding site of the Triac ligand in the isoform β of the Thyroid Hormone Receptor [22]. This binding site contains all the atoms

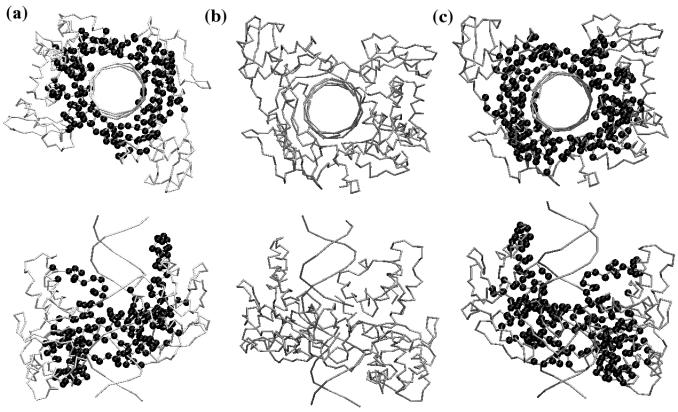

Figure 1: DNA binding site identification in Topoisomerases. (a) Structure of the Human Topoisomerase bound to DNA with the Cα atoms of residues close to DNA drawn as black balls. (b) A different structure of the Human Topoisomerase in which the binding site is to be recognized. (c) The alignment between the binding site Cα atoms of the first structure (black) to the second structure.

(not only Cα) corresponding to the 16 amino acids of the binding site of the Thyroid Hormone Receptor α, bound to the same ligand. The residual indices of the binding site in the Thyroid Hormone Receptor α are 215, 218, 219, 221, 222, 225, 256, 260, 262, 263, 276, 277, 290, 291, 292, and 381. Therefore, few sequential information seems to be useful. The binding site contains 244 atoms and the β isoform contains 4151 atoms.

The similarity between the isoform α and the isoform β allows one to obtain a solution by isoform superposition with rmsd = 0.52 Å. For obtaining this solution we simply align the whole isoform α with the isoform β and we take the subset of atoms corresponding to the binding site. Therefore, for a successful alignment of binding-site versus Isoform β one expects to obtain rmsd$\approx$ 0.52.

However, the alignment of the binding site versus Isoform β using the **INITIAL** heuristic was completely unsuccessful, probably due to the large number of gaps in the solution. Moreover, even using 10000 initial random approximations for **GN** the best rmsd obtained was 1.12Å. (We used $n_{guess} = 0.9\lfloor n_A \rfloor$.) In spite of this lack of success, a remarkable fact was that, for most alignments obtained with 10000 trials, we got rmsd$\leq$1.5Å. Each trial employed an average of 39 **GN** iterations. It is somewhat surprising that many local minimizers can be identified in this problem with very small values of rmsd.

In Figure 2(a) we show the structure of the isoform α bound to Triac. The molecule inside (balls) is the Triac ligand. The binding site is the group of amino acids that interact with the ligand directly, drawn as black sticks. In Figure 2(b) we show the isoform β with the optimal binding site determined by its alignment with the isoform α. In Figure 2(c) we show the isoform β with the best binding site (rmsd=1.12 Å) obtained by **GN**. Observe that this "local" binding site is not close to the optimal one. This example illustrates the fact that the number of local minimizers increases as n_A/n_B decreases. The quotient n_A/n_B was approximately 0.51 in the

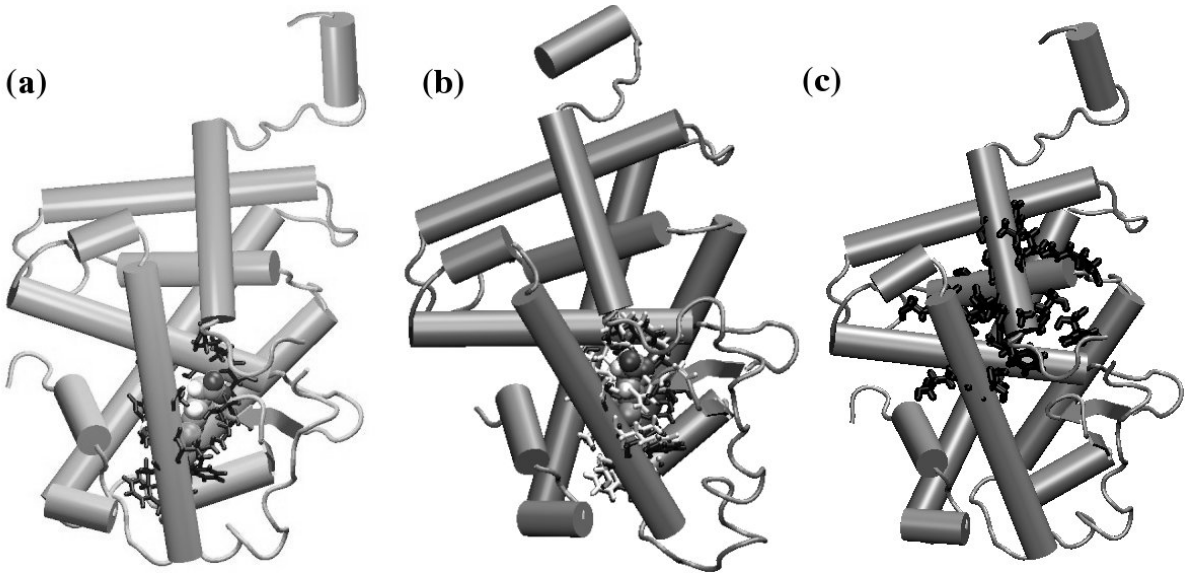

Figure 2: Identification of a pharmacophore with GN alignment: (a) The structure TRα with its pharmacophore (black). (b) The structure of TRβ with its pharmacophore (light grey) and the aligned α pharmacophore (black). (c) The best alignment found for the pharmacophore.

DNA binding site example and 0.059 in the second case.

The fact that the optimal solution could not be obtained in this case suggests that specific initial-point heuristics may be necessary for pharmacophore identification.

5.3.2 Flexible Ligand Alignment

In order to illustrate the capability of the **GN** method to deal with flexibility or other transformations, we show here a flexible ligand alignment. The ligands are T3 (the Thyroid Hormone) and a drug candidate with similar function, the KB-141 ligand [32]. The structures of these ligands are shown in Figures 3(a) and (b). This example was run without the **INITIAL** heuristic, since sequentiality is not relevant for general ligand alignments. We employed 200 trials of the **GN** algorithm with $n_{guess} = n_A$ and starting from random initial points.

The flexibility of the ligand was represented by three internal rotations of chemical relevance. The bonds that define the axes of rotation are indicated in Figure 3(a) by R1, R2 and R3. The results of the alignment are listed in Table 5.

As expected, the introduction of flexibility improves the quality of the alignment. The merit function and rmsd decrease as the number of rotations is increased. The improvement of the alignment can also be observed in Figures 3(c) to (f). On the other hand, the introduction of internal rotations clearly adds difficulties for the obtention of the global solution. As can be seen in Table 5, the number of successful trials (in which the global solution was found) decreases from 54 to 13 by the introduction of one rotation. In this example, the introduction of more rotations did not decrease further the probability of finding the global minimizer, probably because the molecules are small.

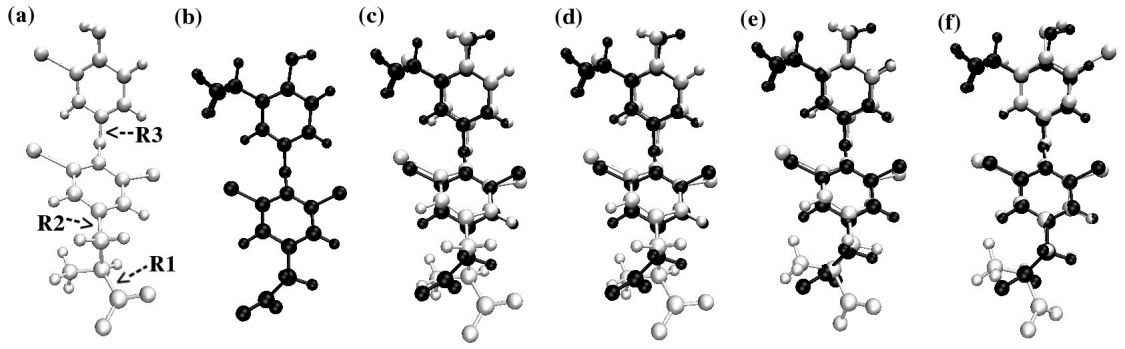

Figure 3: Flexible ligand alignment. (a) Structure of the Thyroid Hormone. The allowed rotations are indicated. (b) Structure of the drug candidate KB-141. Alignments performed with (c) no rotations, (d) R1 allowed, (e) R1 and R2 allowed, (f) R1, R2 and R3 allowed.

Rotations allowed	Best function value	rmsd	Successful trials
None	27.1069780	0.88	54
R1	26.9802612	0.87	13
R1, R2	21.8226606	0.78	13
R1, R2, R3	18.2595802	0.72	14

Table 5: Statistics of the flexible ligand alignments.

5.3.3 Cofactor Identification

Cofactors are molecules that are bound to enzimes and are important for their function. Different enzimes with similar functions may have similar cofactors. The identification of a cofactor within a protein structure may be useful for the analysis of protein function independently of the overall protein structure. In this example, we align the HEME group of the Human Hemoglogin (43 atoms) with the also HEME-containing protein complex of Sperm Whale Myoglobin (HEME group: 42 atoms, protein matrix: 154 Cα atoms). The HEME group of Hemoglobin is represented in Figure 4(a) and the Sperm Whale Myoglobin protein, containing the HEME group is represented in Figure 4(b).

In this case, since the HEME groups contains no sequence information, the **INITIAL** heuristic was not used. Random initial points were generated for 100 **GN** trials and we used $n_{guess} = 0.9n_A$. The correct alignment of the HEME groups was found 4 times, with an rmsd of 1.03Å. Figure 4(c) illustrates the alignment obtained. The total time of the alignment was 1.4s. Therefore, Algorithm **GN** is able to successfully identify the cofactor without any adaptation. On the other hand, as in the case of other non-sequential alignments, specialized heuristics could probably be used to increase the chances of obtaining the global solution.

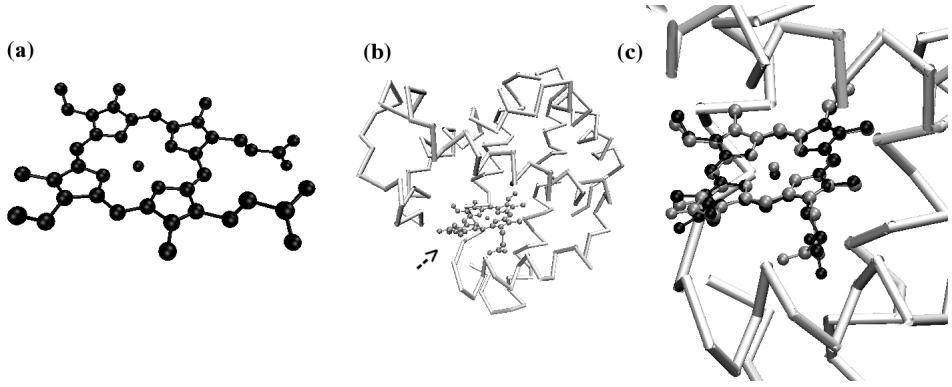

Figure 4: Alignment of HEME groups. (a) The HEME group of Human Hemoglobin. (b) The structure of Sperm Whale Myoglobin containing its HEME group (pointed by the arrow). (c) Detail of the best alignment obtained.

6 Concluding Remarks

In Structure Alignment one tries to find a transformation (movement) of the structure A that optimizes a similarity function with respect to the structure B. The choice of the similarity function depends on the problem under consideration. Moreover, similarity functions must be easy to compute.

A transformation (or movement) can be frequently parameterized with respect to a finite number of variables. Therefore, the similarity function depends on those variables and the Structural Alignment between A and B becomes an Optimization problem in finite dimension.

Numerical Continuous Optimization is a well developed area of Computational Mathematics. So, practical problems that can be reduced to the minimization of a continuous function have good chances to be solved efficiently. In particular, differentiability is a desirable property of the objective function that allows one to use well studied fast Newtonian or gradient-like methods.

The merit function f used in this paper is not differentiable. Instead, it is the minimum of a set of differentiable functions f_i which, in turn, do not need to be exhaustively computed. As a consequence, smooth descent methods using a single gradient per iteration can be defined for minimizing the similarity function. Global convergence in the usual sense of Nonlinear Programming can be proved. The fact that each f_i is a sum of squares motivated us to choose a variation of the Gauss-Newton method for the minimization process.

Ideally, we would like to define f_i as the sum of the squared distances corresponding to a bijection between a subset of the transformed A and a subset of B. However, such a function would not be easily computable. The compromise with computability led us to sacrifice bijectivity of the correspondence that defines f_i. In fact, we showed that the only f_i that needs to be explicitly computed is the one that defines the minimum of all the f_j's.

The Gauss-Newton-like algorithm defined for minimizing the similarity merit function f turned out to be very effective. It is remarkable that in *almost all* the iterations (involving thousands of **GN** executions) only one function evaluation was needed. This indicates that the **GN** quadratic model chosen always represents well the objective function and its minimization

leads to strong descent of f. Roughly speaking, the behavior of **GN** on these problems is similar to the one that would be observed if all the functions f_i were sums of squared *linear* functions.

Concerning the minimization of f, the real difficulty is the presence of many local minimizers. Since the number of parameters is usually small, this difficulty may be many times overcome using multistart techniques or other simple heuristics for initial approximations. When we have additional information about the solution, as in the case of Protein Alignment, specific heuristics may be defined for finding initial points that seem to work well. Experiments on Protein Alignment problems seem to indicate that the Gauss-Newton approach is reliable.

This research continues following at least two main directions. On one hand, we need to go deeper in the approach to specific problems, as Protein Alignment. Stronger heuristics must be used for initial approximations and the greedy post-processing strategy must be replaced by dynamic searches in the Protein-to-Protein distance matrix [5, 30]. Moreover, post-processing may become *middle-processing* without altering the general structure of the algorithm with possible improvement of the overall optimization method. Finally, the use of more powerful computer facilities will allow us to perform all-to-all comparisons and to address the MSTA (Multiple-Structure Alignment) problem [5, 25] in reasonable time.

On the other hand, more general problems should be addressed. The space of admissible movements by means of which flexibility is allowed may be expanded in many ways. For instance, more general internal rotations may be admitted and scale transformations may be incorporated in order to detect size-invariant similarities. Applications to object detection in 2D or 3D images are not excluded.

A Appendix: Protein Identities in the DALI Comparison

The 20 best alignments of DALI for the five examples reported here correspond to the following proteins. The order is from the best to the worst alignment obtained by DALI, as given in the website

`http://ekhidna.biocenter.helsinki.fi/dali/start.`

The code is the Protein Data Bank identifier, and the last letter of the code corresponds to the protein chain compared, when required. Note that the first protein of each group is the same as the reference protein.

For 1LBD: (1) 1LBD, Retinoid X Receptor; (2) 1MV9A, RXR Retinoid X Receptor; (3) 1DKFB, Retinoid X Receptor-Alpha; (4) 1PZLA, Hepatocyte Nuclear Factor 4-Alpha; (5) 2LBD, Retinoic Acid Receptor Gamma; (6) 1HG4A, Ultraspiracle; (7) 1G2NA, Ultraspiracle Protein; (8) 1PK5A, Orphan Nuclear Receptor NR5A2; (9) 1QKNA, Estrogen Receptor Beta; (10) 1LV2A, Hepatocyte Nuclear Factor 4-Gamma; (11) 1NQ2A, Thyroid Hormone Receptor Beta-l; (12) 1QKUA, Estradiol Receptor; (13) 1IE9A, Vitamin D3 Receptor; (14) 1L2JA, Estrogen Receptor Beta; (15) 1N81A, Plasmodium Falciparum Gamete Antigen 27; (16) 1EYXL, R-Phycoerythrin; (17) 1A87, Colicin N; (18) 1DI1A, Aristolochene Synthase; (19) 1Q8CA, Hypothetica Protein MG027; (20) 1IV8A, Maltooligosyl Trehalose Synthase.

For 1NPD: (1) 1NPDB, Hypothetical Shikimate 5-Dehydrogenase; (2) 1NVTA, Shikimate Dehydrogenase (AROE or MJ1084) in complex with NADP+; (3) 1WXDA, Shikimate 5-Dehydrogenase;

(4) 1NPYB, Hypothetical Shikimate 5-Dehydrogenase; (5) 1P74A, Shikimate 5-Dehydrogenase; (6) 1NYTA, Shikimate 5-Dehydrogenase; (7) 1LUAA, Methylene Tetrahydromethanopterin Dehydrogenase; (8) 1VL6A, Malate Oxidoreductase; (9) 1EE9A, 5,10-Methylenetetrahydrofolate Dehydrogenase; (10) 1QPJA, Glutamyl-Trna Reductase; (11) 1SAYA, L-Alanine Dehdrogenase; (12) 1VLVA, Ornithine Carbamoyltransferase; (13) 1A7AA, S-Adenosylhomocysteine Hydrolase; (14) 1B0AA, Fold Bifunctional Protein; (15) 1F8GA, Nicotinamide Nucleotide Transhydrogenase; (16) 1HYLA, N-Acetyl-L-Ornithine Carbamoyltransferase; (17) 1GQ2A, Malic Enzyme; (18) lML4A, Aspartate Transcarbamoylase; (19) 1DIAB, Methylenetetrahydrofolate; (20) 1PG5A, Aspartate Carbamoyltransferase.

For 1KTB: (1) 1KTBA, Alpha-N-Acetylgalactosaminidase; (2) 1R46B, Alpha-Galactosidase A; (3) 1UASA, Alpha-Galactosidase; (4) 1SZNA, Alpha-Galactosidase; (5) 1JIBA, Neopullulanase; (6) 1MXGA, Alpha Amylase; (7) 1M7XB, 1,4-Alpha-Glucan Branching Enzyme; (8) 1JG9A, Amylosucrase; (9) 1UOK, Oligo-1,6-Glucosidase; (10) 2AAA , Acid Alpha-Amylase; (11) 1QHPA, Alpha-Amylase; (12) 1CYG, Cyclodextrin Glucanotransferase; (13) 1LWJA, 4-Alpha-Glucanotransferase; (14) 1QI4A, Exo-Maltotetraohydrolase; (15) 1D3CA, Cyclodextrin Glycosyltransferase; (16) 9CGTA, Cyclodextrin Glycosyltransferase; (17) 2TAAA, Taka-Amylase A; (18) 1IV8A, Maltooligosyl Trehalose Synthase; (19) 1AVAA, Barley Alpha-Amylase 2; (20) 1GJWA, Maltodextrin Glycosyltransferase.

For 1UKX: (1) 1UKXA, GCN2 EIF2α Kinase; (2) 1RU0A, DCOH-like Protein DCOHM; (3) 1NH2B, Transcription Initiatition Factor TFIID; (4) 1H3QA, Sedlin (5) 1JM0A, Four-Helix Bundle Model; (6) 1H3LB, RNA Polymerase Sigma Factor 4; (7) 1WJ2A, Probable Wrky Transcription Factor 4; (8) 1Q2HA, Adaptor Protein with Pleckstrin Homology; (9) 1SKVA, Hypothetical 7.5 KDA Protein; (10) 1IURA, KIAA0730 Protein (11) 1Y2OA, BAI1-Associated Protein 2 Isoform 1; (12) 1W0BA, Alpha-Hemoglobin Stabilizing Protein; (13) 1AVOJ, 11S Regulator; (14) 1OV2A, Alpha-2-Macroglobulin Receptor Associated; (15) 1P84F, Ubiquinol-Cytocrome C Reductase; (16) 1WCRA, PTS System; (17) 1LQ7A, Alpha3W; (18) 1M62A, Bag-Family Molecular Chaperone Regulator 4; (19) 1WPBA, Hypothetical Protein YFBU; (20) 1XVHA, Hypothetical Protein, similar to Streptococcal.

For 1M4U: (1) 1M4UL, Bone Morphogenetic Protein 7; (2) 3BMPA, Bone Morphogenetic Protein 2; (3) 1TFG, Transforming Growth Factor Type Beta 2; (4) 1S4YB, Activin Receptor Type IIB Precursor; (5) 1KLAA, Transforming Growth Factor-Beta 1; (6) 1NYUD, Activin Receptor; (7) 1FZVA, Placenta Growth Factor; (8) 1M4UA, Bone Morphogenetic Protein-7; (9) 1FL7B, Follicle Stimulating Protein Alpha Chain; (10) 1JPYA, Interleukin 17F; (11) 1HCFA, Neurotrophin-4; (12) 1AOCA, Coagulogen; (13) 1DZ7A, Chorionic Gonadotropin; (14) 1JMAB, Herpesvirus Entry Mediator; (15) 1CLZA, Beta2-Glycoprotein-I; (16) 1AFVH, Human Immunodeficiency Virus Type 1 Capsid; (17) 4DPVZ, Canine Parvovirus Strain D Viral Protein 2; (18) 1QZLA, Neural Cell Adhesion Molecule 1; (19) 1CRUB, Soluble Guinoprotein Glucose Dehydrogenase; (20) 1JMZB, Amine Dehydrogenase.

References

[1] R. Andreani, C. Dunder and J. M. Martínez, Order-Value Optimization: formulation and solution by means of a primal Cauchy method, *Math. Methods Oper. Res.* 58 (2003) 387-399.

[2] R. Andreani, J. M. Martínez, L. Martínez and F. Yano. *Low Order-Value Optimization and Applications*. Technical Report MCDO 051013, Department of Applied Mathematics, State University of Campinas, Brazil, 2005.

[3] R. Andreani, J. M. Martínez, M. Salvatierra and F. Yano, Quasi-Newton methods for order-value optimization and value-at-risk calculations, *Pac. J. Optim.* 2 (2006) 11-33.

[4] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov and P.E. Bourne, The Protein Data Bank, *Nucleic Acids Res.* 28 (2000) 235-242.

[5] G. Brinkmann, A. W. M. Dress, S. W. Perrey and J. Stoye, Two applications of the Divide & Conquer principle in the molecular sciences, *Math. Program.* 79 (1997) 71-97.

[6] J. E. Chreneik, A. B. Burgin, Y. Pommier, L. Stewart and M. R. Redinbo, Structural impact of the Leukemia Drug 1-Beta-D-Arabinofuranosyleytosine (Ara-C) on the Covalent Human Topoisomerase I-DNA Complex, *J. Biol. Chem* 278 (2003) 12461-12466.

[7] G. H. Golub and Ch. F. Van Loan, *Matrix Computations*, Johns Hopkins, Baltimore, 1983.

[8] L. Holm and J. Park, DaliLite workbench for protein structure comparison, *Bioinformatics* 16 (2000) 566-567.

[9] L. Holm and C. Sander, Protein structure comparison by alignment of distance matrices, *J. Mol. Biol.* 233 (1993) 123-138.

[10] L. Holm and C. Sander, Mapping the Protein Universe, *Science* 273 (1996) 595–602.

[11] W. Kabsch, A discussion of the solution for the best rotation to relate two sets of vectors, *Acta Crystallog. A* 34 (1978) 827-828.

[12] S. K. Kearsley, On the orthogonal transformation used for structural comparisons, *Acta Crystallog. A* 45 (1989) 208-210.

[13] K. Kedem, L. P. Chew and R. Elber, Unit-vector RMS (URMS) as a tool to analyze molecular dynamics trajectories, *Proteins* 37 (1999) 554-564.

[14] G. J. Kleywegt, Use of non-crystallographic symmetry in protein structure refinement, *Acta Crystallog. D* 52 (1996) 842-857.

[15] R. Kolodny, P. Koehl and M. Levitt, Comprehensive evaluation of protein structure alignment methods: scoring by geometric measures, *J. Mol. Biol.* 346 (2005) 1173-1188.

[16] R. Kolodny and N. Linial, Approximate protein structural alignment in polynomial time, *P. Natl. Acad. Sci. USA* 101 (2004) 12201-12206.

[17] E. Krissinel and K. Henrick, Protein structure comparison in 3D based on secondary structure matching (SSM) followed by C-alpha alignment, scored by a new structural similarity function, In *Proceedings of the First international Conference on Molecular Structural Biology. Vienna, September 3-7* (A. J. Kungl and P. J. Kungl, eds.), 2003.

[18] D. Lupyan, A. Leo-Macias and A. R. Ortiz, A new progressive-iterative algorithm for multiple structure alignment, *Bioinformatics* 21 (2005) 3255-3263.

[19] B. Needleman and C. D. Wounsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, *J. Mol. Biol.* 48 (1970) 443-453.

[20] K-D. Neubert, Flashsort1 Algorithm, *Dr. Dobb's Journal* 23 (1998) 123-124.

[21] J. Nocedal and S. J. Wright. *Numerical Optimization*, Springer, New York, 1999.

[22] A. M. Nunes, R. Aparicio, M. A. M. Santos, R. V. Portugal, S. M. G. Dias, F. A. R. Neves, L. A. Simeoni, J. D. Baxter, P. Webb and I. Polikarpov, Crystallization and preliminary X-ray diffraction studies of isoform alpha 1 of the human thyroid hormone receptor ligand-binding domain. *Acta Crystallog. D* 60 (2004) 1867-1870.

[23] A. R. Ortiz, C. E. Strauss and O. Olmea, MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison, *Protein Sci.* 11 (2002) 2606-2621.

[24] M. R. Redinbo, L. Stewart, P. Kuhn, J. J. Champoux and W. G. J. Hol, Crystal structures of Human Topoisomerase I in Covalent and Noncovalent complexes with DNA, *Science* 279 (1998) 1504-1513.

[25] E. Sandelin, Extracting multiple structural alignments: a comparison of a rigorous and a heuristic approach, *Bioinformatics* 21 (2005) 1002-1009.

[26] I. N. Shyndialov and P. E. Bourne, Protein structure alignment by incremental combinatorial extension (CE) of the optimal path, *Protein Eng.* 1 (1998) 739-747.

[27] S. Subbiah, D. V. Laurents and M. Levitt, Structural similarity of DNA-binding domains of bacteriophage repressors and the globin core, *Curr. Biol.* 3 (1993) 141-148.

[28] W. R. Taylor, Protein structure alignment using iterated double dynamic programming, *Protein Sci.* 8 (1991) 654-665.

[29] W. R. Taylor and C. A. Orengo, Protein structure alignment, *J. Mol. Biol.* 208 (1989) 1-22.

[30] R. A. Wagner and M. J. Fischer, The string-to-string correction problem, *Journal of the ACM* 21 (1974) 168-173.

[31] Eric W. Weisstein, 'Rodrigues' Rotation Formula, *MathWorld–A Wolfram Web Resource* http://mathworld.wolfram.com/RodriguesRotationFormula.html.

[32] L. Ye, Y. L. Li, K. Mellstrom, C. Mellin, L. G. Bladh, K. Koehler, N. Garg, A. M. G. Collazo, C. Litten, B. Husman, K. Persson, J. Ljunggren, G. Grover, P. G. Sleph, R. George R and J. Malm, Thyroid receptor ligands. 1. Agonist ligands selective for the thyroid receptor beta(1), *J. Med. Chem.* 46 (2003) 1580-1588.

[33] J. Zhu and Z. Weng, A novel protein structure alignment algorithm, *Proteins* 58 (2005) 618-627.