Structural Alignment of Large-Size Proteins via Lagrangian Relaxation

[Extended Abstract] *

Al berto Caprara
DEIS, University of Bologna
Viale Risorgimento 2
40136 Bologna, Italy
acaprara@deis.unibo.it

Giuseppe Lancia
DEI, University of Padova
Via Gradenigo 6-A
35131 Padova, Italy
lancia@dei.unipd.it

ABSTRACT

We illustrate a new approach to the Contact Map Overlap problem for the comparison of protein structures. The approach is based on formulating the problem as an integer linear program and then relaxing in a Lagrangian way a suitable set of constraints. This relaxation is solved by computing a sequence of simple alignment problems, each in quadratic time, and near-optimal Lagrangian multipliers are found by subgradient optimization. By our approach we achieved a substantial speedup over the best existing methods. We were able to solve optimally for the first time instances for PDB proteins with about 1000 residues and 2000 contacts. Moreover, within a few hours we compared 780 pairs in a testbed of 40 large proteins, finding the optimal solution in 150 cases. Finally, we compared 10,000 pairs of proteins from a test set of 269 proteins in the literature, which took a couple of days on a PC.

1. INTRODUCTION

Comparing protein structures is a problem of paramount importance in structural genomics. In fact, its solution is instrumental to the solution of other fundamental questions such *as* (i) *Protein* Function Determination and Drug *De*sign; (ii) *Assessment of Fold Prediction;* (iii) *Protein Clus*tering. We now briefly elaborate on each of these issues.

(i) The 3D structure of a protein determines for the largest part how the protein functions and interacts with other molccules [4]. Hence, the function of a new protein can oftentimes be established by comparing its structure to some known ones. Further, the drug design problem consists in the discovery of ad hoc peptides with structure complementary to that of the proteins they must inhibit or enhance [3].

*Research partially supported by CNR and MIUR through project PRIN.

RECOMB '02, April 18-21, 2002 Washington, D.C., USA
 ISBN I-581 13-498-3-02/04 . ..$5.00

(ii) Given a set of "tentative" folds for a protein, and a "correct" one (determined experimentally) the structure alignment problem must be solved to find out which of the guesses is the closest to the right answer. This is the problem faced, e.g., by the CASP (Critical Assessment of Structure Prediction, [26]) jurors, in the international biannual structure–prediction exercise, where the large number of predictions submitted makes the use of sound algorithms for structure comparison a compelling need. In particular, such algorithms are at the base of CAFASP [9], a recent fully automated CASP version.

(iii) Given a set of proteins and their structures, it is possible to group the proteins in families based on structure similarity. In the last years, a number of protein structures classification servers have been created which organize the PDB proteins in related families, such as SCOP [27], FSSP [17], HOMSTRAD [25], CAMPASS [30], and others.

Pairwise structure comparison requires a structure similarity scoring scheme that captures the biological relevance of the chemical and physical steric constraints involved in molecular recognition. Nowadays, the most used scoring schemes are based on three themes: *RMSD (Root* Mean Square *Deviation) of* Rigid-Body Superposition [19], Distance *Map* Simi*larity* [18] and Contact Map Overlap (CMO) [11]. While the first two measures require a preset alignment for the equivalenced residues in the two proteins to be given, the CMO measure does not require a preset alignment. The CMO is based on the basic notion of *contact* between two residues, which is of fundamental statistical mechanics and chemical significance, and at the core of many scoring schemes used in applications of protein structure analysis and simulation. The CMO scoring scheme was extensively studied and empirically validated by the Godzik-Skolnick group at the Scripps Institute [12, 11].

Almost all algorithms for the comparison problem are simple heuristics of many sorts. Many of them try to optimize the RMSD (Root Mean Square Deviation) of a subset of residues in the first structure and a subset in the second structure (an example of such algorithm is MAXSUB [29]). These measures, however, have many recognized flaws; most notably, they are a poor indicator of similarity for structures with a good local agreement on some well-defined domains.

DALI [18], on the other hand, is an example of heuristic algorithm based on distance map similarity. For other examples, one can look at the excellent survey of structure comparison algorithms [21].

As far as exact algorithms are concerned, until last year the situation was that "In structure comparison, we do not even have an algorithm that guarantees an optimal answer for pairs of structures..." [8]. In RECOMB 2001, Lancia, Carr, Walenz and Istrail [20] proposed the first rigorous algorithm for structure comparison, a branch-and-cut procedure for finding the largest alignment of two contact maps. The approach, based on Integer Linear Programming, exploited an exponential number of constraints to obtain tight upper bounds, then used in a branch-and-bound search. The exponential number of constraints was dealt with only implicitly, via a *Separation Algorithm* which dynamically generates only the constraints that are needed.

Although this algorithm was the first rigorous method for aligning structures, capable of finding solutions provably optimal or within a provable gap from optimality, the approach has two main drawbacks. Both drawbacks constitute a serious bottlenck to its practical use, but the first issue is intrinsic to the algorithm, while the second is mainly a problem of software engineering. First and foremost, its USC must be limited to fairly small proteins, i.e., with up to about 70 residues and roughly twice as many contacts, or otherwise the amount of time and memory required for the solution become simply infcasiblc, even on a very powerful mainframe. Second, the algorithm is quite complex to implement, and relics on the existence and availability of an expensive third-party component, i.e., a professional LP-solver. For these reasons, as of today, the algorithm has not been made available on a web server, as was in the authors' intentions, but is implemented only in a prototype form.

In this paper we overcome both these problems, and describe an algorithm for the contact map alignment problem which is much faster than [20], works on large proteins, is easy to implement and does not require an external LP solver. Similarly to the previous method, our new algorithm can be used either to find the optimal solution (via branch and bound), or a good feasible solution together with a provable certificate of near optimality, provided via an upper bound to the true optimum.

Our algorithm is based on a stronger integer programming formulation than that proposed in [20]. A Lagrangian Relaxation of the model is then solved via Subgradient Optimization, to obtain tight bounds as well as good feasible solutions. Lagrangian Relaxation is a powerful technique for solving very large Combinatorial Optimization problems [28, 10]. While Branch-and-Cut has already been employed for both structural as well as sequence alignment problems (see, e.g., [22]), to the best of our knowledge this is the first time that Lagrangian Relaxation techniques are used for an alignment algorithm of any type.

The algorithm we propose is based on a similar approach which was successfully used for other Binary Quadratic Programming problems, such as the Quadratic Assignment Problem [7] or the Quadratic Knapsack Problem [6]. These problems bear many similarities with the structure alignment problem. In particular, there are profits p_{ij} in the objective function which are attained when two binary variables x_i and x_j are *both* set to 1 in a solution. Analogously, in the alignment problem, we may have a profit in choosing to align two specific residues of the proteins and some other two.

We tested our algorithm on real proteins from PDB. We were able to solve optimally for the first time alignment problems for proteins with about 1000 residues and 2000 contacts, whereas the largest instances solved to proved optimality by the method of [20] had about 80 residues and 150 contacts. With respect to this method, ours is capable of computing upper bounds of similar quality within computing times that are orders of magnitude smaller. For this reason, we were able to compare all 780 pairs in a testbed of 40 large proteins, suggested by Jeffery Skolnick, within few hours. In 150 cases, our method found the optimal solution. Moreover, we considered the 269 proteins mentioned in [20] and compared, within the weekend, a wide fraction of the (about) 36000 corresponding pairs.

On the negative side, it has to be remarked that the optimal solution of instances associated with substantially different proteins seems completely out of reach not only for our algorithm (or for the algorithm of [20]) but also with the other methods currently known in Combinatorial Optimization. Such a situation would be analogous to the case of the Quadratic Assignment Problem, for which instances with 40 nodes (the "counterpart" of residues) are quite far from being solved to optimality.

1.1 Contact Maps

A *contact* map [15] of a folded protein of n residues is a 0-1, $n \times n$ matrix C, whose l-elements correspond to pairs of amino acids in three-dimensional "contact". A contact can be defined in many ways. Typically [24], one considers $C_{ij} = 1$ when the distance of two heavy atoms, one from the i-th aminoacid and one from the j-th aminoacid of the protein, is smaller than a given threshold (e.g., 5Å). The framework of the contact map representation of proteins is very appealing, since this intuitive and simple representation is already complex enough to capture the most important properties of the folding phenomenon. It has been shown that it is relatively easy to go from a map to a set of possible structures to which it may correspond [15, 31]. This result has opened the possibility of using contact maps to predict protein structure from sequence, by predicting contact maps from sequence instead.

Besides their use for protein fold prediction, contact maps are exploited to compare 3D structures. The basic idea is that, when two structures are similar, we should expect their contact maps to be similar as well. Hence, one can use an indirect method for structure comparison, i.e., contact map comparison.

We can regard the contact map of a protein p as the adjacency matrix of a graph G,. Each residue is a node of G,, and there is an edge between two nodes if the the corresponding residues are in contact. The *Contact Map Overlap* problem, calls for determining an ordered (i.e., noncrossing)

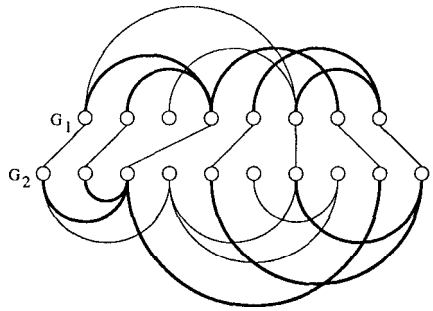

Figure 1: An alignment of value 5

alignment of some residues in the first protein (nodes in G_1) and the second protein (nodes in G_2) which highlights the largest set of common contacts as follows: The value of the alignment is the number of contacts (i.e., edges) in the first map whose endpoints are aligned with residues that are also in contact in the second map. This value is called the overlap for the two proteins, and the optimization problem is to find the maximum overlap. Figure 1 shows two contact maps and a feasible alignment.

This measure was introduced in [12], and its optimization was proved NP-hard in [13], thus justifying the use of sophisticated heuristics or enumerative methods.

1.2 CMO Optimization

In RECOMB 2001, [20] proposed an algorithm for the CMO problem, (called LCWI in the sequel), which can find the optimal alignment of two contact maps. Their approach is based on an *Integer* Linear Programming (ILP) formulation of the problem, solved by Branch-and-Cut. The formulation exploits an exponential number of clique *inequality*-cuts. These are constraints saying that, out of a number of conflicting possible ways of aligning two residues, at most one way can be chosen in a solution. The exponential number of constraints is dealt with implicitly, via a Separation Algorithm which dynamically generates only the constraints that are needed. The solution of the associated (exponentially large) Linear Program (LP) yields an upper bound on the optimal CMO value.

The bound used by LCWI is not the strongest possible for the CMO problem, and in fact in this paper we show how to improve it. Due to the large number of variables/constraints in the LPs and to the use of a bound which is not the most effective possible, the algorithm LCWI can in fact be used only to align proteins of up to 70-80 residues. Although a number of proteins in the pdb (roughly 300) fall within this range size, most proteins are beyond the capability of this algorithm, which drastically limits its interest for practical applications. Furthermore, a bottleneck in the approach used in LCWI is that, for the solution of many, large LPs, it requires an external software, taking a long time to run, while the programmer has no control over this phase of the solution process.

For these reasons, in this paper we have tried to develop a new approach for the CMO problem, pursuing the following goals: (i) the approach should be a rigorous procedure, based on sophisticated mathematical models, which can lead to the optimal solution of the problem. Furthermore, it should be capable also of finding good feasible solutions, with quality certified by an upper bound to the optimum, computed by the procedure. (ii) It should be able to solve problems for proteins of size considerably larger than before, say 200-300 residues within some minutes on, e.g., a fast PC. (iii) It should not require the solution of LPs, because they are too expensive to afford. The whole procedure should be easy to implement, e.g., in the C programming language.

In order to meet these goals, we have adopted a Lagrangian Relaxation (LR) approach, where the optimal Lagrangian multipliers are found by Subgradient Optimization. The theory of Lagrangian Relaxation is a well established branch of Combinatorial Optimization, and has been used successfully in a large number of applications, in different domains [28]. Nowadays, LR is the most successful tool to tackle very large problems. For instance, the state of the art algorithms for the well known Set Covering Problem, the Combinatorial Optimization problem most frequently solved in real-world applications, are based on LR [5]. These algorithms are capable of finding near-optimal solutions to instances with millions of variables and thousands of constraints within minutes on a PC. However, to the best of our knowledge, this is the first time that a similar approach is used for an alignment problem arising in Computational Molecular Biology.

The LR approach is particularly well suited for those cases in which the formulation of a problem consists of two sets of constraints: a set of "nice" constraints and a set of "bad" constraints, whose removal makes the resulting problem, called the Lagrangian relased problem, easily solvable. The strategy then consists in removing the bad constraints from the formulation and putting them in the objective function, each weighed by some coefficient (Lagrangian multiplier). The weight for a constraint represents a penalty which is incurred by a solution which does not satisfy that constraint. To any choice of weights corresponds a (relatively easy) problem whose solution yields a bound to the original problem. The core question of LR is then to determine the optimal weights, i.e., the Lagrangian multipliers yielding the best bound. In most cases, the determination of these multipliers is equivalent to solving a suitable LP, which would be too time consuming in practice. On the other hand, near-optimal multipliers can be found by a simple iterative procedure known as subgradient optimization, in which, at each iteration, the Lagrangian relaxed problem is solved and the multipliers are updated based on the corresponding solution.

Besides yielding an upper bound on the optimal solution of the original problem, the Lagrangian multipliers (and the associated modified costs/profits in the objective function) can be used to drive simple heuristic procedures (in most cases of greedy nature). These procedures typically produce substantially different solutions for different Lagrangian multipliers. Accordingly, if they are embedded within an iterative procedure to define near-optimal multipliers, namely they are called at each iteration with the current multipliers, the best solution found over all iterations tends to be

near-optimal.

2. MATHEMATICAL FORMULATION AND LAGRANGIAN RELAXATION

We can rephrase the CMO problem in graph-theoretic language as follows: We are given two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $n_i = |V_i|$ and $m_i = |E_i|$, for i = 1, 2. A total order is defined on $V_1 = (1,\ldots, n_1\}$ and $V_2 = \{1,. , , n_2\}$. For each $j \in V_i$, we let $N_i(j) := \{k \in V_i :$ (j, $k) \in E_i\}$ denote the set of neighbors of j in G_i.

A non-crossing map of V_1 in V_2 is defined by any two subsets of the same size k, $\{i_1,. , i_k\} \subseteq V_1$ and $\{j_1,\ldots, j_k\} \subseteq V_2$, where $i_1 < i_2 < . < i_k$ and $j_1 < j_2 < \ldots < j_k$. In this map, j_h is the image of i_h for $1 \leq h \leq k$. Two edges $(h_1, l_1) \in E_1$ and $(h_2, l_2) \in E_2$ are shared by the map if there are $p, q \leq k$ s.t. $h_1 = i_p$, $l_1 = i_q$, $h_2 = j_p$ and $l_2 =$ j,. Each pair of shared edges contributes a *sharing* to the objective function. The problem consists in finding the non-crossing map which maximizes the number of sharings.

Let $L = V_1$ x V_2. To avoid confusion with the edges in G_1 and G_2, we call the elements in L lines. Clearly, lines represent mapping of nodes in V_1 into nodes in V_2. We say that two lines 1 = (i_1, i_2) and m = (j_1, j_2) in L are **compatible** if it is feasible to map i_1 to i_2 and j_1 to j_2 at the same time, i.e., if either $i_1 < j_1$ and $i_2 < j_2$ or $i_1 > j_1$ and $i_2 > j_2$, incompatible otherwise – graphically, incompatibility corresponds to the two lines crossing or touching. We let $\mathcal{I}$ denote the (exponentially large) collection of all maximal sets of pairwise incompatible lines $I \subseteq L$,

For l = (i, j) $\in L$ we introduce a binary variable x_l equal to 1 if and only if i is mapped to j in the noncrossing matching. We also let X denote be the set of incidence vectors $x \in \{0,1\}^L$ of all noncrossing matchings in L. For $l = (i_1, i_2)$, $m = (j_1, j_2) \in L$, we let $a_{lm} = 1$ if $(i_1, j_1) \in E_1$ and $(i_2, j_2) \in E_2$, $a_{lm} = 0$ otherwise. The objective function in our first formulation contains terms of the form $a_{lm}x_l x_m$, indicating that a contribute of a sharing is achieved if the three terms in the product take the value 1, i.e., a profit of 1 is attained if both x_l and x_m are equal to 1. In fact, in order to illustrate our method, it is necessary to introduce separately products $x_l x_m$ and $x_m x_l$ in the objective function. To this end, we define separate profits b_{lm} for $x_l x_m$ and b_{ml} for $x_m x_l$. Of course, we have to ensure $b_{lm} + b_{ml} = a_{lm} (= a_{ml})$ for all $l, m \in L$. For instance, a valid (initial) choice is $b_{lm} = b_{ml} = a_{lm}/2$. Later, it will be clear how crucial is for our method to split a_{lm} into b_{lm} and b_{ml} "optimally", even if we will not fix this splitting a priori, but rather use a sort of iterative method to find it.

The problem can now be stated as

$$\max \sum_{l \in L} \sum_{m \in L} b_{lm} x_l x_m \tag{1}$$

subject to

$$\text{x} \in \text{x}. \tag{2}$$

Actually, we know how to represent X with linear constraints. Recalling the definition of $\mathcal{I}$, (2) is equivalent to

$$\sum_{l \in I} x_l \leq 1, \ \forall I \in \mathcal{I} \tag{3}$$

$$x \geq 0, \text{ integer}. \tag{4}$$

Problem (1), (3), (4) is a Binary Quadratic Program. Note in passing that, although the convex hull of the vectors in X is given by (3) and the nonnegativity constraints and therefore the integrality conditions could be removed if (1) were linear, in this case we must keep these conditions (it is false in general that the maximization of a quadratic function over a polytope has a solution which is a vertex of the polytope).

2.1 Linearization and strengthening

To linearize (1) we use a standard technique, introducing variables y_{lm}, for $l, m \in L$, and replacing the product $x_l x_m$ by y_{lm}. Note that we introduce separately y_{lm} and y_{ml}. (In practice, the number of y variables that we will have to introduce explicitly is much smaller, namely $2|E_1||E_2|$, as it can be assumed without loss of generality that the remaining variables take the value 0. However, for the sake of illustration we will describe the model as if all the y variables were present.) The new (linear) objective function becomes

$$\max \sum_{l \in L} \sum_{m \in L} b_{lm} y_{lm}.$$

Moreover, the nonlinear relation $y_{lm} = x_l x_m$ can be replaced by the linear constraints

$$y_{lm} \leq x_m, \ \forall l, m \in L \tag{5}$$

$$y_{lm} = y_{ml}, \ \forall l, m \in L, l < m, \tag{6}$$

where "<" denotes an (arbitrarily defined) total ordering of the lines in order to avoid repeating the same constraint twice. In particular, note that no condition is required to enforce y_{lm} (and y_{ml}) to 1 if both x_l and x_m are 1, since the maximization will guarantee this condition (actually, if $a_{lm} = 0$, $x_l = x_m = 1$ and $y_{lm} = y_{ml} = 0$, setting $y_{lm} = y_{ml} = 1$ yields a feasible solution of the same value). In fact, we do not USC (5) but stronger linear conditions. For this purpose, we adopt a standard procedure used in the convexification of ILPs [1, 23, 2].

If we multiply a constraint (3) associated with set I by x_m for some m $\in L$ and replace $x_l x_m$ by y_{lm}, we get

$$\sum_{l \in I} y_{lm} \leq x_m.$$

Note that, if $m \in I$, we can write x_m instead of y_{mm}, as the variables are restricted to be binary, getting $\sum_{l \in I \setminus \{m\}} y_{lm} \leq 0$, i.e., the trivial condition that all y_{ml} must be 0 if 1 and m are incompatible. By doing the above for all constraints in (3) and variables x,, m $\in L$, we get our new model:

$$\max \sum_{l \in L} \sum_{m \in L} b_{lm} y_{lm} \tag{7}$$

subject to

$$\sum_{l \in I} x_l \leq 1, \ \forall I \in \mathcal{I} \tag{8}$$

$$\sum_{l \in I} y_{lm} \leq x_m, \ \forall I \in \mathcal{I}, \ m \in L \tag{9}$$

$$y_{lm} = y_{ml}, \forall l, m \in L, l < m \tag{10}$$

$$x, y \geq 0, \text{ integer.} \quad (11)$$

Note that relations (5) are not required, as their are implied by (9). An interesting property of the above formulation is that the problem obtained by removing constraints (10) is as difficult as the maximization of a linear function subject to (2) (or, equivalently, (3) and (4)). This property is common to other binary quadratic programs, such as the Quadratic Assignment Problem [7] and the Quadratic Knapsack Problem [6]. Below, we illustrate the situation in detail for our problem.

2.2 Decomposition and Lagrangian relaxation

PROPOSITION 1. *The problem* defined *by (7), (8), (9),* and (11) can *be solved in* $O(|E_1||E_2|)$ *time.*

PROOF. After the removal of equations (10), each variable y_{lm} appears only in the constraints (9) associated with x_m (besides being constrained to be nonnegative and integer by (11)). For each m $\in L$, this implies that, if variable x_m takes the value 0 all variables y_{lm} take the same value, whereas, if variable x_m takes the value 1, the optimal choice of y_{lm} for $1 \in$ L amounts to solving the following:

$$\max \sum_{l \in L} b_{lm} y_{lm} \quad (12)$$

subject to

$$\sum_{l \in I} y_{lm} \leq 1, \ \forall I \in \mathcal{I} \text{ such that } m \notin I \quad (13)$$

$$\sum_{l \in I} y_{lm} \leq 0, \ \forall I \in \mathcal{I} \text{ such that } m \in I \quad (14)$$

$$y \geq 0, \text{ integer.} \quad (15)$$

In other words, the profit achieved if $x_m = 1$, say p_m, is given by the optimal solution of (12)–(15). This is a simple alignment problem that can be solved in quadratic time by dynamic programming, as explained below. Once profits p_m have been computed for all m $\in$ L, we can find the optimal solution to our problem (without (10)) by solving

$$\max \sum_{m \in L} p_m x_m \quad (16)$$

subject to

$$\sum_{l \in I} x_m \leq 1, \ \forall I \in \mathcal{I} \quad (17)$$

$$x \geq 0, \text{ integer,} \quad (18)$$

which is again an alignment problem.

Overall, an optimal solution $(\bar{x}, \bar{y})$ of the relaxed problem is obtained by:

- (i) For each m $\in$ L, computing an optimal solution $\hat{y}_{lm}$ ($l \in$ L) to problem (12)–(15), letting p_m be the associated profit;
- (ii) Computing an optimal solution $\bar{x}$ to problem (16)-(18);
- (iii) Letting $\bar{y}_{lm} := \hat{y}_{lm} \cdot \bar{x}_m$ (l, m $\in$ L).

In particular, note that variable y_{lm} takes the value 1 in the solution of the overall problem if and only if it takes the value 1 in the solution of (12)–(15) and x_m takes the value 1 in the solution of (16)–(18).

We conclude the proof by analyzing the running time of the method. Of course, we can explicitly consider y_{lm} with l = (ii, i_2) and m = (j_1, j_2) only if l and m are compatible, (ii, j_1) $\in E_1$ and (i_2, j_2) $\in E_2$, since the other y_{lm}s can be assumed to be 0 without loss of generality. Accordingly, for each m = (i, j) $\in L$ we can solve (12)–(15) by simply considering the subgraphs of G_1 and G_2 induced, respectively, by $N_1(i)$ and $N_2(j)$ – the sets of neighbors of i and j. For these subgraphs, we have to find the largest-weight (w.r.t. weights b) noncrossing matching, without taking any line incompatible with m. This means that we should find (separately) the largest-weight (w.r.t. weights b) noncrossing matching among the "left" neighbors of i and j as well as among the "right" neighbors of i and j. This *can* be done in time $O(\ N_1(i)\ |N_2(j)|)$ by dynamic programming, see, e.g., [14]. For the solution of (16)–(18), we can again use dynamic programming, with running time $O(|V_1||V_2|)$. The overall complexity is

$$O(|V_1||V_2| + \sum_{i \in V_1} \sum_{j \in V_2} |N_1(i)||N_2(j)|) = O(|E_1||E_2|).$$

cl

Although the quality of the upper bound that we obtain by solving the relaxation without constraints (10) is typically quite poor, we can get much better bounds with a perfectly identical approach by relaxing these constraints in a Lagrangian way. This amounts to assigning a Lagrangian multiplier λ_{lm} to each constraint (10) and adding to the original objective function (7) a linear combination of constraints (10), each weighed by the associated Lagrangian multiplier, obtaining the Lagrangian objective function

$$\max \sum_{l \in L} \sum_{m \in L} b_{lm} y_{lm} + \sum_{l \in L} \sum_{\substack{m \in L: \\ l < m}} \lambda_{lm} (y_{lm} - y_{ml}) \quad (19)$$

The corresponding Lagrangian relaxed problem requires the maximization of (19) subject to (8), (9), and (11). The resulting value is an upper bound on the optimal value of (7)-(11) since, for each feasible solution of the latter, the contribution to (19) of the new term is null.

Defining for convenience $\lambda_{ml} := -\lambda_{lm}$ for $l < m$ (and $\lambda_{mm} := 0$), (19) can be rewritten as

$$\max \sum_{l \in L} \sum_{m \in L} (b_{lm} + \lambda_{lm}) y_{lm} \quad (20)$$

Then, one can see immediately that the effect of Lagrangian relaxation is to re-distribute the profit a_{lm} between the two terms in the objective function associated with y_{lm} and y_{ml}. Clearly, the Lagrangian relaxed problem can be solved as described in Proposition 1, after replacing b_{lm} by $b_{lm} + \lambda_{lm}$ for $l, m \in L$. Let U(X) be the resulting upper bound.

The best upper bound that can be obtained by Lagrangian relaxation is $U(\lambda^*) := \min_\lambda U(X)$, where λ^* denotes the best Lagrangian multipliers. By the above discussion, finding λ^* is the same as splitting profits a_{lm} between b_{lm} and b_{ml} so that the optimal value of the relaxation considered in Proposition 1 is minimized.

It is interesting to compare upper bound $U(\lambda^*)$ to the other upper bounds proposed for the problem by [20]. Both are associated with LP relaxations. The first LP relaxation involves the same variables as ours and, restated according to our notation, is given by (7) subject to (8), (10), the nonnegativity conditions on the variables, and

$$\sum_{l \in \delta_1(j)} y_{lm} \leq x_m, \ \forall m = (i_1, i_2) \in L, \ j \in V_1 \setminus \{i_1\} \qquad (21)$$

$$\sum_{l \in \delta_2(j)} y_{lm} \leq x_m, \ \forall m = (i_1, i_2) \in L, j \in V_2 \setminus \{i_2\} \qquad (22)$$

where $\delta_1(j)$ (resp., $\delta_2(j)$) denotes the set of lines in L incident with node $j \in V_1$ (resp., $j \in V_2$). This LP relaxation is the one actually solved in [20]. Let U_1 denote the corresponding upper bound value. The other relaxation, which is only mentioned in [20] and whose solution seems at present completely out of reach even for very small size instances, involves the y variables only and is defined by (7) subject to (10), the nonnegativity conditions, conditions $y_{lm} = 0$ for l, m incompatible, and the constraints

$$\sum_{(l,m) \in P} y_{lm} \leq 1, \ \forall P \in \mathcal{P}, \qquad (23)$$

where $\mathcal{P}$ denotes the (exponentially large) collection of all maximal sets $P \subset L \times L$ such that, for all (l, m), (p, q) $\in P$, at least two lines among l, m, p, q are incompatible (i.e., for every feasible solution, either $y_{lm} = 0$ or $y_{pq} = 0$). This latter formulation is an Independent Set formulation of our problem, and constraints (23) are the corresponding *Clique* inequalities (see [20]). We let U_2 denote the upper bound corresponding to the value of this latter LP.

PROPOSITION 2. *$U_1 \geq U(\lambda^*) \geq U_2$ and both* inequalities can *be tight.*

PROOF. It is easy to observe that the optimal solution of the relaxed problem (19), (8), (9), and (11) does not change if the integrality constraints are removed. In this case, it is known [10] that $U(\lambda^*)$ coincides with the optimal solution value of the LP relaxation defined by (7)–(9) plus the nonnegativity conditions. This bound is stronger than U_1 since (21) and (22) arc a special case of (9) (both $\delta_1(j)$ and $\delta_2(j)$ belong to $\mathcal{I}$). This shows $U_1 \geq U(\lambda^*)$.

To show that $U(\lambda^*) \geq U_2$, we prove that every feasible solution of the LP relaxation yielding U_2 is also feasible for the LP relaxation yielding $U(\lambda^*)$. Indeed, consider a nonnegative $\overline{y}$ satisfying (23), and define

$$\overline{x}_m := \max_{\substack{I \in \mathcal{I}: \\ m \notin I}} \sum_{l \in I} \overline{y}_{lm} \ (m \in L).$$

Clearly, all constraints (9) are satisfied by $(\overline{x}, \overline{y})$. What remains to be shown is that all constraints (8) are satisfied by $\overline{x}$. For a generic $I \in \mathcal{I}$, we have

$$\sum_{l \in I} \overline{x}_l = \sum_{l \in I} \max_{\substack{Q \in \mathcal{I}: \\ l \notin Q}} \sum_{q \in Q} \overline{y}_{ql}.$$

To see that this cannot be more than 1, consider for each $1 \in L$ an arbitrary clique $Q(Z) \in \mathcal{I}$ with $l \notin Q(1)$. It is easy to check that the set defined by $\{(q, l) : l \in I, q \in Q(l)\}$ belongs to $\mathcal{P}$. Since $\overline{y}$ satisfies (23), the claim follows.

The proof is concluded by showing examples in which the two inequalities in the statement are tight. This is deferred to the full paper. □

Our approach determines a near-optimal λ by a standard *subgradient* optimization procedure; see Held and Karp [16]. Subgradient optimization has the advantage of being easy to implement and has proved effective in many other applications. The procedure generates a series $\lambda^0, \lambda^1, \lambda^2, \ldots$ of multipliers, where $\lambda^0 := 0$ (with $b_{lm} = b_{ml} = a_{lm}/2$ for $l, m \in L$), and, for $k \geq 0$, λ^{k+1} is defined from λ^k as follows. Let $(\overline{x}, \overline{y})$ denote an optimal solution of the Lagrangian relaxed problem associated with λ^k. The corresponding *subgradient vector* is given by

$$s_{lm} := \overline{y}_{lm} - \overline{y}_{ml}, \ l, m \in L, l < m.$$

Using the technique proposed in [16], WC compute the new multipliers by

$$\lambda_{lm}^{k+1} := \begin{cases} \lambda_{lm}^k & \text{if } s_{lm} = 0 \\ \max(\lambda_{lm}^k - \gamma, -b_{lm}) & \text{if } s_{lm} = 1 \\ \min(\lambda_{lm}^k + \gamma, b_{ml}) & \text{if } s_{lm} = -1 \end{cases}$$

for $l.m \in L$, $l < m$. Here, the step size γ is defined by

$$\gamma = \mu \frac{UB - LB}{\sum_{l,m} s_{l,m}^2}$$

where μ is a suitable parameter, while UB and LB are the values of the best upper bound and feasible solution found so far, respectively. In our implementation, μ is initially set to 1, and halved if the upper bound does not decrease within 50 iterations (halving μ is customary within subgradient optimization). The number of iterations is limited by $\max\{1000, 10 \cdot \max\{|E_1|, |E_2|\}\}$, since we experimentally observed that afterwards no substantial improvement occurs. The overall complexity of the upper bound computation is therefore $O(|E_1||E_2| \max\{|E_1|, |E_2|\})$.

The heuristic procedure that we use to compute feasible solutions to the problem is very simple: Wc simply take the $\overline{x}$ vector corresponding to the Lagrangian relaxed solution, found at each iteration. The associated value is of course given by (1).

3. COMPUTATIONAL EXPERIMENTS

The algorithm has been implemented in C and run on a Pentium PC. Initial heuristic solutions were computed by using the heuristic algorithms mentioned in [20], and in particular the Genetic Algorithms which were shown effective for this problem. To test our algorithm, we used contact maps of real proteins from PDB, with a threshold of 5.4° for each contact. The number of available proteins is very large (about

Gap	# inst.	avg. n	avg. m	max n	max m
0	422	56.90	98.42	68	137
1	55	55.69	80.28	63	128
2	93	56.61	82.14	66	137
3	151	56.18	80.09	67	130
4	128	56.12	79.95	67	130
5	208	56.10	79.80	68	132
6	274	56.50	81.11	68	137
7	345	56.11	80.77	68	135
8	335	55.98	82.41	68	135
9	401	55.98	85.13	68	137
10+	7588	57.45	98.52	68	137

Table 1: Results for 10,000 pairs of small proteins

15,000), hence we had to select a subset of all possible pairs for our experiments.

First of all, we considered the 269 proteins mentioned in [20] and compared, in a week-end on a PC, the first 10,000 (in alphabetical order) of the (about) 36,000 corresponding pairs. Computing the upper bounds for all these instances with the method of [20] would have taken more than a year, assuming an average time of 1 hour/problem. For all instances for which the upper bounds of [20] were available, the upper bounds computed with our method turned out to be at least as good (actually in most cases the two bounds coincided). Note that we are not finding the best Lagrangian multipliers and hence, in principle, our upper bound may be worse than U_1. Table 1 subdivides the 10,000 instances according to the final value of the gap (column Gap) between the upper bound and the value of the best solution found. This means that 422 instances were solved to optimality. Columns avg. n and avg. m (resp. max n and max m) give the average (resp. maximum) number of residues and contacts in the instances associated with each gap value. The fact that for 3/4 of the instances the gap is 10 or larger reflects the present difficulty to solve instances associated with substantially different proteins, even of relatively small size. On the other hand, our code can find optimal maps for pairs of similar proteins, even of very large size, within few seconds. For instance, within less than one minute we could optimally align 1hkbA to 1hkcA, that have 891 (resp. 887) contacts and 1944 (resp. 1973) residues, and lqba to lqbb, that have 849 (resp. 848) contacts and 1925 residues.

Finally, in a few hours we compared all 780 pairs in a testbed of 40 large proteins suggested by Jeffery Skolnick and mentioned in [20], within few hours. These proteins have up to 250 residues and 593 contacts. In 150 cases, our method found the optimal solution. Table 2 reports the list of the pairs for which the optimal solution was found (marked by •) – the 40 proteins are classified into 4 families and proteins of the same family appear consecutively in the rows and columns of the table.

4. REFERENCES

[1] W.P. Adams and H.D. Sherali, A Tight Linearization and an Algorithm for Zero-One Quadratic Programming Problems, Management Science 32 (1986) 1274-1290.

[2] E. Balas, S. Ceria, and G. Cornuéjols, A Lift-and-Project Cutting Plane Algorithm for Mixed O-l Programs, *Mathematical* Programming 58 (1993) 295-324.

[3] T. L. Blundell, Structure-Based Drug Design, Nature 384 (1996) 23-26.

[4] C. Brauden and J. Toozc, *Introduction to Protein Structure,* Garland, 1999.

[5] A. Caprara, M. Fischetti and P. Toth, A Heuristic Method for the Set Covering Problem, *Operations Research 47* (1999) 730–743.

[6] A. Caprara, D. Pisingcr and P. Toth, Exact Solution of the Quadratic Knapsack Problem, *INFORMS J.* on Comput., 11 (1999) 125–137.

[7] I'. Carrarcsi and F. Malucelli, A Reformulation Scheme and New Lower Bounds for the QAP, in P.M. Pardalos and H. Wolkowicz (eds.), *Quadratic* Assignment *and Related Problems,* DIMACS series in Discrete Mathematics and Theoretical Computer Science, AMS Press, 147-160, 1994.

[8] I. Eidhammer and I. Jonassen and W. R. Taylor, Structure Comparison and Structure Prediction, to appear *J.* Comp. *Biol.*.

[9] D. Fischer et al.. CAFASP-1: Critical Assessment of Fully Automated Structure Prediction Methods. Proteins Suppl. 3 (1999) 209-217.

[10] M.L. Fisher, The Lagrangian Relaxation Method for Solving Integer Programming Problems, *Management* Science 27 (1981) l-18.

[11] A. Godzik and J. Skolnick, Flexible Algorithm for Direct Multiple Alignment of Protein Structures and Sequences, *CABIOS* 10 (1994) 587-596.

[12] A. Godzik, J. Sklonick and A. Kolinski, A Topology Fingerprint Approach to Inverse Protein Folding Problem, *J. Mol. Biol.227* (1992) 227-238.

[13] D. Goldman, S. Istrail and C. Papadimitriou, Algorithmic Aspects of Protein Structure Similarity, Proceedings *of the 40th IEEE* Symposium on Foundations *of* Computer Science, 512-522, 1999.

[14] D. Gusfield, *Algorithms* on Strings, *Trees* and Sequences: Computer Science and Computational Biology, Cambridge University Press, 1997.

[15] T. F. Havel, I. D. Kuntz and G. M. Crippen, Effect of Distance Constraints on Macromolecular Conformation, *Biopolymers* 18 (1979).

[16] M. Held and R. M. Karp, The Traveling Salesman Problem and Minimum Spanning Trees: Part II, *Mathematical* Programming 1 (1971) 6-25.

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

[18] L. Holm and C. Sander, Protein Structure Comparison by Alignment of Distance Matrices, *J. Mol. Biol. 233* (1993) 123-138.

[19] W. Kabash, A Solution for the Best Rotation to Relate Two Sets of Vectors, *Acta* Cryst. A32 (1978) 922-923.

[20] G. Lancia, R. Carr, B. Walenz and S. Istrail, 101 Optimal PDB Structure Alignments: a Branch-and-Cut Algorithm For The Maximum Contact Map Overlap Problem, *Proc. 5th RECOMB* (2001) 193-201.

[21] C. Lemmen and T. Lengauer, Computational Methods for the Structural Alignment of Molecules, Journal of Computer-Aided Molecular Design 14 (2000) 215-232.

[22] H. P. Lenhof, K. Reinert, M. Vingron, A Polyhedral Approach to RNA Sequence Structure Alignment, *J.* Comp. *Biol., 5 (1998)* 517-530.

[23] L. Lovász and A. Schrijver, Cones of Matrices and Set-Functions and O-l Optimization, *SIAM* Journal on Optimization 1 (1991) 166–190.

[24] L. Mirny and E. Domany, Protein Fold Recognition and Dynamics in the Space of Contact Maps, Proteins 26 (1996) 391-410.

[25] K. Mizuguchi, C. M. Deane, T. L. Blundell, and J. P. Overington, HOMSTRAD: a Database of Protein Structure Alignments for Homologous Families, *Protein Sci.* 7(11) (1998) 2469-2471.

[26] J. Moult, T. Hubbard, K. Fidelis, and J. Pedersen, Critical Assessment of Methods of Protein Structure Prediction (CASP): Round III, Proteins Suppl. 3 (1999) 2-6.

[27] A. G. Murzin, S. E. Brenncr, T. Hubbard and C. Chothia, SCOP: a Structural Classification of Proteins Database for the Investigation of Sequences and Structures, *J. Mol. Biol. 247* (1995) 536-540.

[28] G. L. Nemhauser and L. Wolsey, Integer and Combinatorial *Optimization,* John Wiley and Sons, 1988.

[29] N. Siew, A. Elofsson, L. Rychlewski and D. Fischer, MaxSub: An Automated Measure for the Assessment of Protein Structure Prediction Quality, Bioinformatics 16 (2000) 776-785.

[30] R. Sowdhamini, D. F. Burke, J. F. Huang, K. Mizuguchi, H. A. Nagarajaram, N. Srinivasan, R. E. Steward and T. L. Blundell, CAMPASS: a Database of Structurally Aligned Protein Superfamilies, **Structure** 6(9) **(1998)** 1087–1094.

[31] M. Vendruscolo, E. Kussell and E. Domany, Recovery of Protein Structure from Contact Maps, *Fold. Des. 2* (1997) 295-306.

5. ACKNOWLEDGMENTS

We are very grateful to Brian Walenz for providing us with the contact maps for all PDB proteins and with his implementation of the Genetic Algorithms for this problem. We also thank Bob Carr, Sorin Istrail and Brian Walenz for many useful discussions.

(i,j)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	-	•	•	•	•	•	•	•				•	•	•																										
2	•	-	•		•	•	•	•				•	•	•																										
3	•	•	-	•	•	•	•	•				•	•	•																										
4	•		•	-								•	•	•																										
5	•	•	•		-	•	•	•				•	•	•																										
6	•	•	•		•	-	•	•				•	•	•																										
7	•	•	•		•	•	-	•				•	•	•																										
8	•	•	•		•	•	•	-				•	•	•																										
9									-	•	•																													
10									•	-	•																													
11									•	•	-																													
12	•	•	•	•	•	•	•	•				-	•	•																										
13	•	•	•	•	•	•	•	•				•	-	•																										
14	•	•	•	•	•	•	•	•				•	•	-																										
15															-	•	•	•	•	•	•	•	•																	
16															•	-	•	•	•	•	•	•	•																	
17															•	•	-	•	•	•	•	•	•																	
18															•	•	•	-	•	•	•	•	•																	
19															•	•	•	•	-	•	•	•	•																	
20															•	•	•	•	•	-	•	•	•																	
21															•	•	•	•	•	•	-	•	•																	
22															•	•	•	•	•	•	•	-	•																	
23															•	•	•	•	•	•	•	•	-																	
24																								-	•	•	•	•	•	•	•	•	•	•						
25																								•	-	•	•	•	•	•	•	•	•	•						
26																								•	•	-	•	•	•	•	•	•	•	•						
27																								•	•	•	-	•	•	•	•	•	•	•						
28																								•	•	•	•	-	•	•	•	•	•	•						
29																								•	•	•	•	•	-	•	•	•	•	•						
30																								•	•	•	•	•	•	-	•	•	•	•						
31																								•	•	•	•	•	•	•	-	•	•	•						
32																								•	•	•	•	•	•	•	•	-	•	•						
33																								•	•	•	•	•	•	•	•	•	-	•						
34																								•	•	•	•	•	•	•	•	•	•	-						
35																																			-					
36																																				-	•	•		•
37																																				•	-			
38																																				•		-	•	•
39																																						•	-	•
40																																				•		•	•	-

1 1b00A
2 1dbwA
3 1nat
4 1ntr
5 1qmpA
6 1qmpB
7 1qmpC
8 1qmpD
9 1rn1A
10 1rn1B
11 1rn1C
12 4tmyA
13 4tmyB
14 3chy
15 1bawA
16 1byoA
17 1byoB
18 1kdi
19 1nin
20 1pla
21 2b3iA
22 2pcy
23 2plt
24 1amk
25 1aw2A
26 1b9bA
27 1btmA
28 1htiA
29 1tmhA
30 1treA
31 1tri
32 3ypiA
33 8timA
34 1ydvA
35 1b71A
36 1bcfA
37 1dpsA
38 1fha
39 1ier
40 1rcd

Table 2: Pairs of proteins of the Skolnick data set optimally aligned