An Efficient Lagrangian Relaxation for the Contact Map Overlap Problem

Rumen Andonov*[1], Nicola Yanev[2], and Noël Malod-Dognin[1]

[1] INRIA Rennes - Bretagne Atlantique and University of Rennes 1,
[2] University of Sofia, Bulgaria
randonov@irisa.fr, choby@math.bas.bg, nmaloddg@irisa.fr

Abstract. Among the measures for quantifying the similarity between protein 3-D structures, contact map overlap (CMO) maximization deserved sustained attention during past decade. Despite this large involvement, the known algorithms possess a modest performance and are not applicable for large scale comparison.
This paper offers a clear advance in this respect. We present a new integer programming model for CMO and propose an exact B&B algorithm with bounds obtained by a novel Lagrangian relaxation. The efficiency of the approach is demonstrated on a popular small benchmark (Skolnick set, 40 domains). On this set our algorithm significantly outperforms the best existing exact algorithms. Many hard CMO instances have been solved for the first time. To assess furthermore our approach, we constructed a large scale set of 300 protein domains. Computing the similarity measure for any of the 44850 couples, we obtained a classification in excellent agreement with SCOP.

Key words: Protein structure alignment, contact map overlap, combinatorial optimization, integer programming, branch and bound, Lagrangian relaxation.

1 Introduction

A fruitful assumption in molecular biology is that proteins sharing close three-dimensional (3D) structures are likely to share a common function and in most cases derive from a same ancestor. Computing the similarity between two protein structures is therefore a crucial task and has been extensively investigated [1–4]. Interested reader can also refer to [6–10]. We study here the *contact-map-overlap* (CMO) maximization, a scoring scheme first proposed in [5]. This measure is robust, takes partial matching into account, is translation-invariant and it captures the intuitive notion of similarity very well. Formally, a contact map is a $0-1$ symmetric matrix C where $c_{ij} = 1$ if the Euclidean distance between the alpha carbons (C_α) of the i-th and the j-th amino acid of a protein is smaller than a given threshold in the protein native fold. In the CMO approach one tries

* Corresponding author.

to evaluate the similarity between two proteins by determining the maximum overlap (also called alignment) of their contact maps.

The counterpart of the CMO problem in the graph theory is the maximum common subgraph problem [11], which is APX-hard [12]. CMO is also known to be NP-hard [5]. Designing efficient algorithms that guarantee the CMO quality is an important problem that has eluded researchers so far. The most promising approach seems to be integer programming coupled with either Lagrangian relaxation [2] or B&B reduction technique [13, 14]. This paper confirms once more the superiority of Lagrangian relaxation approach to solve CMO problems. Moreover, the Lagrangian relaxation has been successfully applied to RNA structures alignments [15].

The contributions of this paper are as follows. We propose a new mixed integer programming (MIP) formulation for the CMO problem. We present an efficient Lagrangian relaxation to solve our model and incorporate it into a B&B search. We also developed a second version of our algorithm which performs in agreement with the type of the secondary structure elements (SSE). To the best of our knowledge, such incorporation of biological informations in the CMO optimization search is done for the first time. We compare our approach to the best exact algorithms that exist [2, 14] on a widely used benchmark (the Skolnick set), and we notice that it outperforms them significantly, both in time and in quality of the provided bounds. New hard Skolnick set instances have been solved. Finally, our method was used as a classifier on both the Skolnick set and the Proteus_300 set (a large benchmark of 300 domains that we extracted from SCOP [19]). Again, we are not aware of any previous attempt to apply a CMO approach on such large database. The obtained results are in perfect agreement with SCOP classification and clearly demonstrate that our algorithm can be used as a tool for large scale classification.

2 The mathematical model

Our interest in CMO was provoked by its resemblance with the protein threading problem (PTP) for which we have presented an approach based on the non-crossing matching in bipartite graphs [16]. It yielded a highly efficient algorithm by using the Lagrangian duality [17, 18]. We aim to extend this approach in the case of CMO by presenting it as a matching problem in a bipartite graph, which in turn will be posed as a maximum weight augmented path in a structured graph.

Let us first introduce a few notations as follows. The contact maps of two proteins P1 and P2 are given by graphs $G_m = (V_m, E_m)$ for $m = 1, 2$, with $V_m = \{1, 2, \ldots, n_m\}$. The vertices V_m are better seen as ordered points on a line and correspond to the residues of the proteins. The sets of edges E_m correspond to the contacts. The right and left neighbours of node i are elements of the sets $\delta_m^+(i) = \{j | j > i, (i, j) \in E_m\}$ and $\delta_m^-(i) = \{j | j < i, (j, i) \in E_m\}$. Let $i \in V_1$ be matched with $k \in V_2$ and $j \in V_1$ be matched with $l \in V_2$. We will call a matching *non-crossing*, if $i < j$ implies $k < l$. Feasible alignments of two proteins P_1 and

P_2 are given by non-crossing matchings in the complete bipartite graph B with a vertex set $V_1 \cup V_2$.

Let the weight w_{ikjl} of the matching couple $(i,k)(j,l)$ be set as follows

$$w_{ikjl} = \begin{cases} 1 & \text{if } (i,j) \in E_1 \text{ and } (k,l) \in E_2 \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

For a given non-crossing matching M in B we define its weight $w(M)$ as the sum of weights over all couples of edges in M. CMO consists then in maximizing $w(M)$, where M belongs to the set of all non-crossing matchings in B.

In [16–18] we have already dealt with similar non-crossing matchings (in fact in the PTP they are many-to-one) and we have proposed for them a network flow presentation. This approach is adapted to CMO as follows (see for illustration Fig. 1). The edges of the bipartite graph B are mapped to the points of a $n_1 \times n_2$ rectangular grid $B' = (V', E')$ according to the rule: a point $(i,k) \in V'$ corresponds to the edge (i,k) in B and vice versa.

Definition 1 *A **feasible path** is an arbitrary sequence of points in B' $(i_1,k_1),(i_2,k_2),...,(i_t,k_t)$ such that $i_j < i_{j+1}$ and $k_j < k_{j+1}$ for $j \in [1,t-1]$.*

The correspondence between a feasible path and a non-crossing matching is then obvious. Searching for feasible alignments of two proteins is in this way converted into searching for strictly increasing node sets in B'. We also add arcs $(i,k) \rightarrow (j,l) \in E'$ iff $w_{ikjl} = 1$. In B', solving CMO, corresponds to finding the densest (in terms of arcs) subgraph of B' whose node set is a feasible path.

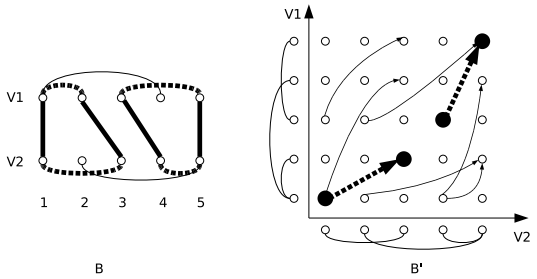

Fig. 1. Left: Vertex 1 from V1 is matched with vertex 1 from V2 and 2 is matched with 3: matching couple $(1,1)(2,3)$. Other matching couples are $(3,4)(5,5)$. This defines a feasible matching $M = \{(1,1)(2,3),(3,4)(5,5)\}$ with weight $w(M) = 2$. Right: The same matching is visualized in the graph B'.

To each node $(i,k) \in V'$ we associate now a 0/1 variable x_{ik}, and to each arc $(i,k) \rightarrow (j,l) \in E'$, a 0/1 variable y_{ikjl}. Denote by X the set of feasible paths. The problem can now be stated as follows :

$$v(CMO) = \max \sum_{(ik)(jl) \in E'} y_{ikjl} \quad (2)$$

subject to

$$x_{ik} \geq \sum_{l \in \delta_2^+(k)} y_{ikjl}, \qquad j \in \delta_1^+(i), \quad i \in [1, n_1 - 1], \quad k \in [1, n_2 - 1]. \tag{3}$$

$$x_{ik} \geq \sum_{l \in \delta_2^-(k)} y_{jlik}, \qquad j \in \delta_1^-(i), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{4}$$

$$x_{ik} \geq \sum_{j \in \delta_1^+(i)} y_{ikjl}, \qquad l \in \delta_2^+(k), \quad i \in [1, n_1 - 1], \quad k \in [1, n_2 - 1]. \tag{5}$$

$$x_{ik} \geq \sum_{j \in \delta_1^-(i)} y_{jlik}, \qquad l \in \delta_2^-(k), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{6}$$

$$x \in X \tag{7}$$

Actually, we know how to represent X with linear constraints. It is easily seen that definition 1 of a feasible path yields the following inequalities

$$\sum_{l=1}^{k} x_{il} + \sum_{j=1}^{i-1} x_{jk} \leq 1, \qquad i \in [1, n_1], \quad k \in [1, n_2]. \tag{8}$$

The same definition also implies that the j-th residue from $P1$ could be matched with at most one residue from $P2$ and vice-versa. This explains the sums on the right hand sides of (3) and (5) (for arcs having their tails at vertex (i, k)); and (4) and (6) (for arcs heading to (i, k)). Any $(i, k)(j, l)$ arc can be activated (i.e. $y_{ikjl} = 1$) iff $x_{ik} = 1$ and $x_{jl} = 1$ and in this case the respective constraints are active because of the objective function.

A tighter description of the polytope defined by (3)–(6) and $0 \leq x_{ik} \leq 1$, $0 \leq y_{ikjl}$ could be obtained by lifting the constraints (4) and (6) as it is shown in Fig. 2. The points shown are just the predecessors of (i, k) in the graph B' and they form a grid of $\delta_1^-(i)$ rows and $\delta_2^-(k)$ columns. Let $i_1, i_2, \ldots, i_s$ be all the vertices in $\delta_1^-(i)$ ordered according to the numbering of the vertices in V_1 and likewise $k_1, k_2, \ldots, k_t$ in $\delta_2^-(k)$. Then the vertices in the l-th column $(i_1, k_l), (i_2, k_l), \ldots (i_s, k_l)$ correspond to pairwise crossing matchings and at most one of them could be chosen in any feasible solution $x \in X$ (see (6)). This "all crossing" property holds even if we add to this set the following two sets: $(i_s, k_1), (i_s, k_2), \ldots, (i_s, k_{l-1})$ and $(i_1, k_{l+1}), (i_1, k_{l+2}), \ldots, (i_1, k_t)$. Denote by $col_{ik}(l)$ the union of these three sets, and analogously, by $row_{ik}(j)$ the corresponding union for the j-th row of the grid. When the grid is one column/row, only the set $row_{ik}(j)/col_{ik}(l)$ is empty.

Now a tighter LP relaxation of (3)–(6) is obtained by substituting (4) with (9), and (6) with (10).

$$x_{ik} \geq \sum_{(r,s) \in row_{ik}(j)} y_{rsik}, \qquad j \in \delta_1^-(i), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{9}$$

$$x_{ik} \geq \sum_{(r,s)\in col_{ik}(l)} y_{rsik}, \qquad l \in \delta_2^-(k), \quad i \in [2, n_1], \quad k \in [2, n_2]. \tag{10}$$

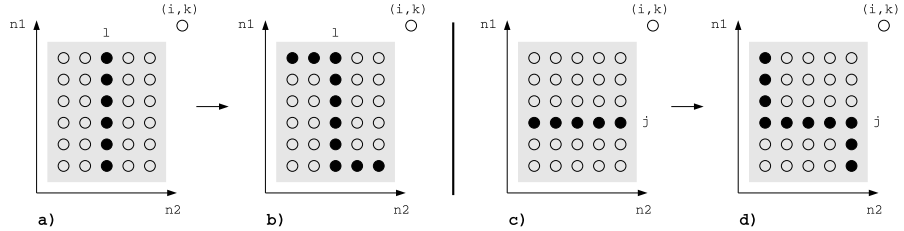

Fig. 2. The shadowed area represents the set of vertices in V' which are tails for the arcs heading to a given point (i, k). In a) : • correspond to the indices of y_{jlik} in (6) for l fixed. b) presents the same for the tightened constraint (10). In c) : • correspond to the y_{jlik} in (4) for j fixed. d) presents the same for the tightened constraint (9).

Remark: Since we are going to apply the Lagrangian technique there is no need neither for an explicit description of the set X neither for lifting the constraints (3) and (5).

3 Method

3.1 Lagrangian relaxation

Here, we show how the Lagrangian relaxation of constraints (9) and (10) leads to an efficiently solvable problem, yielding upper and lower bounds that are generally better than those found by the best known exact algorithm [2].

Let $\lambda^h_{ikj} \geq 0$ (respectively $\lambda^v_{ikj} \geq 0$) be a Lagrangian multiplier assigned to each constraint (9) (respectively (10)). By adding the slacks of these constraints to the objective function with weights λ, we obtain the Lagrangian relaxation of the CMO problem

$$\begin{aligned} LR(\lambda) = \max \sum_{(ik)(jl)\in E_{B'}} y_{ikjl} \quad &+ \sum_{i,k,j\in\delta_1^-(i)} \lambda^h_{ikj}(x_{ik} - \sum_{(r,s)\in row_{ik}(j)} y_{rsik}) \\ &+ \sum_{i,k,l\in\delta_2^-(k)} \lambda^v_{ikl}(x_{ik} - \sum_{(r,s)\in col_{ik}(l)} y_{rsik}) \end{aligned} \tag{11}$$

subject to $x \in X$, (3), (5) and $y \geq 0$.

Proposition 1 *$LR(\lambda)$ can be solved in $O(|V'| + |E'|)$ time.*

Proof: For each $(i,k) \in V'$, if $x_{ik} = 1$ then the optimal choice y_{ikjl} amounts to solving the following : The heads of all arcs in E' outgoing from (i,k) form a $|\delta^+(i)| \times |\delta^+(k)|$ table. To each point (j,l) in this table, we assign the profit $\max\{0, c_{ikjl}(\lambda)\}$, where $c_{ikjl}(\lambda)$ is the coefficient of y_{ikjl} in (11). Each vertex in this table is a head of an arc outgoing from (i,k). Then the subproblem we need to solve consists in finding a subset of these arcs having a maximal sum $c_{ik}(\lambda)$ of profits(the arcs of negative weight are excluded as candidates for the optimal solution) and such that their heads lay on a feasible path. This could be done by a dynamic programming approach in $O(|\delta^+(i)||\delta^+(k)|)$ time. Once profits $c_{ik}(\lambda)$ have been computed for all (i,k) we can find the optimal solution to $LR(\lambda)$ by using the same DP algorithm but this time on the table of $n1 \times n2$ points with profits for (i,k)-th one given by: $c_{ik}(\lambda) + \sum_{j \in \delta_1^-(i)} \lambda^h_{ikj} + \sum_{l \in \delta_2^-(k)} \lambda^v_{ikl}$, where the last two terms are the coefficients of x_{ik} in (11). The inclusion $x \in X$ is explicitly incorporated in the DP algorithm.

3.2 Subgradient descend

In order to find the tightest upper bound on $v(CMO)$ (or eventually to solve the problem), we need to solve in the dual space of the Lagrangian multipliers $LD = \min_{\lambda \geq 0} LR(\lambda)$, whereas $LR(\lambda)$ is a problem in x, y. A number of methods have been proposed to solve Lagrangian duals: dual ascent, constraint generation, column generation, etc... Here, we choose the subgradient descent method, because of our large number of lagrangian multipliers. It is an iterative method in which at iteration t, given the current multiplier vector λ^t, a step is taken along a subgradient of $LR(\lambda)$; then, if necessary, the resulting point is projected onto the nonnegative orthant. It is well known that practical convergence of the subgradient method is unpredictable. For some "good" problems, convergence is quick and fairly reliable, while other problems tend to produce erratic behavior. The computational runs on a rich set of real-life instances confirm that our approach belong to the "good" cases.

In our realization, the update scheme for λ_{ikj} (and analogously for λ_{ikl}) is $\lambda^{t+1}_{ikj} = \max\{0, \lambda^t_{ikj} - \Theta^t g^t_{ikj}\}$, where $g^t_{ikj} = \bar{x}_{ik} - \sum \bar{y}_{jlik}$ (see (9) and (10) for the sum definition) is the sub-gradient component $(0, 1,$or $-1)$, calculated on the optimal solution $\bar{x}$, $\bar{y}$ of $LR(\lambda^t)$. The step size Θ^t is $\Theta^t = \frac{\alpha(LR(\lambda^t) - Z_{lb})}{\sum (g^t_{ikj})^2 + \sum (g^t_{ikl})^2}$ where Z_{lb} is a known lower bound for the CMO problem and α is an input parameter. Into this approach the x-components of $LR(\lambda^t)$ solution provides a feasible solution to CMO and thus a lower bound also. If $LD \leq v(CMO)$ then the problem is solved. If $LD > v(CMO)$ holds, in order to obtain the optimal solution, one could pass to a B&B algorithm suitably tailored for such an upper and lower bounds generator.

3.3 Branch and Bound

From among various possible nodes splitting rules, the one shown in Fig. 3 gives good results (see section 4). Formally, a node of B&B is given by n_2

couples (b_k, t_k) for $k \in [1, n_2]$ representing the zone to be explored (the white area on Fig. 3). A vertex (j, l) of the graph B' belongs to this area if $b_l \leq j \leq t_l$. Let (r_b, c_b) be the $\text{argmax}_{(i,k)\in V'}[\min(D(i,k), U(i,k))]$, where $D(i,k) = \sum_{l \geq k} \max(i - b_l, 0)$ and $U(i,k) = \sum_{l \leq k} \max(t_l - i, 0)$. Now, the two descendants of the current node are obtained by discarding from its feasible set the vertices in $U(r_b, c_b)$ and $D(r_b, c_b)$ respectively. The goal of this strategy is twofold: to create descendants that are balanced in sense of feasible set size and to reduce maximally the parent node's feasible set.

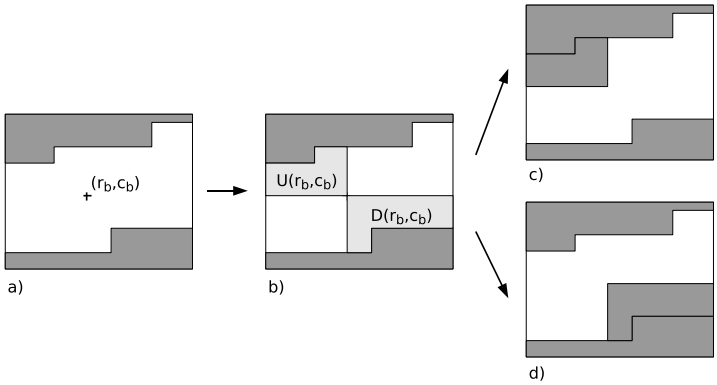

Fig. 3. Sketch of the B&B splitting strategy. a) The white area represents the current node feasible set; b) Fixing the point (r_b, c_b) creates the regions, $D(r_b, c_b)$ and $U(r_b, c_b)$; c) and d) are the descendants of the node a).

Finally, the main steps of the B&B algorithm are as follows:
Initialization: Set L={root} (root is the original CMO problem, i.e. with no restrictions on the feasible paths).
Problem selection and relaxation: Select and delete the problem P from L having the biggest upper bound. Solve the Lagrangian dual of P.
Fathoming and Pruning: Follow classical rules.
Partitioning : Create and add to L the two descendants of P
Termination : if $L = \emptyset$, the solution (x^*, y^*) is optimal.

3.4 Adding biological informations

In it's native state, a protein contains secondary structure elements (SSE), which are highly regular substructures. There are two SSE types: α helix (H) and β strand (b). Residues which are not part of a SSE are said to be in a coil (c).

As defined in (2)-(7), CMO does not consider the SSE type. Potentially, this can conduct to biologically not acceptable matching, such as aligning α with β SSE. To avoid that, we enrich the feasible path definition with the SSE information. A node (i, k) will be not acceptable, if residue i is of type H, while residue k is of type b, or vice versa. The SSE knowledge is already used in non-CMO methods like VAST[1], but has been never used before in a CMO approach.

The impact of such a filter on the CMO behavior is twofold: from one hand it directly leads to a biologically correct alignment, from the other hand, it makes the search space sparser, and accelerates about 2.5 times the solution process.

4 Numerical results

The results presented here were obtained on a computer with AMD Opteron (TM) CPU at 2.4 GHz, 4 Gb Ram. The algorithm was implemented in C++. To generate contact maps we consider two residues to be in contact if their C_α are within 7.5 Å, without taking into account contacts between consecutive residues. When needed, the SSE information was computed using the publicly available software Kaksi[3]. The first version of our algorithm will be denoted by `A_purva`[4], while the version with the SSE filter will be indicated as `A_purva_sse`.

To evaluate these algorithms we performed two kinds of experiments. In the first one (section 4.1), we compared our approach - in terms of performance and quality of the bounds - with the best exact algorithms from the literature [2, 14]. The former one is based on Lagrangian relaxation, and will be denoted here by `LAGR`. Our approach differs from it in two main points: i) in the proposed MIP formulation and ii) in the set of dualized constraints. This can explain the significant differences in the computational behavior of `A_purva` versus `LAGR`. The second algorithm, denoted here by `CMOS`, has been recently described in [14]. The comparison was done on a set of protein domains suggested by J. Skolnick. It contains 40 medium size domains from 33 proteins. The number of residues varies from 97 (2b3iA) to 256 (1aw2A), and the number of contacts varies from 320 (1rn1A) to 936 (1btmA). According to SCOP classification [19], the Skolnick set contains five families (see Table 1).

SCOP Fold	SCOP Family	Proteins
Flavodoxin-like	CheY-related	1b00, 1dbw, 1nat, 1ntr, 3chy 1qmp(A,B,C,D), 4tmy(A,B)
Cupredoxin-like	Plastocyanin /azurin-like	1baw, 1byo(A,B), 1kdi, 1nin 1pla, 2b3i, 2pcy, 2plt
TIM beta/alpha-barrel	Triosephosphate isomerase (TIM)	1amk, 1aw2, 1b9b, 1btm, 1hti 1tmh, 1tre, 1tri, 1ydv, 3ypi, 8tim
Ferritin-like	Ferritin	1b71, 1bcf, 1dps, 1fha, 1ier, 1rcd
Microbial ribonuclease	Fungal ribonucleases	1rn1(A,B,C)

Table 1. The five families in the Skolnick set.

Afterwards (section 4.2), we experimentally evaluated the capability of our algorithm to perform as a classifier on two sets: Skolnick and Proteus_300. The latter benchmark was proposed by us. It contains more, and significantly larger

[3] http://migale.jouy.inra.fr/outils/mig/kaksi/

[4] Apurva (Sanskrit) = not having existed before, unknown, wonderful, ...

proteins: 300 domains, with number of residues varying from 64 (d15bbA_) to 455 (d1po5A_). The maximum number of contacts is 1761 (d1i24A_). These domains are classified by SCOP in 30 families. All our data and results[5] are available on the URL: *http://www.irisa.fr/symbiose/softwares/resources/proteus300*.

4.1 Performance and quality of bounds

The Skolnick set requires aligning 780 pairs of domains. `A_purva` and `LAGR` (whose code was kindly provided to us by G. Lancia) were executed on the same computer and with the same 7.5Å contact maps. For both algorithms, the computation time was bounded to 1800 sec/instance. Table 2 shows the number of instances solved by each algorithm. `A_purva` succeeded to solve 502 couples, while `LAGR` solved only 161 couples. All these 161 instances are "easy", i.e. they align domains from the same SCOP family. In table 2, we also give the number of solved instances by `LAGR` and `CMOS` taken from [14]. These results where obtained on a similar workstation, with the same time limit (this information was kindly provided to us by N. Sahinidis), but with different contact maps (the threshold used was 7Å). `CMOS` solves 161 easy instances. Note that `A_purva` is the only one to solve 338 "hard" instances, i.e. couples with domains from different families. On the other hand, `A_purva` was outperformed by its SSE version, which demonstrated the usefulness of integrating this filter.

	Our contact maps (7.5Å)			CMOS contact maps (7Å)	
	LAGR	A_purva	A_purva_sse	LAGR	CMOS
Easy instances (164)	161	164	164	150	161
Hard instances (616)	0	338	444	0	0
Total (780)	161	502	608	150	161

Table 2. Number of instances solved by the different CMO methods, with a time limit of 1800 sec/instance. A_purva is the only one able to solve all easy instances, as well as many of the hard instances. The advantage of adding a SSE filter is noticeable.

Figure 4 compares the time needed by `LAGR` to that of `A_purva` on the set of instances solved by both algorithms. We observe that `A_purva` is significantly faster than `LAGR` (up to several hundred times in the majority of cases). The use of SSE information can push this even further, since A_purva_sse is about 2.5 times faster than A_purva.

We observed that the time for solving instances (without using SSE information) that align domains from the same family varies between 0.04s and 4.27s (except for two instances); this time varies respectively from 17.9s to more than 1800s when aligning domains from different classes. In this manner our results confirmed once more the property (also observed in [2, 14]) that : instances, such that both domains belong to the same family, seem to be easily solvable; in contrast to instances that align domains from different families.

[5] contact maps, solved instances, upper and lower bounds, run time, classifications...

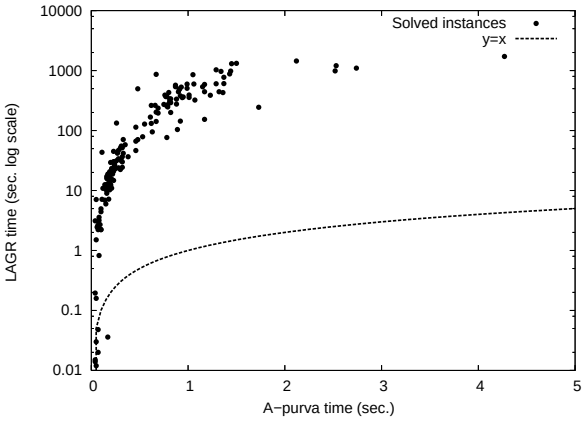

Fig. 4. A_purva versus LAGR running time comparison on the set of Skolnick instances solved by both algorithms. The A_purva time is presented on the x-axis, while the one of LAGR is on the y-axis. Often, A_purva is more than 100 times faster.

Our next observation concerns the quality of gaps obtained by `LAGR` and `A_purva` on the set of unsolved instances. Remember that when a Lagrangian algorithm stops because of time limit (1800 sec. in our case) it provides an upper bound (UB), and a lower bound (LB), which is a real advantage of a B&B type algorithm compared to any meta-heuristics. The relative gap value $(UB - LB)/UB$, measures how far is the optimization process from finding the exact optimum. Fig. 5 shows the relative gaps of `A_purva` plotted against those of `LAGR`. The entire figure is very asymmetric to the advantage of our algorithm since the relative gaps of `A_purva` are always smaller than those of `LAGR`, meaning that the bounds of `A_purva` are always tighter.

4.2 A_purva as a classifier

In this section we are interested in checking the ability of `A_purva_sse` to perform successfully as classifier in a given small lapse of time. We used the following protocol : we limited the runs of `A_purva_sse` to the root (i.e. B&B was not used), with a limit of 500 iterations for the subgradient descent. To measure the similarity between two proteins P_1 and P_2, we used the function defined in [14]: $Sim(P_1, P_2) = 2.LB/(|E_1| + |E_2|)$, where LB is the incumbent value found by `A_purva_sse`. These similarities were given to `Chavl` [20], a publicly available tool which proposes both a hierarchical ascendant classification and the cut corresponding to the best partition level (therefore, it does not require a similarity threshold). The obtained result was compared with the SCOP classification at a family level.

For the Skolnick set, the alignment of all couples was done in less than 810 seconds ($\simeq$ 1.04 sec/couple). The classification returned by `Chavl` was exactly the same as the classification at the family level in SCOP. To get a stronger

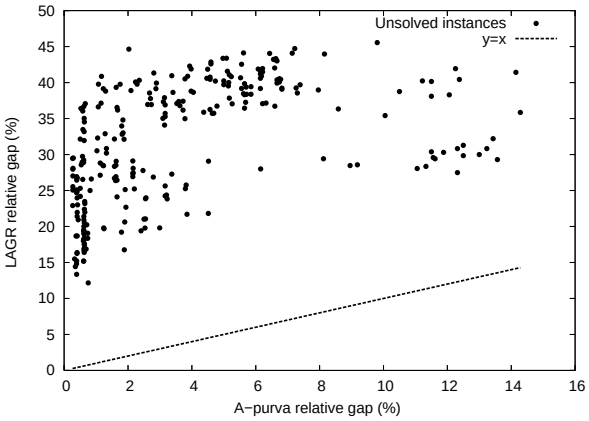

Fig. 5. Comparing relative gaps on the set of unsolved instances. The A_purva gaps (x-axis) are plotted against the LAGR gaps (y-axis). Relative gaps obtained by A_purva are substantially smaller.

confirmation of `A_purva` classifier capabilities, we performed the same operation on the Proteus_300 set. Aligning the 44850 couples required roughly 22 hours ($\simeq$ 1.82 sec/couple). The classification returned by `Chavl` contained 34 classes. The only difference between our classification and the one of SCOP at the family level is that 4 SCOP families were each split in two in our classification.

5 Conclusion

In this paper, we give an efficient exact algorithm for contact map overlap problem. The bounds are found by using Lagrangian relaxation, and the dual problem is solved by sub-gradient approach. The performance of the algorithm is demonstrated on a benchmark set of 40 domains and its superiority over the existing algorithms is obvious. We also propose a suitable filter based on secondary structures information which further accelerates the solution process. The capacity of the proposed algorithm to provide a convenient similarity measure was tested on a large data set of 300 protein domains. We were able to obtain in a short time a classification in very good agreement to the well known SCOP database.

6 Acknowledgment

Supported by ANR grant Calcul Intensif projet PROTEUS (ANR-06-CIS6-008) and by Hubert Curien French-Bulgarian partnership "RILA 2006" N^0 15071XF. N. Malod-Dognin is supported by Région Bretagne. We are thankful to Professor Giuseppe Lancia for numerous discussions and for kindly providing us with the source code of `LAGR`. All computations were done on the Ouest-genopole bioinformatics platform (http://genouest.org). Thanks to Jacques Nicolas and Basavanneppa Tallur for their suggestions.

References

1. J-F. Gibrat, T. Madej and S.H. Bryant. "Surprising similarities in structure comparison". *Curr. Opin. Struct. Biol*, 6, 377-385, 1996.
2. A. Caprara, R. Carr, S. Israil, G. Lancia and B. Walenz. 1001 Optimal PDB Structure Alignments: Integer Programming Methods for Finding the Maximum Contact Map Overlap. *J. Comput. Biol.*, 11(1), 27-52, 2004.
3. I. Halperin, B. Ma, H. Wolfson, et al. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins Struct. Funct. Genet.*, 47, 409-443, 2002.
4. A. Godzik. The Structural alignment between two proteins: is there a unique answer? *Protein Science*, 5, 1325-1338, 1996.
5. D. Goldman, S. Israil, C. Papadimitriu. Algorithmic aspects of protein structure similarity. *IEEE Symp. Found. Comput. Sci.* 512-522,1999.
6. A. Caprara, and G. Lancia. Structural Alignment of Large-Size Protein via Lagrangian Relaxation. *RECOMB 2002*, 100-108.
7. G. Lancia, and S. Istrail. Protein Structure Comparison: Algorithms and Applications. *Prot. Str. Anal. Des.*, 1-33, 2003.
8. R. Carr, and G. Lancia. Compact optimization can outperform separation: A case study in structural proteomics. *4OR*, 2: 221-233, 2004
9. P.K. Agarwal, N.H. Mustafa, and Y. Wang. Fast Molecular Shape Matching Using Contact Maps. *J. Comput. Biol.* , 14, 2, pp 131-147, 2007
10. J. Xu, F. Jiao, B. Berger. A parametrized Algorithm for Protein Structure Alignment. *RECOMB 2006*, LNCS 3909, 488-499.
11. M. Garey, D. Johnson. Computers and Intractability: A Guide to the Thoery of NP-completness. *Freeman and company*, New York, 1979
12. P. Crescenzi, V. Kann. A compendium of NP optimization problems. *http://www.nada.kth.se/∼viggo/problemlist/*
13. D.M. Strickland, E. Barnes, and J.S. Sokol. Optimal Protein Structure Alignment Using Maximum Cliques. *Oper. Res.*, 53, 389-402, 2005.
14. W. Xie, and N. Sahinidis. A Reduction-Based Exact Algorithm for the Contact Map Overlap Problem. *J. Comput. Biol.*, 14, 637-654, 2007.
15. M. Bauer, G.W. Klau, and K. Reinert. Fast and Accurate Structural RNA Alignment by Progressive Lagrangian Optimization. *CompLife 2005*, LNBI 3695, 217-228, 2005.
16. R. Andonov, S. Balev, and N. Yanev. Protein threading: From mathematical models to parallel implementations. *INFORMS J. on Comput.*, 16(4), 2004.
17. P. Veber, N. Yanev, R. Andonov, V. Poirriez. Optimal protein threading by cost-splitting. *WABI 2005*, LNCS, 3692, 365-375.
18. N. Yanev, P. Veber, R. Andonov and S. Balev. Lagrangian approaches for a class of matching problems. *Comp. and Math. with Appl.*, 55(5), 1054-1067, 2008.
19. A. Andreeva, D. Howorth, J-M. Chandonia, S.E. Brenner, T.J.P. Hubbard, C. Chothia, A.G. Murzin. Data growth and its impact on the SCOP database: new developments. *Nucl. Acid Res.* 2007.
20. I.C. Lerman. Likelihood linkage analysis (LLA) classification method (Around an example treated by hand). *Biochimie, Elsevier Editions* 75, 379-397, 1993.