Serdica J. Computing **4** (2010), 93–100

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

SOLVING MAXIMUM CLIQUE PROBLEM FOR PROTEIN STRUCTURE SIMILARITY*

Noël Malod-Dognin, Rumen Andonov, Nicola Yanev

Abstract. Computing the similarity between two protein structures is a crucial task in molecular biology, and has been extensively investigated. Many protein structure comparison methods can be modeled as maximum weighted clique problems in specific k-partite graphs, referred here as alignment graphs.

In this paper we present both a new integer programming formulation for solving such clique problems and a dedicated branch and bound algorithm for solving the maximum cardinality clique problem. Both approaches have been integrated in VAST, a software for aligning protein 3D structures largely used in the National Center for Biotechnology Information, an original clique solver which uses the well known Bron and Kerbosch algorithm (BK). Our computational results on real protein alignment instances show that our branch and bound algorithm is up to 116 times faster than BK.

1. Introduction. A fruitful assumption in molecular biology is that proteins of similar three-dimensional (3D) structures are likely to share a common function and in most cases derive from the same ancestor. Understanding

ACM Computing Classification System (1998): G.2.1, G.2.2.

Key words: protein structure comparison, maximum clique, k-partite graphs, integer programming, branch and bound.

*This work is supported by the ANR project PROTEUS "ANR-06-CIS6-008", by the Brittany Region and by the Bulgarian NSF project DO 02-359/2008.

and computing physical similarity of protein structures is one of the keys for developing protein based medical treatments, and thus it has been extensively investigated [11]. Evaluating the similarity of two protein structures can be done by finding an optimal order-preserving matching (also called alignment) between their components. We show that finding such alignments is equivalent to solving maximum clique problems in specific k-partite graphs referred here as alignment graphs. In this context, we present a new integer programming model for solving the maximum weighted clique problem in alignment graphs. In addition, we also propose a dedicated branch and bound algorithm (B&B) for the maximum clique problem. Both approaches have been integrated and validated in VAST[7] (Vector Alignment Search Tool), a software for aligning protein 3D structures largely used in the National Center for Biotechnology Information[1], and compared to the original VAST clique solver which is based on the Bron and Kerbosch algorithm (BK) [5]. The obtained results on real protein structure comparison instances show that our B&B algorithm is up to 116 times faster than BK, and thus clearly demonstrate the usefulness of our dedicated algorithm.

2. Clique problems and protein structure similarity. In this paper, we focus on grid-alike graphs, which we define as follows. A $m \times n$ **alignment graph** $G = (V, E)$ is a graph in which the vertex set V is depicted by a (m-row) $\times$ (n-column) array T, where each cell $T[i][k]$ contains at most one vertex $i.k$ from V (note that for both arrays and vertices, the first index stands for the row number, and the second for the column number). Two vertices $i.k$ and $j.l$ can be connected by an edge $(i.k, j.l) \in E$ only if $i < j$ and $k < l$. It is easily seen that the m rows form a m-partition of G, and that the n columns also form a n-partition. As for the general case, a clique in G is a subset of V such that any two vertices in it are connected by an edge.

Various clique problems can be formulated in such a graph. The *Maximum Clique* problem (**MCC**) consists in finding in G a clique of maximum cardinality, denoted by MCC(G). MCC is one of the first problems shown to be NP-Complete [8]. If we associate to each vertex $i.k$ a weigth S_{ik}, and to each edge $(i.k, j.l)$ a weight C_{ikjl}, then other maximum clique problems arise. The most general one is the *Maximum Weighted Clique* problem (**MWC**), which consists in finding the clique having the maximum sum of vertex and edge weights. Its particular cases – MCC, the clique with maximum sum of vertex weights and the clique with maximum sum of edge weights – have been extensively investigated [1, 4, 6].

From a general point of view, two proteins P_1 and P_2 can be represented

[1]`http://www.ncbi.nlm.nih.gov/Structure/VAST/vast.shtml`

by their ordered set of components N_1 and N_2, and estimating their similarity can be done by finding an optimal matching between the elements of N_1 and N_2. In [2], we show that such matchings can be represented in an $|N_1|\times|N_2|$ alignment graph $G = (V, E)$, where each row corresponds to an element of N_1 and each column corresponds to an element of N_2. A vertex $i.k$ is in V (i.e. matching $i \leftrightarrow k$ is possible), only if elements $i \in N_1$ and $k \in N_2$ are compatible, and this compatibility can be represented by a weight S_{ik}. An edge $(i.k, j.l)$ is in E if and only if (i) $i < j$ and $k < l$, for order preserving, and (ii) matching $i \leftrightarrow k$ is compatible with matching $j \leftrightarrow l$. Again, this compatibility can be represented by a weight C_{ikjl}. A feasible matching of P_1 and P_2 is then a clique in G. There is a multitude of alignment methods and they differ mainly by (i) the nature of the elements of N_1 and N_2, (ii) the compatibility definitions between elements and between pairs of matched elements, and (iii) the kind of maximum clique to find in G. For example, in VAST, N_1 and N_2 contain 3D vectors representing the secondary structure elements of P_1 and P_2. Matching $i \leftrightarrow k$ is possible if vectors i and k have similar norms and correspond either both to α-helices or both to β-strands. Finally, matching $i \leftrightarrow k$ is compatible with matching $j \leftrightarrow l$ only if the couple of vectors (i, j) from P_1 can be well superimposed in 3D-space with the couple of vectors (k, l) from P_2. The longest alignment corresponds to $MCC(G)$.

3. Integer programming model for MWC. By using the properties of our alignment graphs, we designed a new integer programming (IP) model (whose formulation is very different from [10, 3]) for solving the maximum weighted clique problem, where the weights are all in $\mathbb{R}$. To each vertex $i.k \in V$ (in row $i \in V_1$ and column $k \in V_2$), we associate a binary variable x_{ik} such that:

$$x_{ik} = \{1 \text{ if vertex } i.k \text{ is in the clique}, 0 \text{ otherwise}\}.$$

We also associate to each edge $(i.k, j.l) \in E$ a binary variable y_{ikjl} such that:

$$y_{ikjl} = \{1 \text{ if edge } (i.k, j.l) \text{ is in the clique}, 0 \text{ otherwise}\}.$$

The goal is to find a clique which maximizes the sum of its vertex weights and the sum of its edge weights. This leads to the objective function:

$$Z_{MWC} = \max \sum_{i.k} S_{ik}\ x_{ik} + \sum_{(i.k,j.l)} C_{ikjl}\ y_{ikjl}. \tag{1}$$

The one-to-one matching implies special order set constraints. In each row $i \in V_1$, at most one vertex can be chosen (2), and the same holds for the columns (3).

$$\sum_{k} x_{ik} \le 1, \quad \forall i \in V_1. \tag{2}$$

$$(3) \qquad \sum_i x_{ik} \leq 1, \quad \forall k \in V_2.$$

These special order set constraints lead to compact formulations of the relations between vertices and edges. Denote by $d^{+}_{col}(i.k)$ the set of columns l, $l > k$, such that $\exists(i.k, j.l) \in E$. In a similar way, $d^{-}_{col}(i.k)$ is the set of columns l, $l < k$, such that $\exists(j.l, i.k) \in E$. $d^{+}_{row}(i.k)$ is the set of rows j, $j > i$, such that $\exists(i.k, j.l) \in E$. And finally, $d^{-}_{row}(i.k)$ is the set of rows j, $j < i$, such that $\exists(j.l, i.k) \in E$. Edge-driven activations of vertices can be formulated with (4), (5), (6) and (7):

$$(4) \qquad x_{ik} \geq \sum_j y_{ikjl}, \quad \forall i.k \in V,\ \forall l \in d^{+}_{col}(i.k).$$

$$(5) \qquad x_{jl} \geq \sum_i y_{ikjl}, \quad \forall j.l \in V,\ \forall k \in d^{-}_{col}(j.l).$$

$$(6) \qquad x_{ik} \geq \sum_l y_{ikjl}, \quad \forall i.k \in V,\ \forall j \in d^{+}_{row}(i.k).$$

$$(7) \qquad x_{jl} \geq \sum_k y_{ikjl}, \quad \forall j.l \in V,\ \forall i \in d^{-}_{row}(j.l).$$

Vertice-driven activations of edges can be formulated with (8) and (9) :

$$(8) \qquad \sum_i x_{ik} + \sum_j x_{jl} - \sum_{ij} y_{ikjl} \leq 1,\ \forall k \in V_2,\ \forall l \in V_2,\ k < l.$$

$$(9) \qquad \sum_k x_{ik} + \sum_l x_{jl} - \sum_{kl} y_{ikjl} \leq 1,\ \forall i \in V_1,\ \forall j \in V_1,\ i < j.$$

This IP formulation is an improved version of the one that we proposed in [9].

4. Branch and Bound approach for MCC. We present here a new branch and bound algorithm for solving the MCC problem in the previously defined alignment graph $G = (V, E)$. Let us first introduce some notions and notations. A **successor** of a vertex $i.k \in G$ is an element of the set $\Gamma^{+}(i.k) = \{j.l \in V \text{ s.t. } (i.k, j.l) \in E, i < j \text{ and } k < l\}$. Similarly, a **predecessor** of a vertex $i.k \in G$ is an element of the set $\Gamma^{-}(i.k) = \{j.l \in V \text{ s.t. } (j.l, i.k) \in E, j < i \text{ and } l < k\}$. $G^{\Gamma^{+}(i.k)}$, $G^{\Gamma^{-}(i.k)}$ denote the subgraphs of G induced by the vertices in $\Gamma^{+}(i.k)$and in $\Gamma^{-}(i.k)$. A **feasible path** in G is an ordered sequence "$i_1.k_1$, $i_2.k_2$, …, $i_t.k_t$" of vertices $\in V$, such that $\forall n \in [1, t-1]$, $(i_n.k_n, i_{n+1}.k_{n+1}) \in E$ and $i_n < i_{n+1}$, $k_n < k_{n+1}$.

Branching: Each node of the B&B tree is characterized by a couple (C, $Cand$) where C is the clique under construction and $Cand$ is the set of candidate vertices to be added to C. All B&B nodes can also access C_{best}, the best clique found so far during the exploration of the B&B tree (initially set to $\emptyset$). Starting from the root node $(\emptyset, V)$, successors of a B&B node $(C, Cand)$ are the nodes $(C \bigcup \{i.k\}, Cand \bigcap \Gamma^+(i.k))$, for all vertices $i.k \in Cand$. Branching follows the lexicographic increasing order (row first).

Fathoming: For a given a B&B node $(C, Cand)$ and a current best clique C_{best}, we denote by $MCC_{i.k}(G)$ the maximum cardinality clique in G containing vertex $i.k \in Cand$. If $|MCC_{i.k}(G)| \leq |C_{best}|$, then we do not miss the solution by discarding $i.k$ from $Cand$. Furthermore, denote by $C_{i.k}$ the best clique that can be found by branching on the vertex $i.k$, and let $MCC_{i.k}(G^{Cand})$ be the maximum cardinality clique in G^{Cand} (the subgraph of G induced by the vertices in $Cand$) containing $i.k$. It is easily seen that $|C_{i.k}| = |C| + |MCC_{i.k}(G^{Cand})|$. Any vertex $i.k \in Cand$ such that $|MCC_{i.k}(G^{Cand})| \leq |C_{best}| - |C|$ leads to non-interesting leaves, and thus, can be removed from $Cand$.

Bounds: We are not going to compute $|MCC_{i.k}(G)|$ or $|MCC_{i.k}(G^{Cand})|$, but we replace them with upper bounds based on feasible paths. Denote by $P(G)$ the longest (in terms of vertices) feasible path in G. Note that computing $|P(G)|$ can be done by dynamic programming in $O(|E|)$ time. For any vertex $i.k \in V$, we denote by $P_{i.k}(G)$ the longest feasible path in G containing $i.k$, such that for any vertex $j.l \neq i.k$ in the feasible path, $j.l$ is connected to $i.k$ (i.e. $(i.k, j.l) \in E$ or $(j.l, i.k) \in E$). By definition, $P_{i.k}(G) = P(G^{\Gamma^-(i.k)}) \bigcup \{i.k\} \bigcup P(G^{\Gamma^+(i.k)})$, and $|P_{i.k}(G)| = |P(G^{\Gamma^-(i.k)})| + 1 + |P(G^{\Gamma^+(i.k)})|$. It is easily seen that $|MCC_{i.k}(G)| \leq |P_{i.k}(G)|$ for all $i.k \in V$. Similarly, $|MCC_{i.k}(G^{Cand})| \leq |P_{i.k}(G^{Cand})|$ for all $i.k \in Cand$. Thus any vertex $i.k \in Cand$ such that: (i) $|P_{i.k}(G)| \leq |C_{best}|$, or (ii) $|P_{i.k}(G^{Cand})| \leq |C_{best}| - |C|$, can be safely removed from $Cand$.

5. Results.

All results were obtained on a PC with an Intel Pentium 4^{tm} CPU at 3GHz. The IP based solver (MIP) was implemented with Ilog Cplex 10.0, and the B&B solver was implemented in C. These two clique solvers were compared to (BK)[2] [5]. All algorithms were used to solve maximum cardinality clique problems. The comparison was performed on real protein structure comparison instances. We used two different benchmarks[3] which significantly differ by the number of secondary structure elements (SSE) per protein chain. The first benchmark, the Skolnick set, contains 40 small protein chains having from 5 to 20 SSEs. The second benchmark, the S2 set, contains 36 long protein

[2]VAST's clique solver, BK, returns all maximal cliques in a graph and thus can be used to solve any kind of clique problems.

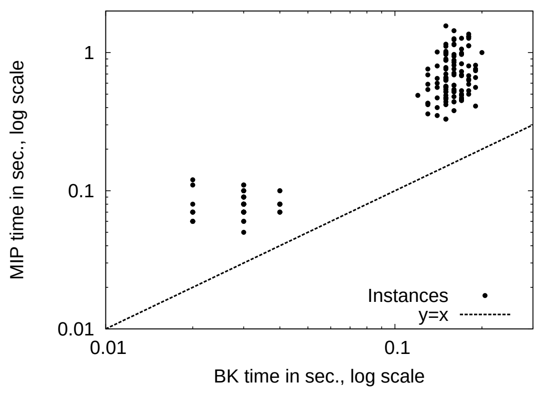
[3]The full description of both benchmarks is availlable at:
`https://www.irisa.fr/symbiose/old/softwares/resources/proteus300`

chains having from 51 to 87 SSEs. Note that for the Skolnick set, we only considered the 170 instances leading to alignement graphs having at least 100 vertices. Table 1 presents the characteristics of the corresponding alignment graphs. One peculiarity is their low density, less than 20% for the Skolnick set and less than 6% for the S2 set.

Table 1. Characteristics of the alignment graphs

Set name	Number of vertices min, average, max	Number of edges min, average, max	Density min, average, max
Skolnick	100, 158.92, 208	886, 2368.69, 3547	0.16, 0.18, 0.20
S2	1390, 2384.97, 5582	45278, 144206.44, 604793	0.03, 0.05, 0.06

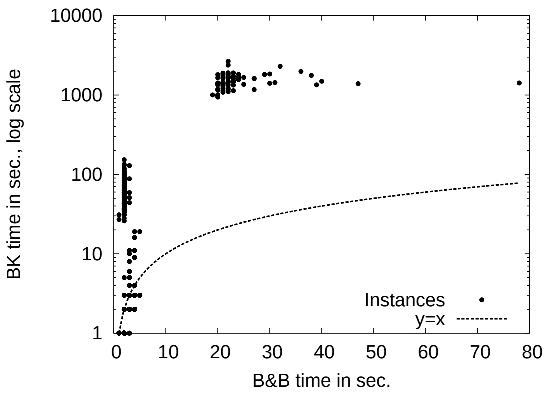
Figure 1 compares the time needed by MIP to the one of BK on the 170 Skolnick instances. On the average, MIP is 3.35 times slower than BK. This is not surprising, since dedicated solvers are expected to be faster than general purpose solvers (CPLEX in this case). This observation motivated us to go further in developing a fast special purpose clique solver. Figure 2 compares the



For each instance the execution time of MIP is plotted on the x-axis, while the one of BK is depicted on the y-axis. All points are above the $x = y$ line (i.e. BK is always faster than MIP).

Fig. 1. MIP vs BK running time comparison on a Skolnick set

time needed by B&B to the one of BK on set S2. We observed that B&B is in average 15.57 times faster than BK, and on the biggest instances (where both proteins contain more than 80 SSEs), it is up to 116.7 times faster. Such big instances are solved by B&B in less than 79 seconds (25 sec. on average) while BK needs up to 2660 seconds (1521 sec. on average).



The execution time of B&B is presented on the x-axis, while the one of BK is on the y-axis (in log scale). Any point above the $x = y$ line is an instance for which B&B is faster than BK.

Fig. 2. B&B vs BK running time comparison on an S2 set

6. Conclusion. We presented a new IP model for solving the maximum weighted clique problem arising in the context of protein structure comparison, which was implemented and validated on a small benchmark. We also presented a new dedicated B&B algorithm for the maximum cardinality clique problem. The computational results show that on big instances, our B&B is significantly faster than the Bron and Kerbosch algorithm (up to 116 times for the largest proteins). In the near future, we intend to study the behavior of the proposed algorithms on arbitrary graphs, conveniently transformed into grid graphs in a preprocessing step.

REFERENCES

[1] Abello J., P. M. Pardalos, M. G. C. Resende. On maximum clique problems in very large graphs, Ext. Mem. Alg., 1999, 119–130.

[2] Andonov R., N. Yanev, N. Malod-Dognin. An efficient lagrangian relaxation for the contact map overlap problem. In: Proceedings of WABI'08, Lecture Notes in Computer Science, Vol. **5251**, Springer, Berlin/Heidelberg, 2008, 162–173.

[3] Balas E., S. Ceria, G. Cornuejols, G. Pataki. Polyhedral methods

for the maximum clique problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, **26** (1996), 11–28.

[4] Bomze I. M., M. Budinich, P. M. Pardalos, M. Pelillo. The maximum clique problem. Handbook of Combinatorial Optimization, 1999.

[5] Bron C., J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, **16** (1973), No 9, 575–577.

[6] Busygin S. A new trust region technique for the maximum weight clique problem. *Discrete Appl. Math.*, **154** 2006, No 15, 2080–2096.

[7] Gibrat J-F., T. Madej, S. H. Bryant. Surprising similarities in structure comparison. *Current Opinion in Structural Biology*, **6** (1996), No 3, 377–385.

[8] Karp R. M. Reducibility among combinatorial problems. *Complexity of Computer Computations*, **6** (1972), 85–103.

[9] Malod-Dognin N., R. Andonov, N. Yanev, J-F. Gibrat. Modèle de PLNE pour la recherche de cliques de poids maximal. In: ROADEF 2008, 307–308.

[10] Pardalos P. M., G. P. Rodgers. A branch and bound algorithm for the maximum clique problem. *Comput. Oper. Res.*, **19** (1992), No 5, 363–375.

[11] Sierk M. L., G.J. Kleywegt. Déjà vu all over again: Finding and analyzing protein structure similarities .*Structure*, **12** (2004), No 12, 2103–2111.

N. Malod-Dognin, R. Andonov
IRISA–Université de Rennes 1
Campus de Beaulieu
35042 Rennes Cedex, France
e-mail: `nmaloddg@irisa.fr`, `randonov@irisa.fr`

N. Yanev
Faculty of Mathematics and Informatics
University of Sofia
and
Institute of Mathematics and Informatics
Acad. G. Bonchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: `choby@math.bas.bg`

Received November 2, 2009
Final Accepted February 4, 2010