

MBI TR, 2004
A Successful Application of Compact Optimization.
The Protein Contact Map Overlap Problem *

Robert D. Carr[†] Giuseppe Lancia[‡]

Abstract

In Combinatorial Optimization, one is frequently faced with linear programming (LP) problems with exponentially many constraints. These problems can be solved either using *separation* or what we call *compact optimization*. The former technique relies on a *separation algorithm*, which, given a fractional solution, tries to produce a violated valid inequality. The technique of compact optimization relies on describing the feasible region of the LP by a polynomial number of constraints, in a higher dimensional space.

A commonly held belief is that compact optimization does not perform as well as separation in practice. In this paper, we report on the first application in which compact optimization does in fact largely outperform separation. The problem is a fundamental question in computational molecular biology, i.e. the comparison of 3-dimensional protein folds. Our computational results show that compact optimization achieves an improvement of up to two orders of magnitude over separation. This fact allowed us to find provably optimal solutions to several real alignment problems of proteins from the Protein Data Bank, PDB [2]. We discuss some theoretical reasons why compact optimization works in this case but not, e.g., for the LP relaxation of the TSP.

1 Introduction

In Combinatorial Optimization, one is frequently faced with linear programming (LP) problems with exponentially many constraints. In fact, the use of such constraints

*This material is based upon work supported by the National Science Foundation under Agreement No. 0112050.

[†]Sandia National Laboratories, Albuquerque NM, USA.

[‡]Dipartimento di Matematica e Informatica, University of Udine. Via delle Scienze 206, 33100 Udine, Italy.

allows oftentimes tight formulations for Integer Programming (IP) problems to be solved by enumerative methods such as branch and bound. Among the most famous examples, we find NP-hard problems such as the Traveling Salesman Problem (with the exponentially many *subtour elimination* constraints) and the Steiner Tree Problem (with the *cutset inequalities*). However, even for some polynomial problems an LP formulation may require an exponential number of constraints. The most notable one is the weighted matching problem on a general graph, which, in its known LP formulation, requires the exponentially many *odd-clique* constraints.

The optimization of a linear function over a polyhedron P in a family $\mathcal{P} := (P_i \mid i \in \mathbf{N})$ of polyhedra whose description requires exponentially many inequalities in i can still be carried out, in many cases, in polynomial time in terms of i . There are essentially two ways of doing so. The first, due to Grötschel, Lovász and Schrijver [8], relies on expressing $P \in \mathcal{P}$ only implicitly, through a so called *separation algorithm*. This is a polynomial procedure that, given a point x , asserts that either $x \in P$, or returns an inequality that separates x from P . The second major technique for optimizing over exponentially-defined polyhedrons is what we call *Compact Optimization* [4, 18, 20]. In Compact Optimization, P is lifted to a polyhedron P' in a higher dimensional space (i.e. with additional variables) whose description is polynomially-sized in terms of i . Hence, optimization over P' is polynomial, e.g., by using the interior-point method for Linear Programming [17]. The optimal solution over P is then recovered by *projecting* back onto the space of the original variables. The idea of compact optimization is the subject of [18, 20] and [4], the latter of which shows precisely how separation and compact optimization relate to each other.

Although the idea of compact optimization has been informally circulating for a while, it has been fully developed only recently in [4], where we gave necessary and sufficient conditions under which compact optimization is possible. We suspect that the reason why it took so long for such an important concept to be considered as a viable optimization technique is that people may have tried it and dismissed as inferior to the use of separation algorithms. So, we are facing a commonly held belief that compact optimization does not perform as well as separation in practice. Although this is certainly the case for compact versions of the *subtour LP relaxation* of the famous Traveling Salesman Problem (TSP), this paper strives to show that compact optimization can and does outperform separation in some cases. The contact map overlap LP relaxation to a computational biology application, [15, 16, 10], is the example we use in this paper to make our point.

A *contact map* of a protein is an undirected graph representing the structure of the protein once folded in its 3D shape. There is a node for each of its amino acids, and two nodes are joined by an edge if the corresponding amino acids are “close enough” when the protein is folded. The *Contact Map Overlap* problem consists, given two contact maps, in finding their largest common sub-structure. This can be used as a measure of structure similarity, and exploited for protein clustering, data base search

and drug–design [9, 12]. Contact map overlap has emerged in the last few years as one of the most reliable and robust measure of protein structure similarity. Although the optimization problem is NP-hard [15], there are integer programming formulations for which the gap is relatively small, suggesting that this problem can be effectively solved by branch and bound. The formulation proposed in [10], for instance, is based on the use of exponentially many *clique inequalities* which can be separated in polynomial time. Although this procedure was successfully used to obtain, for the first time, the solution of some real–life alignment problems, it had to be limited to very small proteins, and even for those there were many instances which could not be solved. The reason is that the computation of the bound requires the solution of a large number of LPs, one after each run of the separation algorithm, and could take up to several hours of CPU time, even with *warm starting* (i.e. using the previous LP solution to solve the current linear program). By using compact optimization, for which the solution of a single LP is required, we have been able to reduce these times by up to two orders of magnitude. This way, we have been able to solve to optimality a large number of alignment problems (for proteins of larger size than before), and to give provably near–optimal solutions to many others.

The success of compact optimization on this particular problem raises a natural question about when a formulation is suitable to such approach. In this paper, we take first steps to addressing such a question and examine, in particular, the case of the subtour relaxation of the TSP, for which compact optimization does not perform similarly well.

The remainder of the paper is organized as follows. In section 2 we review the theory of separation and compact optimization. The Contact Map Overlap problem is described in section 3. In section 4 we investigate some theoretical reasons why compact optimization is effective for this problem and not, for example, for the TSP. Finally, in section 5 we report the results of our computational experiments.

2 Background: Separation, Optimization and their compact counterparts

2.1 Linear Programming Problems

A linear program consists in a linear *objective function*, or *cost function*, involving a finite set of *decision variables*, and a finite set of linear inequalities that the decision variables, and possibly finitely many *additional variables*, must satisfy. A *solution* to a linear program is a minimizer (maximizer) to the objective function (and its objective function value) subject to the constraints of the linear program, or else an assurance that the LP is unbounded or infeasible.

The problem of linear programming takes a constraint matrix A , right hand side

b , and a cost matrix c as input, and has as the goal to say minimize $c \cdot x$ subject to $A \cdot x \geq b$. In this paper, we are particularly interested in linear programs arising from some combinatorial object (such as a graph or a set), and for which the size of the data, i.e. of A , b , and c , is exponential with respect to the size of the combinatorial object. A classic example of this would be the subtour relaxation of the TSP, in which A contains exponentially many subtour elimination inequalities with respect to the number of vertices in the underlying graph.

Hence, for the purpose of this paper, it is a mistake to consider the A , b and c as given above to be the input to a linear programming problem. In fact, if our input is, e.g., a matrix A containing all subtour-elimination inequalities for a graph, we cannot call it an exponential-size LP, as it would not be explicit with respect to what it is exponential.

Therefore, we define the input of a linear programming problem to consist of both a cost function c and integers $n, m \in \mathbf{N}$ that encode the constraint matrix $A = A(n, m)$ and right hand side $b = b(n, m)$. By convention, n will encode the number of decision variables, and m any additional data (such as edge capacities), which are needed to determine A and b . For the subtour relaxation on a complete graph, n could be the number of vertices in this graph.

An *instance* of a linear programming problem arises from a particular input within the bounds specified above. The output of a linear programming problem is a solution to the linear program specified by the input for the instance at hand.

We define the *separation* problem for a particular linear programming problem as follows. The input is a number $n \in \mathbf{N}$, which determines the number n' of decision variables, any fractional point $x^* \in \mathbf{R}^{n'}$, and an $m \in \mathbf{N}$ encoding additional data determining the polyhedron for the linear program, as before. The output is either an inequality that is valid for this polyhedron, but is violated by this fractional point, or an assurance that x^* is in the polyhedron.

The famous Grötschel, Lovász, Schrijver theorem is the following:

Theorem 1 *A linear programming problem is polynomially solvable if and only if its separation problem is polynomially solvable.*

The “if” direction follows from solving a linear programming problem instance by applying the ellipsoid method with a provably polynomial number of calls to our separation algorithm. The “only if” direction can be proven by constructing a linear program that serves as a separation algorithm, and which can be solved in polynomial time by the “if” part. A similar proof to this sketch of the “only if” direction appears in [5].

2.2 Compact optimization and separation

We say that *compact optimization* of a linear programming problem is possible if there is an LP formulation (possibly using additional variables) of this problem whose linear

programs have sizes that are bounded by a polynomial of the size of the input to the linear programming problem. We say that *compact separation* of a linear programming problem is possible if there is an LP formulation of the separation problem for this linear programming problem whose linear programs have sizes that are bounded by a polynomial of the size of the input to this (the original) linear programming problem. In actuality, we restrict this LP formulation to have the special form shown in (1) to make our analysis easier, although this is not a fundamental restriction.

The main theorem relating these two concepts in [4] is the following.

Theorem 2 *Compact optimization of a linear programming problem is possible if and only if compact separation for the linear programming problem is possible.*

We give the LP formulations of [4] again here. Suppose an LP formulation of the separation problem for the original linear programming problem is given by:

$$\begin{aligned} \min \quad & x^* \cdot a - a_0 \\ \text{s.t.} \quad & \\ & (D_a \ D_{a_0} \ D_w) \cdot \begin{pmatrix} a \\ a_0 \\ w \end{pmatrix} \geq d. \end{aligned} \tag{1}$$

Then an LP formulation exhibiting compact optimization is given by:

$$\begin{aligned} \min \quad & c \cdot x \\ \text{s.t.} \quad & \\ & y \cdot D_a \leq x \\ & y \cdot D_{a_0} \leq 1 \\ & y \cdot D_w \leq 0 \\ & y \cdot d \geq 0. \end{aligned} \tag{2}$$

The converse of this is proved in [4].

We pose the following interesting, though probably false conjecture.

Conjecture 1 *Compact optimization (compact separation) is possible for a linear programming problem if and only if it is polynomially solvable.*

A potential counterexample to this conjecture is the *matching* problem on a general graph. In [21], Yannakakis gives a partial result concerning the impossibility of a compact formulation for the matching problem. He showed that any such compact formulation would have to have a lot of asymmetry, as defined in [21]. In the off-chance that Conjecture 1 is correct, an interesting consequence is that the ellipsoid method is no longer needed to prove the famous Grötschel, Lovász, Schrijver separation theorem.

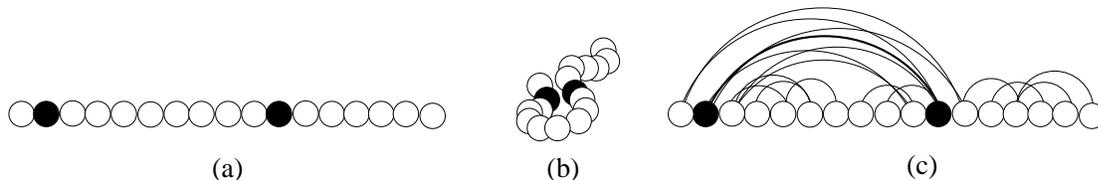


Figure 1: (a) An unfolded protein. (b) After folding. (c) The contact map graph.

3 The Contact Map Overlap Problem

A *protein* consists of a linear chain of *residues*, also called *amino acids*, along its *backbone*. In an aqueous solution, in which the protein occurs naturally, the backbone does not remain straight in its minimum energy *native* state, but folds into a very compact form (see Figure 1 (a) and (b)). The way a protein folds is often indicative of its function. Hence, it is valuable to have an abstract description of this folding that captures its important features. One such description is called a *contact map*.

A contact map for a protein is a graph. The vertices are the residues of the protein. The vertex set is linearly ordered by the order in which the residues occur on the backbone. There is an edge between two vertices whenever the distance between the residues is within a certain threshold (e.g., 5 Å) in the native state of the protein (see Figure 1 (c)). It is valuable to cluster proteins by making pairwise comparisons of the similarity of their folding patterns. A measure of similarity between two proteins created in [15] that uses the information of their folding patterns that their contact maps capture is the *maximum contact map overlap* (CMO).

The CMO tries to evaluate the similarity in the 3-D folds of two proteins by determining the similarity of their contact maps (the rationale being that a high contact map similarity is a good indicator of high 3-D similarity). Given two folded proteins, the CMO problem calls for determining an alignment between the residues of the first protein and of the second protein. The alignment specifies the residues that are considered equivalent in the two proteins. The goal is to find the alignment which highlights the largest set of common contacts. The value of an alignment is given by the number of pairs of residues in contact in the first protein which are aligned with pairs of residues that are also in contact in the second protein.

Translated in graph-theoretical language, the maximum contact map overlap is the following: Find the maximum number of edges in one of an isomorphic pair of subgraphs of two graphs in which the vertices are linearly ordered. The (bijective) isomorphism mapping between the vertices of the two graphs must preserve their ordering (i.e., it must be *non-crossing*: If a vertex i is aligned with a vertex j , then no vertex following i can be aligned with a vertex preceding j). We call the problem of finding the maximum contact map overlap the CMO problem. In Figure 2 we show two graphs and an alignment of value 5. The CMO problem was shown to be NP-hard

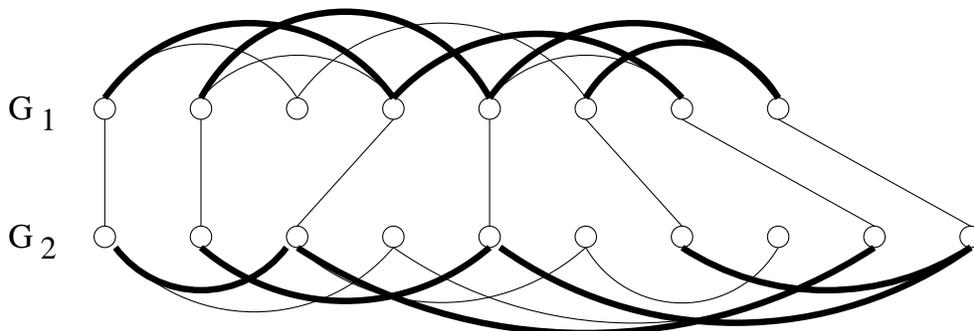


Figure 2: An alignment of value 5.

in [15]. An integer programming (IP) formulation, with contact maps $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, is as follows. There are 0-1 variables x_{ij} , for $i \in V_1$ and $j \in V_2$, representing the alignment of residues in the two proteins. Furthermore there are 0-1 variables y_{ef} , that are set to 1 whenever two contacts $e \in E_1$ and $f \in E_2$ are in common for an alignment. The IP formulation, first proposed in [10], is the following:

$$\begin{aligned}
& \max \quad \mathbf{1} \cdot y \\
& \text{s.t.} \\
& \quad x_{iu} + x_{jv} \leq 1 \quad \text{if } (i, u), (j, v) \text{ cross} \\
& \quad \sum_{j \in \delta^+(i)} y_{(i,j)(u,v)} \leq x_{iu} \quad \forall i \in V_1, \forall (u, v) \in E_2 \\
& \quad \sum_{j \in \delta^-(i)} y_{(j,i)(u,v)} \leq x_{iv} \quad \forall i \in V_1, \forall (u, v) \in E_2 \\
& \quad \sum_{v \in \delta^+(u)} y_{(i,j)(u,v)} \leq x_{iu} \quad \forall u \in V_2 \forall (i, j) \in E_1 \\
& \quad \sum_{v \in \delta^-(u)} y_{(i,j)(v,u)} \leq x_{iv} \quad \forall u \in V_2 \forall (i, j) \in E_1 \\
& \quad x, y \in \{0, 1\}.
\end{aligned}$$

The cryptic constraints relating the y variables to the x variables are just capturing the simpler relationship $y_{(i,j)(u,v)} \leq \min(x_{iu}, x_{jv})$.

The Linear programming relaxation for this IP formulation is actually quite weak because the non-crossing relationship between the x variables was not fully exploited. But if we strengthen this LP relaxation with the *clique inequalities* for the x variables described next, it becomes fairly tight. A subset $F \subset V_1 \times V_2$ of indices of x variables form a *clique* if any two $(u, v), (u', v') \in F$ cross when G_1 and G_2 are drawn true to the linear orderings of V_1 and V_2 . Given a clique $F \subset E$, the valid inequality

$$\sum_{(u,v) \in F} x_{uv} \leq 1$$

is called a clique inequality.

Although the LP relaxation with all clique inequalities has exponentially many constraints, it can still be solved in polynomial time because there are polynomial time separation algorithms for the class of clique inequalities. Two different such

separation algorithms are given in [11] and [10]. They are both based on *dynamic programming* (or finding the *longest path* in an acyclic graph), and have similar time performance bounds. The former algorithm is simpler and seems generally preferable, so we present just this algorithm here.

Construct a directed grid graph whose vertices are the pairs $(i, u) \in V_1 \times V_2$. With $n_i := |V_i|$ for $i = 1, 2$, put vertex $(1, n_2)$ in the lower left corner and vertex $(n_1, 1)$ in the upper right corner.

Consider an internal vertex (i, u) . Have its neighbors below and above it be $(i, u+1)$ and $(i, u-1)$ respectively. Direct these edges from below to above, with the length of edge $((i, u+1), (i, u))$ being x_{iu}^* . Have its neighbors to its left and right be $(i-1, u)$ and $(i+1, u)$ respectively. Direct these edges from left to right, with the length of edge $((i-1, u), (i, u))$ being x_{iu}^* .

Then the most violated clique inequality (if any) is found by taking the longest path from $(1, n_2)$ to $(n_1, 1)$ and adding x_{1, n_2}^* to the result. There is a violated clique inequality if and only if this result is greater than 1.

We could solve our LP relaxation using the traditional approach of running this separation algorithm to find violated clique inequalities, and then adding these *cuts* to the linear program. However, the subject of this paper is that there is often an alternative. In this case, compact separation is also possible since there is a compact LP formulation for the problem of finding the longest path in an acyclic graph with non-negative path lengths. A compact LP formulation of this separation algorithm is as follows. In this formulation, $\delta^+(v)$ ($\delta^-(v)$) is the set of edges out of (into) a vertex v , $E \subset (V_1 \times V_2)^2$ is the set of edges of the grid graph, and for $F \subset E$, $a(F) := \sum_{e \in F} a_e$.

$$\begin{aligned}
& \max && x^* \cdot a - a_0 \\
& \text{s.t.} && \\
& && a_0 = 1 \\
& && a_{1, n_2} = 1 \\
& && a'(\delta^+(v)) = a'(\delta^-(v)) \quad \forall v \in V_1 \times V_2 \setminus \{(1, n_2), (n_1, 1)\} \\
& && a'(\delta^+((1, n_2))) = 1 \\
& && a'(\delta^-((n_1, 1))) = 1 \\
& && a_{i, u} = a'(\delta^-((i, u))) \quad \forall (i, u) \in V_1 \times V_2 \setminus \{(1, n_2)\} \\
& && a'_e \geq 0 \quad \forall e \in E
\end{aligned}$$

As we know, this means that compact optimization is possible for this new LP relax-

ation. A compact LP formulation is as follows:

$$\begin{aligned}
& \max && \mathbf{1} \cdot y \\
& \text{s.t.} && \\
& && \sum_{j \in \delta^+(i)} y_{(i,j)(u,v)} \leq x_{iu} \quad \forall i \in V_1, \forall (u,v) \in E_2 \\
& && \sum_{j \in \delta^-(i)} y_{(j,i)(u,v)} \leq x_{iv} \quad \forall i \in V_1, \forall (u,v) \in E_2 \\
& && \sum_{v \in \delta^+(u)} y_{(i,j)(u,v)} \leq x_{iu} \quad \forall u \in V_2 \forall (i,j) \in E_1 \\
& && \sum_{v \in \delta^-(u)} y_{(i,j)(v,u)} \leq x_{iv} \quad \forall u \in V_2 \forall (i,j) \in E_1 \\
& && l_{n_1,1} \leq 1 \\
& && l_v - l_{v'} \geq x_{i,u} \quad \forall (v',v) \in \delta^-((i,u)) \forall (i,u) \in V_1 \times V_2 \\
& && l_{1,n_2} = x_{1,n_2} \\
& && 0 \leq x, y \leq 1
\end{aligned}$$

We will give empirical evidence in section 5 that indicates that using a compact formulation actually speeds up the solution of the LP relaxation over the traditional approach using a separation algorithm. A preliminary theoretical analysis of the reasons why this might be the case is given in the next section.

4 A Theoretical Analysis of Compact Optimization

4.1 The Traveling Salesman Problem

Let $G = (V, E)$ be an undirected graph, with a cost c_e for each edge $e \in E$. A *Hamilton cycle* in G is a cycle that visits every vertex in V exactly once. The goal of the *traveling salesman problem* (TSP) is to find a Hamilton cycle in G of minimum cost. Typically, it is assumed that G is a complete graph, and that the costs satisfy the *triangle inequality*. An integer programming formulation for the TSP, where $\delta(S)$ is the set of edges with exactly one endpoint in $S \subset V$, is as follows:

$$\begin{aligned}
& \min && c \cdot x \\
& \text{s.t.} && \\
& && x(\delta(v)) = 2 \quad \forall v \in V \\
& && x(\delta(S)) \geq 2 \quad \forall \emptyset \neq S \subset V \\
& && x_e \in \{0, 1\} \quad \forall e \in E.
\end{aligned}$$

The LP relaxation that results from relaxing the integrality constraints is called the *subtour relaxation*. The constraints in one-to-one correspondence with the subsets $S \subset V$ are called the *subtour elimination constraints*. Although there are exponentially many such constraints, the subtour relaxation can be solved in polynomial time because there is a polynomial time separation algorithm for this class of inequalities. This algorithm involves finding a minimum cut in a capacitated graph, whose capacities are given by the (non-negative) edge variable values of the fractional point x^* .

But, in fact, compact separation is possible too. A compact LP formulation of the separation algorithm is as follows:

$$\begin{aligned}
& \min && x^* \cdot a - a_0 \\
& \text{s.t.} && \\
& && a_0 = 2 \\
& && a_{ij}^k \geq u_i^k - u_j^k && \forall k \in V \setminus \{1\} \forall \{i, j\} \in E \\
& && a_{ij}^k \geq u_j^k - u_i^k && \forall k \in V \setminus \{1\} \forall \{i, j\} \in E \\
& && u_1^k = 0 \\
& && \sum_{k \in V \setminus \{1\}} u_k^k = 1 \\
& && a_{ij} = \sum_{k \in V \setminus \{1\}} a_{ij}^k && \forall \{i, j\} \in E.
\end{aligned}$$

This means that compact optimization is possible for the subtour relaxation as well. A compact LP formulation, which was independently re-discovered by several authors [6, 19], is as follows:

$$\begin{aligned}
& \min && c \cdot x \\
& \text{s.t.} && \\
& && x(\delta(v)) = 2 && \forall v \in V \\
& && f^k(\delta^+(1)) = 2 && \forall k \in V \setminus \{1\} \\
& && f^k(\delta^-(k)) = 2 && \forall k \in V \setminus \{1\} \\
& && f^k(\delta^+(v)) = f^k(\delta^-(v)) && \forall v \in V \setminus \{1, k\} \forall k \in V \setminus \{1\} \\
& && f_e^k \leq x_e && \forall e \in E \forall k \in V \setminus \{1\} \\
& && 1 \geq x_e \geq 0 && \forall e \in E.
\end{aligned}$$

There are in fact several such compact formulations known for the subtour relaxation, [3][1]. The cycle-shrink relaxation (Carr), [3], is particularly interesting since a similar compact LP formulation may be possible for the matching problem, and would not be ruled out by Yannakakis' partial result because of cycle-shrink's high degree of asymmetry.

4.2 On the success of compact optimization

Unlike in the contact map overlap application, compact optimization for the subtour relaxation does not seem to be competitive with the traditional approach for solving the subtour relaxation using separation.

The choice between using a traditional separation approach and a compact optimization approach hinges on whether it is best to solve many smaller linear programs (aided by warm starts) and equally many calls to a separation algorithm, or it is best to solve one larger linear program. In order to answer these questions for the contact map overlap and the subtour linear programs, we look at the sizes of the linear programs to be solved using both the traditional approach and compact optimization.

We use the simplistic models that the time needed to solve a linear program is proportional either to the number of non-zeros in the constraint matrix or the number of rows times the number of columns in this matrix.

For the contact map overlap linear program, the number of non-zeros, rows, and columns in the linear program one starts with when using the traditional separation approach are described in the following tables. First, the number of nonzeros:

	# of nonzeros =
Relating x to y	$4 E_1 E_2 + 2 V_1 E_2 + 2 V_2 E_1 +$
Lower, Upper bounds	$2 E_2 E_1 + 2 V_2 V_1 $

Then the number of rows and columns, excluding the upper and lower bounds:

# of rows	# of columns
$2 V_1 E_2 + 2 V_2 E_1 $	$ E_1 E_2 + V_1 V_2 $

Using compact optimization instead of the traditional separation approach adds only $6|V_1||V_2| + 1$ more non-zeros, $2|V_1||V_2| + 1$ more rows, and $|V_1||V_2|$ more columns. Even with sparse contact maps, say $|E_i| = 2|V_i|$, the linear program for compact optimization has about only $\frac{4}{3}$ times the number of non-zeros as the original linear program. By our crude model, it may take only a small constant times the time to solve this compact optimization LP as is needed for solving just one original LP. We will comment on this in the next section. Clearly, compact optimization is worth trying in this case.

Now lets consider the subtour relaxation. The original number of non-zeros before adding in the subtour elimination constraints is $2|E|$ (degree constraints) $+|E|$ (bounds). The original number of rows is $|V|$, and the original number of columns is $|E|$. Resorting to compact optimization adds $4|E|(|V| - 1)$ more non-zeros, $(|V| - 1)^2 + |E|(|V| - 1)$ more rows, and $(|V| - 1)|E|$ more columns. In other words, the compact optimization LP is an order of magnitude larger, and so it makes sense that the traditional approach would do better here.

5 Computational Results

In order to compare the two approaches of compact optimization and separation for the contact map problem, we performed the following test. Out of a large set of pairs of real proteins, coming from the public-domain Protein Data Base (PDB, [2]) and suggested to us by J. Skolnick [13, 14], we picked a random subset of 20 problems.

Table 1: Separation vs. Compact Optimization.

Instance				SEPARATION				COMPACT OPTIMIZATION				
prot1	prot2	n	m	cols	rows	nLPs	time	cols	rows	nLPs	time	spedup
1b3c	1svf	92	101	4359	6085	1944	28072	6474	10225	1	207	136x
1nmg	1svf	92	112	4722	6932	2371	38877	6837	11072	1	314	124x
1svf	2b3c	92	101	4359	6085	2118	26996	6474	10225	1	232	116x
1bw5	1svf	96	91	4209	5393	1776	14010	6504	9889	1	136	103x
1bct	1h1h	105	104	5354	7196	1477	18159	8104	12593	1	186	98x
1bw5	1joy	100	115	5805	8055	2434	62426	8304	12955	1	663	94x
1svf	1szt	97	101	4584	6349	1118	9744	6924	10934	1	142	69x
1svf	2new	93	91	4074	5624	1349	10147	6234	9853	1	149	68x
1joy	1svf	94	90	4086	5667	1174	6719	6291	9985	1	115	58x
1f22	1svf	93	88	3975	5423	827	4937	6135	9652	1	110	45x
1h1h	2new	98	120	5996	10658	1126	28941	8396	13112	1	829	35x
1qr9	1svf	100	105	4851	6738	454	3328	7326	11590	1	116	29x
1mdy	1svf	100	89	4323	5545	558	2382	6798	10397	1	83	29x
1bhb	1svf	97	104	4683	6352	442	2860	7023	10937	1	165	17x
1bct	1svf	100	75	3861	4662	316	739	6336	9514	1	54	14x
1tn9	1bmr	109	191	12068	15432	328	73449	15036	21164	1	5185	14x
1sfc	1svf	101	86	4269	5773	173	513	6789	10714	1	49	10x
1svf	1wdc	99	82	4047	5318	170	427	6477	10081	1	61	7x
1fza	1fzb	145	194	14664	21142	144	7811	19920	31511	1	1471	5x
1ehj	1f22	100	116	5851	8346	65	284	8347	13240	1	96	3x

For each, we solved the LP corresponding to the root node of a branch-and-cut by both separation and compact optimization. The results are reported in Table 1. The proteins sizes are of up to 70 residues and 100 contacts each. These are larger values than the ones dealt with in [10]. Columns n and m report the total number of residues and contacts per problem. Running times are in seconds, on a Pentium PC 266Mhz, with LP solver CPLEX6.0. Column “nLPs” reports the total number of LPs requires by each of the two approaches. The results show that compact optimization largely outperforms separation on this problem, with impressive improvements, ranging from 3 times (minimum) to 136 times faster (maximum). On average, compact optimization was 53 times faster than separation. Since separation can take several hours per node, branch and cut could not have been performed on these problems. However, branch and bound with compact optimization could. We implemented such an algorithm and tried it on 597 problems from the PDB. We were able to solve to optimality 42 problems, and for 362 problems (60 percent) the gap between best solution and best upper bound was at most 5. The results of the branch-and-bound are reported in table 2. The lower bounds were obtained by simple local search heuristics not described here. For details on the heuristics procedure used, the reader is referred to [10].

Table 2: Branch-and-bound performance.

Gap =	0	1	2	3	4	5	> 5
# instances	42	48	72	71	76	95	193
avg # residues	66.4	66.8	66.7	67.0	77.03	66.8	66.8
max # residues	69	72	71	72	71	72	80
avg # contacts	61.1	56.3	57.3	59.7	61.5	64.7	71.4
max # contacts	92	89	93	95	88	89	133

References

- [1] T.S. Arthanari (1982) On the Traveling Salesman Problem, presented at the Symposium on Mathematical Programming, Bonn, West Germany.
- [2] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I.N. Shindyalov and P.E. Bourne (2000) The Protein Data Bank, *Nucleic Acids Research*, 28:235–242.
- [3] R. Carr (1996) Separating Over Classes of TSP Inequalities Defined by 0 Node Lifting in Polynomial Time, Proceedings of Integer Programming and Combinatorial Optimization (IPCO).
- [4] R. Carr and G. Lancia (2002) Compact vs. Exponential-size LP Relaxations, *Operations Research Letters*, 30(1):57–65.
- [5] R. Carr and S. Vempala (2000) Randomized Metarounding, Proceedings of the Symposium on the Theory of Computing (STOC).
- [6] A. Claus (1984) A new formulation for the travelling salesman problem, *SIAM Journal on Algorithms and Discrete Methods* 5:21–25.
- [7] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank and A. Schrijver (1998) *Combinatorial Optimization*, John Wiley and Sons.
- [8] M. Grötschel, L. Lovász and A. Schrijver (1981) The Ellipsoid Method and its Consequences in Combinatorial Optimization, *Combinatorica* 1:169–197.
- [9] I. Eidhammer, I. Jonassen and W. R. Taylor (2000) Structure Comparison and Structure Prediction, to appear *Journal of Computational Biology*.
- [10] G. Lancia, R. Carr, B. Walenz and S. Istrail (2001) 101 Optimal PDB Structure Alignments: A Branch-and-Cut Algorithm for the Maximum Contact Map Overlap Problem, Proceedings of 5th ACM Intl. Conf. on Computational Molecular Biology (RECOMB), 193–202.

- [11] H. P. Lenhof, K. Reinert and M. Vingron (1998) A Polyhedral Approach to RNA Sequence Structure Alignment, *Journal of Computational Biology*, 5(3):517–530.
- [12] A. Godzik (1996) The structural alignment between two proteins: Is there a unique answer?, *Protein Science*, 5:1325–1338.
- [13] A. Godzik, J. Skolnick and A. Kolinski (1992) A topology fingerprint approach to inverse protein folding problem, *Journal of Molecular Biology*, 227:227–238.
- [14] A. Godzik and J. Skolnick (1994) Flexible algorithm for direct multiple alignment of protein structures and sequences, *CABIOS*, 10(6):587–596.
- [15] D. Goldman, S. Istrail and C. Papadimitriou (1999) Algorithmic Aspects of Protein Structure Similarity, Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science, 512–522.
- [16] D. Goldman (2000) PhD Thesis, Dept. of Computer Science, UC Berkeley.
- [17] N. Karmarkar (1984) A New Polynomial-Time Algorithm for Linear Programming, Proceedings of ACM Symposium on Theory of Computing (STOC), 302–311.
- [18] K. Martin (1991) Using Separation Algorithms to Generate Mixed Integer Model Reformulations, *Operations Research Letters* 10:119–128.
- [19] A. Tamir (1991) On the core of network synthesis games, *Mathematical Programming*, 50:123–135.
- [20] R.T. Wong (1984) A dual ascent approach for Steiner tree problems on a directed graph, *Mathematical Programming*, 28:271–287.
- [21] M. Yannakakis (1988) Expressing Combinatorial Optimization Problems by Linear Programs, Proceedings of ACM Symposium on Theory of Computing (STOC), 223–228.