Algorithmic Operations Research Vol.3 (2008) 110–129

Quadratic Binary Programming Models in Computational Biology*

Richard J. Forrester

Dickinson College, USA

Harvey J. Greenberg

Denver, Colorado, USA

Abstract

In this paper we formulate four problems in computational molecular biology as 0-1 quadratic programs. These problems are all NP-hard, and the current solution methods used in practice consist of heuristics or approximation algorithms tailored to each problem. Using test problems from scientific databases, we address the question, "Can a general-purpose solver obtain good answers in reasonable time?" In addition, we use the latest heuristics as incumbent solutions to address the question, "Can a general-purpose solver confirm optimality or find an improved solution in reasonable time?" Our computational experiments compare four different reformulation methods: three forms of linearization and one form of quadratic convexification.

Key words: integer programming, quadratic binary programming, computational biology, sequence alignment, protein folding, contact map overlap, rotamer assignment, protein similarity

AlgOR established a Supplementary Site (http://journals.hil.unb.ca/index.php/AOR/rt/suppFiles/5930/9786), which contains code and data for the experiments reported in the paper.

1. Introduction

We present Quadratic Binary Programming (QBP) models of four problems in computational molecular biology. The problems have been subject to computational research for more than a decade, but each has been formulated in its own way and solved by special algorithms designed for the individual problem under consideration. We address the question, "Can these problems be solved by the same general-purpose solver?" An advantage of using a general-purpose modeling framework, like QBP, is that the algorithm does not depend on the model. Although applying a common strategy can require more computational time to reach a solution, it has the advantage of providing an "umbrella" that makes it unnecessary for the scientist to know the details of numerous algorithms. It also enables the incorporation of new features, such as additional constraints, with no new algorithm design.

In this paper we consider two solution strategies for solving the QBP. The first strategy, called *linearization*, is to reformulate the QBP as a Mixed-Integer Linear Program (MILP) through the introduction of auxiliary variables and constraints. Then, the reformulation can be solved using any standard mixed-integer linear solver. The second strategy, known as *convex quadratic reformulation*, is to rewrite the QBP into an equivalent problem with a convex quadratic objective function. This does not require auxiliary variables or constraints, but it does require an adjustment to the objective, called *convexification*, to make the quadratic form a convex function.

This paper is organized as follows. In §2we give some background for understanding the problems and QBP approaches, leaving details to the references we cite. Also, we assume some familiarity with mathematical programming terminology — see the *Mathematical Programming Glossary* [28] for details. In §3. we present four models and numerical results for our test problems. We conclude with some observations and indications of avenues for further research.

2. Background

We begin this section by providing background in the basic, underlying biology for the problems we study.

Email: Richard J. Forrester [forrestr@dickinson.edu], Harvey J. Greenberg [hjgreenberg@gmail.com].

We then briefly review our approaches to solving QBP, namely linearization and convex quadratic reformulations. Finally, we discuss the data source for our test problems, which is primarily the Protein Data Bank (PDB) [7]. We further describe how we implemented our models to obtain the numerical results in §3. Related descriptions of linear integer programming models are given in [9,24], as well as references to more background material.

2.1. *A Little Molecular Biology*

All life depends on three critical molecules:

(1) DNA, which contains information about how a cell works;
(2) RNA, which makes proteins and performs other functions;
(3) Proteins, which are regarded as the workers of the cell.

DNA is a double-stranded sequence of nucleic acids: Adenine, Cytosine, Guanine, and Thymine. RNA is a single-stranded sequence of nucleic acids: Adenine, Cytosine, Guanine, and Uracil. The genetic code maps each triple of nucleic acids into one of 20 amino acids. A *peptide bond* is a bonding of two amino acids. A protein is determined by a sequence of successively bonded amino acids.

The *central dogma* is the information flow: DNA $\Rightarrow$ mRNA $\Rightarrow$ Protein. The first mapping, from the DNA molecule to messenger RNA, is called *transcription*. It removes some of the nucleic acids, leaving only those that code for the protein. The second mapping, from mRNA to a protein, is called *translation*, which uses the *genetic code* to map triples of nucleic acids into amino acids. Proteins are formed by the bonding of the sequence of amino acids. Figure 1 shows how two amino acids bond — the *carboxyl end* of the first amino acid bonds with the *amino end* of the second.

The amino acids have certain properties: basic vs. acidic, polar vs. non-polar, and hydrophilic (water loving) vs. hydrophobic (water hating). This last property has special meaning in our study, upon which we elaborate in §3.2..

In principle, the function of a protein is determined by its amino acid sequence, but full understanding of that mapping is beyond current science. It embodies a celebrated problem in computational biology, known as the *protein folding* problem: How does the sequence of amino acids determine the structure of the protein? Related to this is understanding how structure deter-

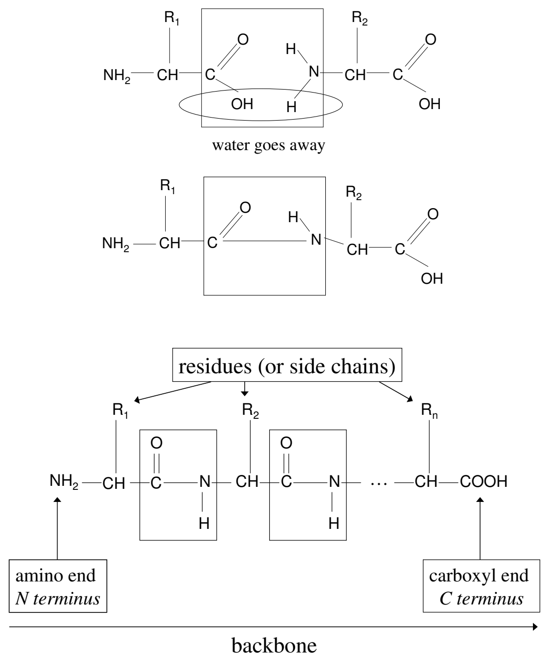


Fig. 1. Amino Acids Bonding

mines function — that is, whether the protein transports molecules (like oxygen from the lung to the brain), catalyzes other reactions, protects the organism from disease, or performs some other function. Knowledge of such things has been increasing dramatically due to new ways of getting huge amounts of data and developing algorithms to process the data. Optimization plays a fundamental role in all of this, and we illustrate with four basic problems in the next section.

This should be enough biology terms and concepts to get started; consult [11,29] for broader and deeper introductions. Further, we define more basics as needed when we develop the models.

2.2. *Solution Strategies*

We provide in this section details of both the linear and convex quadratic reformulation strategies. To establish notation, consider the general form of the quadratic binary program:

$$\max\ cx + \tfrac{1}{2}x'Qx :\ x \in X \cap \{0,1\}^n, \qquad (1)$$

where X denotes a polyhedral set. We assume, without loss in generality, that Q is symmetric, and $Q_{ii} = 0$ (x_i^2 term can be included in the linear portion since $x_i^2 = x_i$).

Although all of the biology models presented in this paper have binary variables with multiple indices, and some have continuous variables as well, we use (1) in this section to simplify the discussion of the reformulation strategies. Specific implementation details can be found in our model files at this article's supplement site.

2.2.1. *Linearization*

The first class of reformulation methods we consider is linearization. The strategy is to convert the quadratic program into an equivalent MILP through the introduction of auxiliary variables and constraints. The linear reformulation is then solved by a standard MILP algorithm.

There are many different linearization strategies in the literature (see [2,3] for recent surveys). By definition, they are all equivalent when the binary restrictions are enforced; however, their formulation size and continuous relaxation strength can vary greatly. We consider three methods here: *Standard, Reformulation-Linearization Technique* (RLT), and *Glover's.*

A standard way to linearize the QBP is to replace each product $x_i x_j$ in the objective function with the continuous variable w_{ij} and add four linear inequalities as auxiliary constraints: $w_{ij} \le x_i$, $w_{ij} \le x_j$, $w_{ij} \ge x_i + x_j - 1$, and $w_{ij} \ge 0$. Collectively, these imply $w_{ij} = x_i x_j$ for all binary values of x. Incorporating the symmetry reduction, the domain of w is $\mathrm{dom}_{\mathrm{std}}(w) = \{(i,j) : i < j, Q_{ij} \ne 0\}$. We can reduce the number of auxiliary constraints by using optimality and the sign of Q_{ij}. Let $Q^+ = \{(i,j) \in \mathrm{dom}_{\mathrm{std}}(w) : Q_{ij} > 0\}$ and $Q^- = \{(i,j) \in \mathrm{dom}_{\mathrm{std}}(w) : Q_{ij} < 0\}$. Then, we can omit the upper bound constraints for $(i,j) \in Q^-$, and we can omit the lower bound constraints for $(i,j) \in Q^+$. We thus define the *Standard linearization* of QBP:

$$\begin{aligned}
&\max\ cx + \sum_{(i,j)\in \mathrm{dom}_{\mathrm{std}}(w)} Q_{ij} w_{ij} : x \in X \cap \{0,1\}^n \\
&w_{ij} \ge x_i + x_j - 1 \text{ and } w_{ij} \ge 0 \quad \text{for } (i,j) \in Q^- \\
&w_{ij} \le x_i \qquad\qquad \text{and } w_{ij} \le x_j \quad \text{for } (i,j) \in Q^+.
\end{aligned}$$

We do not require the auxiliary variables (w) to be binary; that is implied by binary values of x. This tells the solver to branch on only the primary decision variables (x). Also, note that we do not declare $w_{ij} \ge 0$ unless it is necessary. This can provide a computational advantage because if a "free variable" [28] enters the Linear Programming Relaxation (LPR) basis it will remain there.

A strengthening of the Standard linearization is the *Reformulation Linearization Technique* (RLT) introduced and developed by Adams and Sherali [5,34]. The RLT can be used to generate a hierarchy of progressively tighter linear programming relaxations. The level-1 RLT representation is constructed by multiplying the constraints of X by each binary variable x_i and its complement, $1 - x_i$, and then using the auxiliary variable w_{ij} to replace each product term $x_i x_j$, along with setting $x_i^2 = x_i$.

Given the sizes of the problems in molecular biology that we consider, the full level-1 RLT is, in general, not a viable approach due to its huge memory requirements, and therefore we consider a $partial$ level-1 RLT representation here.[1] The idea is to include only a subset of the RLT restrictions by judicially selecting subsets of constraints and variables from which to generate the RLT restrictions. We illustrate below by applying the RLT to a selection (or assignment) constraint. Such constraints are prevalent in biology problems that involve combinatorial optimization. The specific details of each partial RLT implementation are discussed in §3.

Suppose that we require

$$\sum_{j\in J} x_j = 1 \tag{2}$$

for some index set J. Multiply (2) by x_i for $i \notin J$ to obtain the quadratic constraints:

$$\sum_{j\in J} x_i x_j = x_i, \forall i \notin J. \tag{3}$$

Now substitute $w_{ij} = x_i x_j$ for $i < j$ to reformulate (3) as the RLT constraints

$$\sum_{j\in J:\, i<j} w_{ij} + \sum_{j\in J:\, i>j} w_{ji} = x_i, \forall i \notin J. \tag{4}$$

The constraints (4) are then added to the Standard linearization to strengthen the LPR. We point out that the domain of w is generally much larger than that of the Standard linearization as it typically includes a large number of w_{ij} variables for which $Q_{ij} = 0$. In particular, the domain of w for the level-1 RLT is

$$\mathrm{dom}_{\mathrm{RLT}}(w) = \{(i,j) : i < j\}.$$

[1] Some of the many open research questions about using the RLT are discussed in our conclusions (§4.).

Our partial RLT formulations use a subset of this.

As we shall see in the experiments, the RLT constraints can be a very strong addition to the Standard linearization, sometimes enabling the root LP to solve the MILP, but the price can also be enormous (to the point of requiring so much memory that the problem cannot be setup). The (partial) *RLT linearization* of QBP is thus defined:

$$\max\ cx + \sum_{(i,j)\in \mathrm{dom}_{\mathrm{RLT}}(w)} Q_{ij}w_{ij} \,:\, x \in X \cap \{0,1\}^n,$$
$$\textstyle\sum_{j\in J} x_j = 1$$
$$w_{ij} \geq x_i + x_j - 1 \qquad \text{for } (i,j) \in Q^-$$
$$w_{ij} \leq x_i \text{ and } w_{ij} \leq x_j \ \text{ for } (i,j) \in Q^+$$
$$\textstyle\sum_{j\in J:\, i<j} w_{ij} + \sum_{j\in J:\, i>j} w_{ji} = x_i \text{ for } i \notin J$$
$$w \geq 0.$$

Note that we include $w \geq 0$ for all auxiliary variables, as opposed to just those w_{ij} with $(i,j) \in Q^-$. While this is not necessary for the RLT restrictions to be valid, computational experience has shown that this is beneficial because the nonnegativity restrictions enhance the strength of the RLT constraints.

A compact linearization strategy that we consider is the formulation by Glover [20]. This formulation is more economical than the Standard linearization in terms of the required number of auxiliary variables and constraints. The method replaces for each i the expression $x_i \sum_{j>i} Q_{ij}x_j$ in the objective with a continuous variable w_i, and enforces that $w_i = x_i \sum_{j>i} Q_{ij}x_j$ for binary x through the introduction of four linear auxiliary constraints. (Recall that we assume Q is symmetric, so the quadratic portion of the objective function satisfies $\frac{1}{2}\, x'Qx = \sum_i \sum_{j=i+1}^n x_iQ_{ij}x_j$.) This method results in the following formulation of QBP:

$$\max cx + \textstyle\sum_{i=1}^n w_i : x \in X \cap \{0,1\}^n$$
$$L_ix_i \leq w_i \leq U_ix_i$$
$$w_i \geq \textstyle\sum_{j>i} Q_{ij}x_j - U_i(1-x_i)$$
$$w_i \leq \textstyle\sum_{j>i} Q_{ij}x_j - L_i(1-x_i)$$
$$\forall\, i,$$

where L_i and U_i are lower and upper bounds, respectively, on $\sum_{j>i} Q_{ij}x_j$.

When $x_i = 0$, the first pair of inequalities forces $w_i = 0$, while the second pair is redundant. When $x_i = 1$, the first pair is redundant, and the second pair forces $w_i = \sum_{j>i} Q_{ij}x_j$. Thus, w_i equals the desired value for each binary value of x_i.

The bounds (L, U) can be computed as

$$L_i = \min\ \left\{\textstyle\sum_{j>i} Q_{ij}x_j \,:\, x \in X \cap \{0,1\}^n\right\}$$
$$U_i = \max\ \left\{\textstyle\sum_{j>i} Q_{ij}x_j \,:\, x \in X \cap \{0,1\}^n\right\}.$$

For the sake of computational ease, any relaxation can be used, such as

$$L_i = \sum_{j:(i,j)\in Q^-} Q_{ij} \text{ and } U_i = \sum_{j:(i,j)\in Q^+} Q_{ij}.$$

As shown in [1], two simple modifications of Glover's original formulation reduces the number of auxiliary constraints. First, the lower bounds on w are redundant at optimality and can therefore be removed. Second, the number of structural constraints can be further reduced by the substitution of variables: $s_i = U_ix_i - w_i$ for each i. This substitution enables the replacement of structural constraints with the same number of non-negativity restrictions. This yields the linearization:

$$\max\ cx + \sum_{i=1}^n (U_ix_i - s_i) \,:\, x \in X \cap \{0,1\}^n,\ s \geq 0$$
$$s_i \geq (U_i - L_i)x_i - \textstyle\sum_{j>i} Q_{ij}x_j + L_i \ \ \forall\, i.$$

In this formulation, the number of auxiliary variables (s) equals the number of binary variables (n), but we may be able to reduce it further.

The domain of s can be reduced to those for which $Q_{ij} \neq 0$ for some $j > i$ (otherwise, $\sum_{j>i} Q_{ij}x_j$ is identically zero for all x). We thus define the (extended) *Glover linearization* of QBP:

$$\max\ cx + \sum_{i\in I} (U_ix_i - s_i) \,:\, x \in X \cap \{0,1\}^n,\ s \geq 0$$
$$s_i \geq (U_i - L_i)x_i - \textstyle\sum_{j>i} Q_{ij}x_j + L_i \ \ \forall\, i \in I,$$

where $I = \{i \,:\, Q_{ij} \neq 0 \text{ for some } j\}$. This domain is a proper subset of the domain of x in some of the problems we consider.

2.2.2. *Convex Quadratic Programming Reformulation (CQPR)*

The second class of reformulation methods we consider is convex quadratic reformulation. The idea is to replace the quadratic form $f(x) = x'Qx$ with the function $g(x) = x'\Lambda x + \lambda x$, such that $f(x) = g(x)$ for all $x \in \{0,1\}^n$ and Λ is negative semi-definite.

A simple way to do this is to add $\sum_i \lambda_i(x_i - x_i^2)$ to the original objective, where $\lambda > 0$. For λ sufficiently large, $\Lambda = Q - \mathrm{diag}(\lambda)$ is negative semi-definite. This stems from the early work of Hammer and Rubin [25]. Recently, this framework was expanded by Billionnet et al. [8], who use the solution of a semi-definite program to obtain a convex reformulation with a continuous bound that is at least as tight as that of [25].

In our study we rely on the automatic convexification by CPLEX® 10 because our goal is to see if we can solve these biology problems with a standard solver, rather than a structure-exploiting algorithm. CPLEX sets λ such that Λ is diagonally-dominant, which is sufficient for Λ to be negative semi-definite. In particular, CPLEX sets λ to satisfy

$$\lambda_i \geq \sum_{j \neq i} |Q_{ij}|, \ \ \forall i. \tag{5}$$

(Recall $Q_{ii} = 0$ in all of our formulations, using cx to include the square since $x_i^2 = x_i$.)

2.3. *Numerics*

Our data are primarily from the PDB, a rich database of protein structures and sequences. The protein identifier is a 4-character code, such as 1abo. The problem formulations were modeled using AMPL® and solved with CPLEX with a one hour time limit. The data and program files are available at this article's supplement site.

For comparative purposes, we executed our tests under both MS Windows XP® (henceforth, called Windows) and Unix computing environments. Our PC is an IBM Thinkpad®, equipped with a 3.0 GHz processor and 2 GB RAM (+ 4 GB of virtual memory). Our Unix system is a Sun V440, equipped with four 1.6 GHz processors and 16 GB RAM, running Solaris 10.

Since we ran our tests with the same version of CPLEX on both platforms, the main difference, other than processor speed, is that our version of Windows has a memory limitation of 2 GB per process due to the 32-bit addressing. This memory restriction prevented us from solving some problems under Windows due to insufficient memory for AMPL/CPLEX to setup and solve the root linear programming relaxation. We therefore focus on our test results under Unix, while indicating the results under Windows.

For each of our problems we had three start conditions: cold, warm, and hot. For the cold start we simply passed our formulation to CPLEX, while for both the warm and hot starts we provided an incumbent solution. The warm start gives a simple feasible solution that is problem-dependent, but does not exploit any information from a particular instance. The hot start sets the variables equal to a solution obtained from an instance-dependent heuristic. The question addressed with the hot start is, "Can a general-purpose solver find a better solution, or confirm the optimality of the heuristic solution, within some reasonable time?" We do not report the time to compute the hot start because that does not affect the answer to our question.

3. QBP Models and Computational Results

In this section we consider four problems in computational molecular biology: Multiple Sequence Alignment (MSA), Lattice Protein Folding (LPF), Contact Map Overlap (CMO), and Rotamer Assignment (RoA). We present a QBP formulation for each problem, and numerical results for the linearizations and the convex quadratic reformulation.

3.1. *Multiple Sequence Alignment*

Two fundamental biological sequences are taken from the alphabet of nucleic acids, {`a,c,g,t`}, and from the alphabet of amino acids, {`A,R,N,D,C,Q,E,G,H,I,L,K,M,F,P,S,T,W,Y,V`}. The former are segments of DNA (or RNA if `t` is replaced by `u`). The latter are segments of proteins.

The Multiple Sequence Alignment (MSA) Problem is to seek similarities among a given set of sequences from the same alphabet. This might be to:

- understand life through evolution;
- identify families of proteins to infer structure or function from sequence;
- diagnose disease;
- retrieve similar sequences from databases.

An early application of MSA that illustrates its importance is given by Riordan et al. [33], who discovered the Cystic Fibrosis Transmembrane Regulator gene and its connection to Cystic Fibrosis. Even before then, algorithms for MSA [31,35] were developed for the biologists who used computational methods for their research. A good introduction to dynamic programming methods is by Fuellen [17].

One key to defining an objective is the notion of a *gap*. This is a sequence of insertions or deletions (called *indels*) that occur during evolution. For example, if one applies a simple Hamming distance to the sequences, `acacta` and `tacact`, they are six characters apart. However, inserting gaps we obtain the *extended sequences*,

```
 -acacta
  |||||
 tacact-
```

This has a Hamming distance of only two. The middle sequence, `acact`, is the same in the alignment. (See Doolittle [16] for a clear account of the biology and references to early works.)

Two sequences can be optimally aligned by dynamic programming, where "optimal" is one that maximizes an objective that has two parts:

(1) a *scoring function*, given in the form of an $m \times m$ matrix S, where m is the size of the alphabet. The value of S_{ij} measures a propensity for the i^{th} alphabet-character in one sequence to align with the j^{th} alphabet-character in some position of the other sequence.
Example: Let $s =$ `agt` and $t =$ `gtac`. In the alignment of the first character of s with the first character of t, the score is $S_{\texttt{ag}}$, which is the propensity for `a` to be aligned with `g`.

(2) a *gap penalty function*, expressed in two parts: a "fixed cost" of beginning a gap, denoted G_{open}, and a cost to "extend" the gap, denoted G_{ext}.
Example: Let $s =$ `agt` and $t =$ `gtac`. One alignment is to put a gap at the end of the first sequence:

```
agt-
gtac
```

Figure 2 shows three different alignments for the two nucleic acid sequences, `agt` and `gtac`. Suppose the scoring matrix is

$$S = \begin{array}{c} \begin{array}{cccc} \texttt{a} & \texttt{c} & \texttt{g} & \texttt{t} \end{array} \\ \left[\begin{array}{rrrr} 2 & -1 & -2 & 0 \\ -1 & 2 & 0 & -2 \\ -2 & 0 & 2 & -1 \\ 0 & -2 & -1 & 2 \end{array}\right] \end{array} \begin{array}{l} \\ \texttt{a} \\ \texttt{c} \\ \texttt{g} \\ \texttt{t} \end{array}$$

Then, the scores (without gap penalties) are 4, 0, and -3, respectively.

```
agt--      -a-gt      agt-
 ||
-gtac      gtac-      gtac
```

Fig. 2. Three Alignments for Two Sequences

The total objective function for the 2-sequence alignment problem has the form

$$\sum_{i,j} S_{s_i t_j} - G_{\text{open}}(N_s + N_t) - G_{\text{ext}}(M_s + M_t),$$

where the sum is over aligned characters, s_i from sequence s with t_j from sequence t. The number of gaps opened is N_s in sequence s and N_t in sequence t; the number of gap characters (`-`) is M_s in sequence s and M_t in sequence t. In the example of Figure 2, if G_{open}=2 and G_{ext}=1, the gap penalties are 7, 9, and 3, respectively.

There are different scoring methods, but this is the common one, which we shall use. One way to evaluate an MSA is by summing pairwise scores. Figure 3 shows an example. Using the same scoring matrix as above, the sum-of-pairs score is shown for each column. For example, column 1 has $3S_{\texttt{aa}} + 3S_{\texttt{ac}} = 3$. The sum of pairwise scores for column 2 is zero because we do not score the gaps by columns; they are penalized for each sequence (row of alignment). The total objective value is $31 - 28 = 3$.

		Gap penalty
	`a-gagt-act---`	8
	`aagtat--at---`	7
	`a--tataa----t`	8
	`c-gta--actcct`	5
score:	`3066020606002`	28 = Total
	Total = 31	

Fig. 3. A Multiple Alignment of Four Sequences

3.1.1. *QBP Model and Test Data*

The QBP model is as follows. Let

$$x_{i\ell k} = \begin{cases} 1 & \text{if } i^{\text{th}}\text{character of sequence } \ell \text{ is} \\ & \text{assigned to column } k \\ 0 & \text{otherwise} \end{cases}$$

$$y_{\ell k} = \begin{cases} 1 \text{ if sequence } \ell \text{ has a gap in column } k; \\ 0 \text{ otherwise.} \end{cases}$$

$$z_{\ell k} = \begin{cases} 1 \text{ if sequence } \ell \text{ opens a gap in column } k; \\ 0 \text{ otherwise.} \end{cases}$$

Then, the QBP for the MSA of sequences $s^1, \ldots, s^m$ is given by:

$$
\begin{aligned}
\max\ \textstyle\sum_k \sum_{\ell'>\ell} \sum_{i,j\le k} S_{s_i^\ell s_j^{\ell'}}\, x_{i\ell k} x_{j\ell' k}& \\
-\textstyle\sum_{k,\ell} (G_{\text{open}} z_{\ell k} + G_{\text{ext}} y_{\ell k}) :& \\
\textstyle\sum_{k\ge i} x_{i\ell k} &= 1 \quad \forall\, i,\ell \\
\textstyle\sum_{k'>k} x_{i+1,\ell k'} &\ge x_{i\ell k}\ \forall\, i < L_\ell, \ell, k \ge i \\
\textstyle\sum_{k'=i-1}^{k-1} x_{i-1,\ell k'} &\ge x_{i\ell k}\ \forall\, 2 \le i \le L_\ell, \ell, k \ge i \\
y_{\ell k} + \textstyle\sum_{i\le k} x_{i\ell k} &= 1 \quad \forall\, k,\ell \\
y_{\ell k} - y_{\ell k-1} - z_{\ell k} &\le 0 \quad \forall\, k,\ell\ (y_{\ell 0} = 0) \\
x \in \{0,1\},\ 0 \le y, z \le 1.&
\end{aligned}
$$

where L_ℓ is the length of sequence l. We do not explicitly require y, z to be binary; that is implied by x binary for basic solutions. (An interior solution yields fractional values of z if G_{open}=0.)

The first sum in the objective is the sum-of-pairs score, from which we subtract the total gap penalty. The index condition $\ell' > \ell$ is to avoid double counting; the conditions $i, j \le k$ reflect the fact that we cannot assign a character to a column number that is less than the character's position. For example, we cannot assign character 5 to column 4.

The first constraint requires that each character, i, in each sequence, ℓ, be assigned to some column, k ($\ge i$). The next two constraints preserve the character order of each sequence — if the i^{th} character of string ℓ is assigned to column k ($x_{i\ell k} = 1$), its successor ($i + 1$) must be assigned to a subsequent column ($k' > k$), while its predecessor must be assigned to a previous column ($k' < k$). The fourth constraint requires that, for each column of each sequence, either a character is assigned or it is in a gap. Finally, the last constraint requires that a gap is opened (i.e., $z_{\ell k}$ is forced to 1) if it changed from no gap assignment ($y_{\ell\, k-1} = 0$) to a gap assignment ($y_{\ell k} = 1$).

The number of binary variables is the size of the domain of x:

$$
\begin{aligned}
\big|\{(i,\ell,k) : k \ge i\}\big| &= \sum_{\ell=1}^{m} \sum_{i=1}^{L_\ell} (N - i + 1) \\
&= \sum_{\ell=1}^{m} \big((N+1)L_\ell - \tfrac{1}{2} L_\ell (L_\ell + 1)\big) \\
&= N \sum_{\ell=1}^{m} L_\ell - \tfrac{1}{2} \sum_{\ell=1}^{m} L_\ell (L_\ell - 1), \qquad (6)
\end{aligned}
$$

where N is the number of columns in the alignment.

We do not know in advance how many columns will be in the alignment; that depends on gap lengths. Theoretically, there could be $\sum_{\ell=1}^{m} L_\ell$ columns (aligning no letter), but our implementation does the following. Define $L_{\max} = \max_\ell L_\ell$, so the minimum number of columns is $L_{\max}$, and we set $N_{\max} = \min\{2L_{\max}, \sum_\ell L_\ell\}$. After solving the QBP, we remove trailing gaps that have no character assignment. We do this with auxiliary variables, called "excess," e_k, for $k = L_{\max} + 1, \ldots, N_{\max}$. The column assignment constraint is modified as follows:

$$
\begin{aligned}
y_{\ell k} + \textstyle\sum_{i\le k} x_{i\ell k} &= 1 \quad \text{for } \ell,\ k = 1, \ldots, L_{\max} \\
y_{\ell k} + \textstyle\sum_{i\le k} x_{i\ell k} + e_k &= 1 \quad \text{for } \ell, k = L_{\max} + 1, \ldots, N_{\max} \\
e_k \in \{0,1\}.&
\end{aligned}
$$

The only way to have $e_k = 1$ is to have no character assignment for *any* string ($x_{i\ell k} = 0$ for all i, ℓ) and no gap assignment ($y_{\ell k} = 0$ for all ℓ). Thus, the entire column is the gap character, which contributes zero to the score. If $G_{\text{ext}} > 0$, nothing more is needed, but if $G_{\text{ext}} = 0$, we want to be sure that the excess is composed of trailing columns, which can then be cut to produce the final alignment. We thus add the constraint:

$$e_{k+1} \ge e_k \text{ for } k = L_{\max} + 1, \ldots, N_{\max} - 1.$$

The total number of binary variables in QBP is the size of the domain of x, given by equation (6), plus $N_{\max} - L_{\max}$ excess variables. In addition, it has $2mN_{\max}$ linear variables (y, z domain size). For example, the QBP to align three sequences of lengths 20, 50, and 100 has 100 excess variables plus $200(170) - \frac{1}{2}(20 \times 19 + 50 \times 49 + 100 \times 99)$ x-variables, for a total of 27,735 binary variables. This is a very large QBP despite the biology problem being fairly small.

Our implementation economizes on the domain of the quadratic form by excluding terms with zero score. This happens for amino acid sequence alignments, using a standard scoring matrix like the one we used: BLOSUM50 [13].

We ran three problems, called small, medium, and large, with the characteristics shown in Table 2. They were obtained from an example in the MATLAB® *Bioinformatics User's Guide,* starting with human and mouse *open reading frames*. The actual sequence lengths were greater than 100, so we truncated them to derive two subsequences. These were copied and modified to create the sequences used.

Table 1

Problem	m	Lengths	# Binary Variables
small	3	13, 16, 17	1,247
medium	4	52, 63, 82, 83	36,560
large	6	41, 58, 59, 60, 78, 88	54,887

Characteristics of MSA Problems

Table 3 shows the numbers of auxiliary variables and constraints required by each linearization. It is interesting to note how compact Glover's formulation is compared to the other linearizations. The Standard linearization requires $w_{i\ell j\ell' k}$ to replace the product $x_{i\ell k}x_{j\ell' k}$. The domain is determined as follows. The domain of the summation is

$$\mathcal{S} = \{(i,\ell,j,\ell',k) : \ell' > \ell,\, i \le k, j \le k\}.$$

We can, however, discard indexes for which the score is zero, and we can partition the nonzero scores to add only upper or lower bounds, not both, according to the sign of the score:

$$\mathcal{S}^- = \{(i,\ell,j,\ell',k) \in \mathcal{S} : S_{s_i^{\ell'}\, s_j^{\ell}} < 0\}$$
$$\mathcal{S}^+ = \{(i,\ell,j,\ell',k) \in \mathcal{S} : S_{s_i^{\ell'}\, s_j^{\ell}} > 0\}.$$

The domain of w is $\mathcal{S}^- \cup \mathcal{S}^+$, and the auxiliary constraints are

$$w_{i\ell j\ell' k} \ge x_{i\ell k} + x_{i\ell k} - 1 \text{ for } (i,\ell,j,\ell',k) \in \mathcal{S}^-$$
$$w_{i\ell j\ell' k} \le x_{i\ell k},\; w_{i\ell j\ell' k} \le x_{i\ell k} \text{ for } (i,\ell,j,\ell',k) \in \mathcal{S}^+$$

(not counting the non-negativity restrictions over $\mathcal{S}^-$). The sizes of these domains are given in Table 3 for the three test problems. For example, in the Standard linearization the small problem requires $|\mathcal{S}^-| + |\mathcal{S}^+| = 13,021$ auxiliary variables and $|\mathcal{S}^-| + 2|\mathcal{S}^+| = 17,820$ auxiliary constraints.

Our partial RLT formulation was constructed by multiplying the character-assignment constraints by each character-assignment variable $x_{i'\ell' k'}$ for all (i', ℓ, k') with $\ell' > \ell$ and $k' \ge i'$:

$$\sum_{k \ge i} x_{i\ell k} = 1 \;\forall\, i, \ell$$

implies

$$\sum_{k \ge i} x_{i\ell k}\, x_{i'\ell' k'} = x_{i'\ell' k'} \;\forall\, i, \ell, i', \ell' > \ell, k' \ge i'.$$

Table 2

Problem		Standard	RLT	Glover
small	Variables	13,021	502,632	1,230
	Constraints	17,820	36,286	1,230
medium	Variables	3,155,499	495,656,781	36,477
	Constraints	3,803,991	7,723,834	36,477
large	Variables	7,252,773	1,242,683,653	54,799
	Constraints	8,485,328	17,430,037	54,799

Variables and Constraints Added for Linearizations of MSA

Upon substituting $w_{i\ell k i'\ell' k'}$ for $x_{i\ell k}\, x_{i'\ell' k'}$ we obtain the RLT constraints:

$$\sum_{k \ge i} w_{i\ell k i'\ell' k'} = x_{i'\ell' k'} \;\forall\, i, \ell, i', \ell' > \ell, k' \ge i',$$

which were added to the Standard linearization. Note that the auxiliary variables w have all six indexes (as opposed to the five indices of the Standard linearization), and thus the domain of w is limited only by the domain of the binary variables and the domain of summation:

$$\mathrm{dom}_{\mathrm{RLT}}(w) = \{(i,\ell,k,i',\ell',k') : \ell' > \ell, i \le k, i' \le k'\}.$$

The number of auxiliary variables is

$$|\mathrm{dom}_{\mathrm{RLT}}(w)| = \sum_{\ell=1}^{m-1} T_\ell \sum_{\ell'=\ell+1}^{m} T_{\ell'}, \tag{7}$$

where $T_\ell = L_\ell\big(N_{\max} - \frac{1}{2}(L_\ell - 1)\big)$. The number of auxiliary constraints equals the number in the Standard linearization plus the number of equations, given by

$$\#\mathrm{Eqns} = \sum_{\ell=1}^{m-1} L_\ell \sum_{\ell'=\ell+1}^{m} T_{\ell'}. \tag{8}$$

(See this article's supplement site for the derivation of (7) and (8).)

Glover's linearization requires an auxiliary variable, $w_{i\ell k}$, to represent the product, $x_{i\ell k} \sum_{j \le k, \ell' > \ell} S_{s_i^{\ell'}\, s_j^{\ell}} x_{j\ell' k}$ (for $i \le k$). The size of this domain cannot exceed the number of x-variables; it is less if the sum is identically zero for some (i, ℓ, k). Such is the case in our problems upon comparing the number shown in Table 3 with the associated number of binary variables shown in Table 2.

3.1.2. *Warm and Hot Starts*

The warm start aligns the sequences with just one gap at the end for sequences whose length is less than the maximum sequence length. In the model, this means for all sequences (ℓ):

$$\begin{aligned} x_{k\ell k} &= 1 \text{ for } k = 1, \ldots, L_\ell \\ y_{\ell k} &= 1 \text{ for } k = L_\ell + 1, \ldots, L_{\max} \\ z_{\ell\, L_\ell+1} &= 1 \text{ if } L_\ell < L_{\max} \\ e_k &= 1 \text{ for } k = L_{\max} + 1, \ldots, N_{\max} \end{aligned}$$

All other variables equal zero. This is illustrated by the diagram in Figure 4.

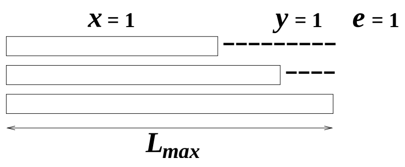


Fig. 4. MSA Warm Start

The hot start takes the solution from the Bioinformatics Toolbox in MATLAB, which is the same algorithm used in practice. All MATLAB solutions were obtained in less than one second, and the m-files are posted at this article's supplement site.

3.1.3. *Numerical Results*

Our numerical results are shown in Table 4, where we report the CPU times and best objective values found for each case. A case is defined by three specifications:

Problem instance:	small, medium, or large.
Linearization strategy:	Standard, RLT, Glover, or None.
Start condition:	cold, warm, or hot.

We found that CPLEX terminated before confirming an optimal solution due to reaching our time limit (1 hr) or our memory limit (16 GB).

Glover's linearization clearly outperformed the other three reformulation methods for the small problem. Interestingly, it found the best solution overall from a cold start. Further, no reformulation method made any progress with the medium and large problems — they did not find a feasible solution from a cold start, and they did not improve upon the warm and hot start values. In addition, the RLT linearization aborted due to memory errors for all start conditions. These memory errors occurred before CPLEX was able to process the initial solution, which is why we put the value of "none" in the Value column. Finally, while CQPR was not quite as effective as Glover's linearization, it still performed quite well. Using CQPR has the advantage that it can be passed directly to CPLEX as a quadratic program, without the use of a linearization method. We experienced similar results under Windows, except that CQPR ran out of memory, rather than time. Also, Glover's linearization obtained a score of only 178 from a cold start. It improved the hot start for the small problem, but not as much; it gave an alignment with a score of 272, instead of 279.

The performance by Glover's linearization is encouraging because it obtained a better solution than the heuristic used by MATLAB. This heuristic is the same as is currently used in practice, and most scientists believe they are getting the best alignment possible. The improved alignment for the small problem is shown in Figure 5. Note that both alignments have the same core conservation region but differ in where they place the gaps and their lengths.

Table 5 summarizes the warm and hot start values, along with the overall best objective values found for each problem under Unix.

Table 4

Problem	Warm Start	Hot Start	Best	Method
small	14	270	280	Glover
medium	179	1,930	1,930	Hot start
large	2,856	4,352	4,352	Hot start

Solution Summary for MSA

3.2. *Lattice Protein Folding Problem*

The *protein folding problem* is among the most celebrated problems in molecular biology that have used global optimization models for energy minimization. Dill [14] introduced a biological simplification of the full model that considers six forms of energy interactions. He kept only the effect of hydrophobicity, which is dominant in globular proteins. (A full biological explanation, with benefit of hindsight, is given by Dill et al. [15].) Hart and Istrail [26,27] approached Dill's lattice model as a combinatorial optimization problem, and they developed a foundation for Lattice Protein Folding (LPF) approximation algorithms. We formulate that problem here as a QBP.

The LPF Problem is to determine an assignment of a sequence of amino acids to grid points, in 2 or 3 dimensions, so as to maximize the number of

Table 3

		Standard		Linearization RLT		Glover		No linearization (CQPR)	
Problem	Start	CPU	Value	CPU	Value	CPU	Value	CPU	Value
small	cold	T	96	T	245	T	280	T	239
	warm	T	156	T	17	T	255	T	239
	hot	T	270†	T	270†	T	279	T	270†
medium	cold	T	none	M	none	T	none	T	none
	warm	T	179†	M	none	T	179†	T	179†
	hot	T	1,930†	M	none	T	1,930†	T	1,930†
large	cold	T	none	M	none	T	none	T	none
	warm	T	2,856†	M	none	T	2,856†	T	2,856†
	hot	T	4,352†	M	none	T	4,352†	T	4,352†

†Same as initial value; M out of memory (16 GB); T out of time (1 hr).

Unix CPU Times and Best Values Found for QBP of MSA

```
--CACGTAACATCT-C--
ACGACGTAACATCTTCT-
A-AACGTAACATCT-CGC
```

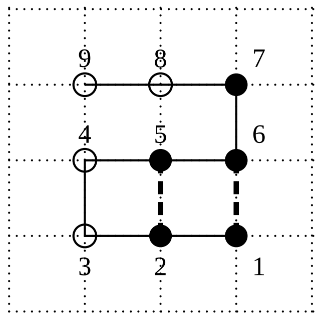
(a) Hot Solution

```
-C--ACGTAACATCT---C-
ACG-ACGTAACATCTT--CT
A--AACGTAACATCT-CGC-
```

(b) Improved Alignment

Fig. 5. Improved Alignment of small Problem

hydrophobic acids that become neighbors. Figure 6 gives an example. Each amino acid in the given sequence is mapped into a binary sequence H such that $H_i = 1$ means the i^{th} acid is hydrophobic. Two hydrophobic neighbors are created, shown by the dark, dashed lines connecting acids 1-6 and 2-5.



H=110011100

Fig. 6. Fold for HP Example (● is hydrophobic; ○ is hydrophyllic)

3.2.1. *QBP Model and Test Data*

The QBP model is as follows. Let

$$x_{ip} = \begin{cases} 1 \text{ if acid } i \text{ is assigned to point } p; \\ 0 \text{ otherwise.} \end{cases}$$

Let $\mathcal{H}$ denote the set of hydrophobic acids (i.e., $\mathcal{H} = \{i : H_i = 1\}$). Further, let $\mathcal{N}(p)$ denote the neighbors of point p (excluding p). For a square grid, we use the Manhattan distance to define neighbors:

$$\mathcal{N}(p) = \{q : |X_p - X_q| + |Y_p - Y_q| = 1\},$$

where (X_p, Y_p) denotes the coordinates of point p.

Then, the QBP for the LPF Problem for a sequence of n amino acids is:

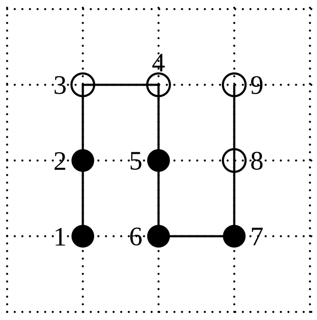
$$\begin{array}{lll} \max \sum_p \sum_{q\in\mathcal{N}(p)} \sum_{i,j\in\mathcal{H}:j>i+1} x_{ip}x_{jq} & & \\ \sum_p x_{ip} & = 1 & \forall i \\ \sum_i x_{ip} & \leq 1 & \forall p \\ \sum_{q\in\mathcal{N}(p)} x_{i+1,q} & \geq x_{ip} & \forall p, i < n \\ \sum_{q\in\mathcal{N}(p)} x_{i-1,q} & \geq x_{ip} & \forall p, i > 1 \\ x \in \{0,1\}. & & \end{array}$$

The objective scores a 1 when hydrophobic acids i and j are assigned to neighboring points p and q ($x_{ip} =$

$x_{jq} = 1$). The added condition $j > i + 1$ is to avoid counting those that are already adjacent in the given sequence.

The first constraint requires that each acid be assigned to exactly one point; the second requires that at most one point is assigned to an acid. The last two constraints require backbone neighbors to remain neighbors in the fold — that is, if acid i is assigned to point p ($x_{ip} = 1$), acid $i \pm 1$ must be assigned to some neighbor ($x_{i\pm1,q} = 1$ for some $q \in \mathcal{N}(p)$).

We must add *symmetry exclusion* constraints [6] — any rigid motion, like translation and rotation, appears as an alternative optimum since the variables have different values, however, such groups are the same fold. For example, the fold in Figure 6 can be rotated, as shown in Figure 7. We rotated it 90^o clockwise about the middle acid (#5).



H=110011100

Fig. 7. Rotation of Fold in Figure 6

Without symmetry exclusion, branch-and-bound would take unnecessary searches for what would appear to be a potentially better subtree. Here are (global) *symmetry exclusion constraints*:

Fix middle acid at mid-point of grid:

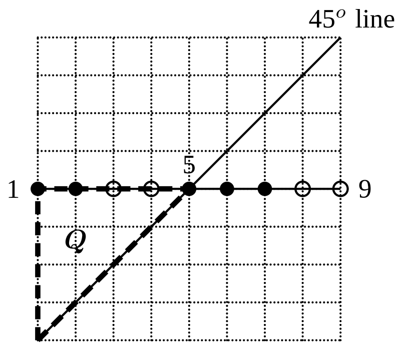
$$x_{mp_m} = 1$$

Restrict acid 1 to the upper half of quadrant III:

$$\textstyle\sum_{p \in Q} x_{1p} = 1 \quad .$$

Fixing the middle acid prevents translation, and restricting acid 1 to the upper half of quadrant III prevents equivalent folds by some rotations and reflection about the 45^o line. Figure 8 shows the grid for a sequence of length 9, with the example of Figure 6 assigned to the horizontal axis. (Point # 5 is the middle acid, which is fixed at the middle point of the grid.)

In our preliminary computations, this model, with all grid points defined for the domain of each acid, could



H=110011100

Fig. 8. Excluding Some Symmetries

not be solved (with confirmed optimality), even for small proteins. Note that the number of points in the grid is n^2, so the domain size of the assignment variables is n^3. Thus, for a modest-size protein of only 100 acids, we have 1,000,000 binary variables! We therefore consider making the set of admissible points depend upon the acid number. Fixing the middle acid makes its domain just one, namely the middle point; the predecessor and successor acids are therefore restricted to only four points. As the acid number is farther from the middle acid, its number of possible point assignments is greater. The condition that limits the number of points that are possible assignments for acid i is that the distance from the middle acid (m) must be within its distance along backbone:

$$\mathcal{P}(i) = \{p : |X_p - X_m| + |Y_p - Y_m| \le |m - i|\}. \tag{9}$$

With only this reduction, the number of binary variables for an n-acid protein is about $\left(\frac{n}{2}\right)^2$. Thus, a 100-acid protein has about 2,500 binary variables, rather than 1,000,000 in the original model.

Further reductions are possible. For example, we can exclude point p_m from each $P(i)$ for $i \neq m$. More significantly, points in $\mathcal{P}(i)$ must be within 1 of points in $\mathcal{P}(i \pm 1)$ $(1 < i < n)$, which we could take into account by fanning out from m. For example, Figure 9 shows points in $\mathcal{P}(m \pm 3)$, as defined by (9), that can be removed.

The general case is that $\mathcal{P}(m \pm i)$ contains alternate diamonds around the middle point. Define the i^{th} diamond:

$$\mathcal{D}(i) = \{p : |X_p - X_m| + |Y_p - Y_m| = |m - i|\}. \tag{10}$$

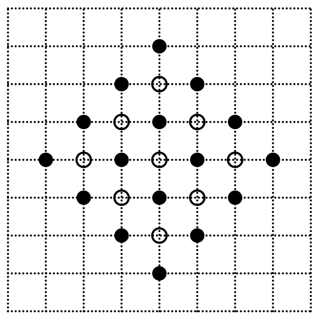


Fig. 9. $\mathcal{P}(m \pm 3)$ — Points Not Filled Can Be Removed

Let $\mathcal{P}(m), \mathcal{P}(m \pm 1), \mathcal{P}(m \pm 2)$ be given by (9). Then, fan out from m as:

$$\mathcal{P}(m \pm i) = D(i) \cup \mathcal{P}(m \pm (i-2)) \text{ for } i = 3, \dots, m. \quad (11)$$

The reduced LPF model that we implemented is thus:

$$\begin{array}{ll} \max \sum_{i,j \in \mathcal{H}: j > i+1} \sum_{p \in \mathcal{P}(i)} \sum_{q \in \mathcal{N}(p) \cap \mathcal{P}(j)} x_{ip} x_{jq} & \\ \sum_{p \in \mathcal{P}(i)} x_{ip} & = 1 \quad \forall\, i \\ \sum_{i: p \in \mathcal{P}(i)} x_{ip} & \le 1 \quad \forall\, p \\ \sum_{q \in \mathcal{N}(p) \cap \mathcal{P}(i+1)} x_{i+1,q} & \ge x_{ip} \ \forall\, i < n, p \in \mathcal{P}(i) \\ \sum_{q \in \mathcal{N}(p) \cap \mathcal{P}(i-1)} x_{i-1,q} & \ge x_{ip} \ \forall\, i > 1, p \in \mathcal{P}(i) \\ & x \in \{0,1\}. \end{array}$$

We do not need symmetry exclusion constraints because they are represented by the construction of the point domains, with the added restriction on the assignment of the first residue:

$$\mathcal{P}(1) \leftarrow \mathcal{P}(1) \cap \big\{p : X_p \le X_m,\ Y_p \le Y_m,$$
$$(X_m - X_0)(Y_p - Y_0) \ge (Y_m - Y_0)(X_p - X_0)\big\},$$

where (X_0, Y_0) is the origin. (These added constraints correspond to being in Q, shown in Figure 8.) The LPF and MSA problems highlight an important modeling practice:

Restrict domains by model logic, not by constraints on the variables.

We ran four problems, using proteins from the PDB, with the characteristics shown in Table 6. Table 7 shows the number of auxiliary variables and constraints required by each linearization. We omit the details of the Standard and Glover linearizations; our implementations are described in the AMPL source codes, which are included at this article's supplement site. Our partial RLT formulation was constructed by multiplying the acid assignment constraints by each hydrophobic acid x_{jq} for $j \in \mathcal{H}, q \in \mathcal{P}(j)$ with $i < j$:

$$\sum_{p \in \mathcal{P}(i)} x_{ip} = 1\, \forall\, i \Rightarrow$$
$$\sum_{p \in \mathcal{P}(i)} x_{ip} x_{jq} = x_{jq}\, \forall\, i, j \in \mathcal{H}, q \in \mathcal{P}(j), i < j \Rightarrow$$
$$\sum_{p \in \mathcal{P}(i)} w_{ipjq} = x_{jq}\, \forall\, i, j \in \mathcal{H}, q \in \mathcal{P}(j), i < j.$$

Table 5

Problem	# Acids n	# Hydrophobic $\lvert\mathcal{H}\rvert$	# Points $\lvert \cup \mathcal{P}(i) \rvert$	# Binary Variables
1abo	72	20	2,663	32,618
1bbz	69	17	2,515	28,848
1kwa	88	31	3,959	59,036
1n5z	106	29	5,723	102,508

Characteristics of LPF Problems

Table 6

Protein		Standard	RLT	Glover
1abo	Variables	76,142	130,440,588	9,175
	Constraints	152,284	442,380	9,175
1bbz	Variables	52,008	45,575,468	7,376
	Constraints	104,016	210,648	7,376
1kwa	Variables	282,904	6.09×10^8	20,611
	Constraints	565,808	1,535,709	20,611
1n5z	Variables	307,772	1.14×10^9	23,905
	Constraints	615,544	1,796,760	23,905

Variables and Constraints Added for Linearizations of LPF

3.2.2. *Warm and Hot Starts*

The warm start assigns the acids along the horizontal mid-line to the midpoint, then folds back, like a horseshoe. In the model, this means

$$\begin{array}{lll} \text{for } i = 1, \dots, \frac{n}{2}: & x_{ip} = 1 & \text{for } p : X_p = i,\ Y_p = Y_m \\ \text{for } i = \frac{n}{2}+1, \dots n: & x_{ip} = 1 & \text{for } p : X_p = n - i + 1, \\ & & Y_p = Y_m + 1. \end{array}$$

(Round $\frac{n}{2}$ when n is odd.) This is illustrated in Figure 10.

The hot start uses a new heuristic by Rego et al. [32]. (We include the solution files provided by César Rego and MATLAB scripts that convert them to our AMPL hot start files at this article's supplement site.)

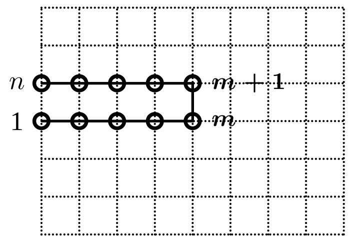


Fig. 10. LPF Warm Start

3.2.3. *Computational Results*

Our numerical results are reported in Table 8, and we summarize our findings in Table 9. The results indicate that CPLEX had difficulty with all reformulations of the LPF Problem — it did not find a feasible solution to any cold start, and it did not improve upon the warm and hot start values. These results are not particularly surprising given the enormous number of binary variables (c.f., Table 6). Further, it is likely that the hot start solutions are optimal since the heuristic [32] used to determine the values is known to produce the best solutions in the literature. Our results were similar under Windows, except for proteins 1kwa and 1n5z, for which the Standard linearization reached the memory limit and produced no feasible solution, even with warm and hot starts.

Table 8

Problem	Warm Start	Hot Start	Best	Method
1abo	4	16	16	Hot start
1bbz	0	14	14	Hot start
1kwa	3	26	26	Hot start
1n5z	5	23	23	Hot start

Solution Summary for LPF

3.3. *Protein Comparison by Contact Maps*

The *contact map* of a protein is a graph, $G = [V, E]$, where V represents its sequence of amino acids that bonded to form the protein. We presume we know its native state, and we define an edge between two nodes if their distance is within some threshold (our data used 4.5Å).

Given the contact maps of two proteins, G_1, G_2, we define their similarity to be the largest subgraphs that are isomorphic. Here, "largest" is measured by the number of edges. The Contact Map Overlap (CMO) Problem is to find the largest isomorphic subgraphs, constrained to have the same node ordering (to preserve the backbone order). Measuring protein similarity is a longstanding problem in molecular biology. Sequence alignment is one measure; CMO is another.

The MILP approach was introduced by Lancia et al. [30]. They developed deep cuts by exploiting the problem structure, particularly with respect to constraining the same backbone order. Recently, Xie and Sahinidis [37] presented a reduction-based, exact algorithm.

The preservation of the backbone order is equivalent to not having any *crossing* — that is, associations $i \leftrightarrow j$ and $\ell \leftrightarrow k$ such that $i < \ell$ but $j > k$. The situation is depicted in Figure 11. Our model must disallow any crossing, but it is sufficient to eliminate all 2-crossings.

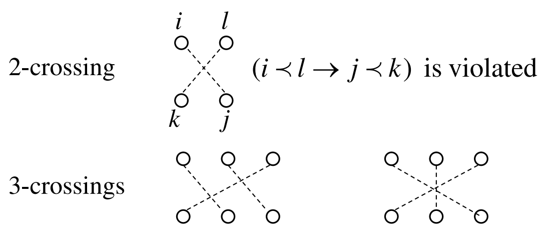


Fig. 11. Crossings to be Excluded

3.3.1. *QBP Model and Test Data*

The QBP model is as follows. Let

$$x_{ij} = \begin{cases} 1 \text{ if node } i \in V_1 \text{ is associated with node } j \in V_2; \\ 0 \text{ otherwise.} \end{cases}$$

Then, the QBP for the CMO Problem is:

$$\max \sum_{\substack{(i,k)\in E_1 \\ (j,\ell)\in E_2}} x_{ij}x_{k\ell}$$

$$\textstyle\sum_j x_{ij} \le 1 \,\forall\, i, \ \sum_i x_{ij} \le 1 \,\forall\, j$$

$$x_{ij} + x_{k\ell} \le 1 \text{ for } 1 \le i < k < |V_1|, \ 1 \le \ell < j < |V_2|$$

$$x \in \{0, 1\}.$$

The objective scores a 1 when edge $(i, k) \in E_1$ is associated with edge $(j, \ell) \in E_2$. That happens when the endpoint nodes are associated. The first constraint limits a node in V_1 to be associated with at most one node in V_2. The second is likewise. The conflict constraints preserve the backbone ordering by disallowing any 2-crossing.

Table 7

		Standard		Linearization RLT		Glover		No linearization (CQPR)	
Protein	Start	CPU	Value	CPU	Value	CPU	Value	CPU	Value
1abo	cold	T	none	M	none	T	none	T	none
	warm	T	$4^\dagger$	M	none	T	$4^\dagger$	T	$4^\dagger$
	hot	T	$16^\dagger$	M	none	T	$16^\dagger$	T	$16^\dagger$
1bbz	cold	T	none	M	none	T	none	T	none
	warm	T	$0^\dagger$	M	none	T	$0^\dagger$	T	$0^\dagger$
	hot	T	$14^\dagger$	M	none	T	$14^\dagger$	T	$14^\dagger$
1kwa	cold	T	none	M	none	T	none	T	none
	warm	T	$3^\dagger$	M	none	T	$3^\dagger$	T	$3^\dagger$
	hot	T	$26^\dagger$	M	none	T	$26^\dagger$	T	$26^\dagger$
1n5z	cold	T	none	M	none	T	none	T	none
	warm	T	$5^\dagger$	M	none	T	$5^\dagger$	T	$5^\dagger$
	hot	T	$23^\dagger$	M	none	T	$23^\dagger$	T	$23^\dagger$

†Same as initial value; M out of memory (16 GB); T out of time (1 hr).

Unix CPU Times and Best Values Found for QBP of LPF

Figure 12 shows an example with the optimal solution. The dashed lines indicate the seven pairs of nodes that are associated. The darkened edges are associated, for a total score of 5.

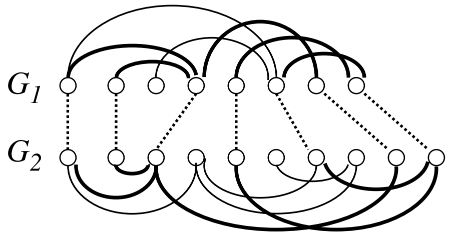


Fig. 12. Example of Two Contact Maps with Associated Nodes

The 2-crossing conflict constraints are known to be weak, compared to easily-derived stronger systems of inequalities (e.g., see [30]), but our experiments use this model with just the degree-2 inequalities.

We ran four problems, using proteins from the PDB, with the characteristics shown in Table 10. We include comparing 1f22 with itself, which has 55 edges. Table 11 shows the number of auxiliary variables and constraints needed by each linearization.

We once again omit the details of the Standard and Glover linearizations, and refer the reader to the AMPL models included at this article's supplement site. Our partial RLT formulation was constructed by multiplying both sets of association constraints by edge indexes — that is, the first set is multiplied by x_{kh} for $(i,k) \in E_1$, and the second set is multiplied by x_{kh} for $(j,h) \in E_2$:

$$\begin{aligned}
\textstyle\sum_j x_{ij} \le 1 \,\forall\, i \;&\Rightarrow \textstyle\sum_j x_{ij}x_{kh} \le x_{kh} \;\; \forall\, h, (i,k) \in E_1 \\
&\Rightarrow \textstyle\sum_j w_{ijkh} \le x_{kh} \quad \forall\, h, (i,k) \in E_1 \\
\textstyle\sum_i x_{ij} \le 1 \,\forall\, j \;&\Rightarrow \textstyle\sum_i x_{ij}x_{kh} \le x_{kh} \;\; \forall\, k, (j,h) \in E_2 \\
&\Rightarrow \textstyle\sum_i w_{ijkh} \le x_{kh} \quad \forall\, k, (j,h) \in E_2.
\end{aligned}$$

Table 9

Proteins	Nodes		Edges		# Binary Variables
1avy 1f22	58	48	56	55	2,784
1qr8 1qr9	54	55	75	72	2,970
1f22 1f22	48	48	55	55	2,304
8msi 9msi	58	59	108	112	3,422

Characteristics of CMO Problems

3.3.2. *Warm and Hot Starts*

The warm start associates the first n nodes of each graph in their given order, where $n = \min\{|V_1|, |V_2|\}$. In the model, this means

$$x_{ii} = 1 \text{ for } i = 1, \ldots, n.$$

Table 10

Proteins		Standard	RLT	Glover
1avy 1f22	Variables	3,080	310,964	2,784
	Constraints	6,160	12,038	2,784
1qr8 1qr9	Variables	5,400	431,427	2,970
	Constraints	10,800	18,813	2,970
1f22 1f22	Variables	3,025	250,415	2,304
	Constraints	6,050	11,330	2,304
8msi 9msi	Variables	12,096	740,620	3,422
	Constraints	24,192	37,060	3,422

Variables and Constraints Added for Linearizations of CMO

The hot starts were provided by Giuseppe Lancia, based on [30]. (We include those solution files and a MATLAB code to convert them into our AMPL hot start file at this article's supplement site.)

3.3.3. *Computational Results*

Our numerical results are reported in Table 12, and we summarize our findings in Table 13. The RLT stands out, as it solved three of the problems to optimality, and for problem 1avy1f22, it found a better solution from a cold start than the other methods. It is encouraging that CQPR found the optimal solution for problem 1f221f22 from a cold start (although it was unable to confirm optimality). Both the Standard and Glover linearizations performed poorly.

Table 12

Proteins	Warm Start	Hot Start	Best	Method
1avy 1f22	5	21	21	Hot Start
1qr8 1qr9	24	61	61	RLT
1f22 1f22	55	55	55	RLT
8msi 9msi	10	105	105	RLT

Solution Summary for CMO

3.4. *Rotamer Assignment*

Part of the protein folding problem is knowing the side-chain *conformations* — that is, knowing the torsion angles of the bonds. The rotation about a bond is called a rotamer, and there are libraries that give likelihoods for each amino acid. The Rotamer Assignment (RoA) Problem is to find an assignment of rotamers to sites that minimizes the total energy of the molecule. For the protein folding problem, we know the amino acid at each site. There are about 10 to 50 rotamers per amino acid, depending upon what else we know (such as knowing that the amino acid is located in a helix), so there are about 10^n to 50^n rotamer assignments for a protein of length n.

Besides its role in determining a protein's structure, the RoA Problem is a useful tool in drug design. Specifically, the RoA Problem can be used to determine a minimum-energy docking site for a *ligand*, which is a small molecule, such as a hormone or neurotransmitter. The ligand-protein docking problem is characterized by only a few sites because a ligand is (by definition) a small molecule. If the protein is known, the problem dimensions are small enough that the RoA Problem can be solved exactly within a minute. But, if the protein is to be engineered, there can be about 500 rotamers per site (20 acids @ 25 rotamers each). (Fung et al. [18,19] considered protein design without rotamer assignments, just the amino acid assignments, which are at most 20 per site.) There are other bioengineering problems associated the RoA Problem, such as determining protein-protein interactions. While the mathematical structure is the same, the applications have different energy data which can affect algorithm performance.

We note that it is often useful to determine multiple near-optimal solutions to the RoA Problem because the energy data is approximate. The QBP approach provides a distinct advantage over heuristics, not only because it can (theoretically) guarantee optimality, but also because binary solutions can be ranked by sequentially eliminating previous solutions with the exclusionary constraints:

$$\sum_{j\in\sigma(x^k)} x_j \le \left|\sigma(x^k)\right| - 1 \text{ for all } k,$$

where x^k is a previously generated solution and $\sigma(x^k) = \{j : x^k_j = 1\}$ (called the *support set* of x^k). The use of index j is generic, and the above applies to any QBP. The domain of x for the RoA Problem is a double index, defined in the next section.

There are several approaches for solving the RoA Problem in the literature. Gordon et al. [22] used a strengthened form of Dead-End Elimination (DEE) [21,23], and Kingsford et al. [12] considered semi-definite bounding instead of LPR. Xie and Sahinidis [36] applied a global optimization algorithm using iterative reductions with DEE, combined with local search. Here, we develop a QBP formulation, equivalent to that of Fung et al. [18,19].

Table 11

		Linearization						No linearization	
		Standard		RLT		Glover		(CQPR)	
Proteins	Start	CPU	Value	CPU	Value	CPU	Value	CPU	Value
1avy 1f22	cold	T	none	T	4	T	1	T	none
	warm	T	$5^\dagger$	T	$5^\dagger$	T	$5^\dagger$	T	$5^\dagger$
	hot	T	$21^\dagger$	T	$21^\dagger$	T	$21^\dagger$	T	$21^\dagger$
1qr8 1qr9	cold	T	none	28.75	61	T	2	T	none
	warm	T	$24^\dagger$	29.80	61	T	$24^\dagger$	T	$24^\dagger$
	hot	T	$61^\dagger$	21.25	$61^\dagger$	T	$61^\dagger$	T	$61^\dagger$
1f22 1f22	cold	T	2	17.99	55	T	1	T	55
	warm	T	$55^\dagger$	16.46	$55^\dagger$	T	$55^\dagger$	T	$55^\dagger$
	hot	T	$55^\dagger$	16.49	$55^\dagger$	T	$55^\dagger$	T	$55^\dagger$
8msi 9msi	cold	T	none	16.90	105	T	none	T	none
	warm	T	$10^\dagger$	17.73	105	T	$10^\dagger$	M	none
	hot	T	$105^\dagger$	15.32	$105^\dagger$	T	$105^\dagger$	M	none

†Same as initial value; M out of memory (16 GB); T out of time (1 hr).

Unix CPU Times and Best Values Found for QBP of CMO

3.4.1. *QBP Model and Test Data*

Our QBP model is as follows. Let $r \in \mathcal{R}_i$ = set of rotamers that can be assigned to site i, and

$$x_{ir} = \begin{cases} 1 & \text{if rotamer } r \text{ is assigned to site } i; \\ 0 & \text{otherwise.} \end{cases}$$

Then, the QBP for the RoA problem is the quadratic semi-assignment problem:

$$\min \sum_i \sum_{r \in \mathcal{R}_i} \left(\mathcal{E}_{ir} x_{ir} + \sum_{j>i} \sum_{t \in \mathcal{R}_j} E_{irjt} x_{ir} x_{jt} \right) : \\ \textstyle\sum_{r \in \mathcal{R}_i} x_{ir} = 1 \; \forall i, \; x \in \{0,1\}.$$

The objective function includes two types of energy: (1) within a site, $\mathcal{E}_{ir}$, and (2) between rotamers of two different sites, E_{irjt} for $i \neq j$. The summation condition $j > i$ is to avoid double counting, where $E_{irjt} = E_{jtir}$.

We ran four problems, based on proteins from the PDB; the "Full" indicates that many rotamers are possible for each site, using energy data provided by Diana Roe, Sandia National Laboratories, from her molecular dynamics simulation. Table 14 contains problem characteristics, and Table 15 contains the number of auxiliary variables and constraints needed by the linearizations. We include the search space size in Table 14 to have a comparative sense of problem size, particularly with the test problems used in [18]. (For rotamer assignment and for the *de novo* protein design problem, there could be additional conflict constraints that reduce the search space by disallowing combinations of assignments, particularly at neighbor sites, but this would not reduce the number of binary variables.)

Table 13

Protein	Sites n	Rotamers $\sum_{i=1}^n \lvert\mathcal{R}(i)\rvert$	# Binary Variables	Search Space Size $\prod_{i=1}^n \lvert\mathcal{R}(i)\rvert$
1aboFull	10	1,546	1,546	$3.418\ 10^{21}$
1bbzFull	10	1,614	785	$4.602\ 10^{21}$
1ddvFull	6	1,016	1,016	$9.111\ 10^{12}$

Characteristics of RoA Problems

Table 14

Protein		Standard	RLT	Glover
1aboFull	Variables	247,635	1,044,837	1,546
	Constraints	490,274	13,914	1,546
1bbzFull	Variables	146,443	264,873	785
	Constraints	290,046	7,065	785
1ddvFull	Variables	369,676	392,052	1,016
	Constraints	725,828	5,080	1,016

Variables and Constraints Added for Linearizations of RoA

Since the only constraints of the RoA Problem are assignment equations, we implemented the full level-1

RLT formulation. That is, we multiplied the assignment constraints by x_{jt} for all (j,t) with $j \neq i$:

$$\sum_{r\in\mathcal{R}_i} x_{ir} = 1\ \forall\, i \Rightarrow \sum_{r\in\mathcal{R}_i} x_{ir}x_{jt} = x_{jt}\ \forall\, i,j,t,j \neq i.$$

Upon substituting $x_{ir}x_{jt}$ with w_{irjt} for $i < j$ we obtain the RLT constraints:

$$\begin{aligned}\sum_{r\in\mathcal{R}_i} w_{irjt} &= x_{jt}\ \forall\, i < j, t \in \mathcal{R}_j \\ \sum_{r\in\mathcal{R}_i} w_{jtir} &= x_{jt}\ \forall\, i > j, t \in \mathcal{R}_j\end{aligned} \tag{12}$$

The auxiliary constraints of the Standard linearization are implied by (12) with $w \geq 0$, so we do not include them. (This observation is implied by the reductions discussed in the seminal work of Adams and Sherali [5].)

The full level-1 RLT is thus:

$$\begin{aligned}\min\ \textstyle\sum_i \sum_{r\in\mathcal{R}_i} \Big(\mathcal{E}_{ir}x_{ir} + \sum_{j>i}\sum_{t\in\mathcal{R}_j} E_{irjt}w_{irjt}\Big) :\\ \textstyle\sum_{r\in\mathcal{R}_i} w_{irjt} = x_{jt}\ \forall\, i < j, t \in \mathcal{R}_j\\ \textstyle\sum_{r\in\mathcal{R}_i} w_{jtir} = x_{jt}\ \forall\, i > j, t \in \mathcal{R}_j\\ w \geq 0,\ x \in \{0,1\}.\end{aligned}$$

Note in Table 15 that the level-1 RLT has more auxiliary variables than the Standard linearization, but has significantly fewer auxiliary constraints. This is because the continuous relaxation of the RLT is at least as tight as the Standard linearization. In fact, it is typically much tighter!

3.4.2. *Warm and Hot Starts*

The warm start assigns the min-energy rotamer to each site, ignoring energies across sites. This is equivalent to minimizing just the linear term:

$$\min_x \sum_i \sum_{r\in\mathcal{R}_i} \mathcal{E}_{ir}x_{ir} = \sum_i \min_{r\in\mathcal{R}_i} \mathcal{E}_{ir}.$$

The hot start uses DEE in an algorithm by William Hart (unpublished), who provided the hot start files.

3.4.3. *Computational Results*

Our numerical results are presented in Table 16. What is striking is that Glover's linearization solved all problems to confirmed optimality in less than a minute, even with a cold start! RLT also solved each problem to optimality and took less time than Glover's in two of the test problems (consistent with [18]). Note that the objective value in some cases appears to be the same as the optimal value (found by Glover's linearization and the RLT), but that is due to rounding within the table. For example, the optimal value for 1bbzFull is -58.294, but the closest that the other two methods came is -58.124.

The Standard linearization performed poorly — it could not obtain a feasible solution for any of the cold starts, nor could it improve upon the warm and hot starts. The CQPR did quite well, and actually found the optimal solution to 1aboFull, although it was not able to confirm optimality. We mention that our tests under Windows were very similar in nature; there were no memory errors for any of the problem instances.

Table 17 summarizes the solutions. Overall, these experiments suggest that the RLT and Glover's linearization are well suited to RoA, and the Standard linearization is not. We suggest that CQPR should be explored for further development.

Table 16

Protein	Warm Start	Hot Start	Best	Method
1aboFull	131	-54	-59	RLT & Glover
1bbzFull	1.3×10^6	-58	-58	RLT & Glover
1ddvFull	-34	-35	-35	RLT & Glover

Solution Summary for RoA

Fung et al. [18] did a comprehensive study of different RLT formulations. Their analysis differs from ours in that they considered several variations. The main issue, given the question we address throughout this study, is that the RLT formulations require w_{irjt} to be defined even if $E_{irjt} = 0$. Our test problems are much larger than theirs, and the sparsity becomes increasingly important as we increase the number of sites and/or the number of possible assignments per site. The *de novo* protein problem that they address with the same quadratic semi-assignment model has 20 amino acids per site, whereas our small test problems have more than five times that number. Their results combined with ours suggest that RLT is an excellent approach when the problem size is small enough that memory is not an issue, but its need for huge amounts of memory may make it prohibitive to use for very large QBP.

Table 15

		Linearization						No linearization	
		Standard		RLT		Glover		(CQPR)	
Protein	Start	CPU	Value	CPU	Value	CPU	Value	CPU	Value
1aboFull	cold	T	none	2.71	−59	0.15	−59	T	−59
	warm	T	131†	2.82	−59	0.14	−59	T	−59
	hot	T	−54†	2.76	−59	0.14	−59	T	−59
1bbzFull	cold	T	none	0.30	−58	0.74	−58	T	−58
	warm	T	$1.3\times10^{6\dagger}$	0.34	−58	0.73	−58	T	−58
	hot	T	−58†	0.34	−58	0.56	−58	T	−58†
1ddvFull	cold	T	none	0.83	−35	0.42	−35	T	−30
	warm	T	−34†	0.87	−35	0.45	−35	T	−34†
	hot	T	−35†	0.89	−35	0.44	−35	T	−35†

†Same as initial value; CPU is minutes; T out of time (1 hr).

Unix CPU Times and Best Values Found for QBP of RoA

4. Conclusions

In this paper we proposed QBP models of four problems in computational molecular biology and studied their effectiveness under a general solver. Such a unified approach has three main advantages. First, each problem can be solved using standard optimization software. This allows us to benefit from the robustness of mixed-integer programming software and provides an alternative to the tailored algorithms used in practice, which can be difficult to implement and maintain. Second, we could easily include additional constraints without a new algorithm design. Third, while current solution methods consist of heuristic and approximation algorithms, QBP models could, in theory, be used to find exact solutions. Further, as software and hardware improve, we could generate a ranked list of solutions that include alternative and near-optimal solutions. These are grand goals, and our study suggests that we are not quite able to fulfill them for all problems considered. There are, however, many signs of encouragement, and we suggest some avenues for further research.

One avenue is to consider a preprocessor at the modeling level that would exploit problem structure and generate strong cuts before passing the problem to the solver. We did this by partially applying the level-1 RLT to selected constraints. For both simplicity and consistency, we applied the RLT to the assignment constraints contained in our problems, but more research could be done in selecting constraints and variables from which to generate the RLT restrictions. Such research would need to find a balance between the strength offered by the RLT constraints and the memory requirements of implementing them.

Beyond the use of RLT for LPR strengthening, there are other avenues to explore for *generic* cut generation, such as the derivation of clique inequalities for the CMO Problem. The goal, however, is to exploit general properties of these biology problems without relying on knowledge of the specific problem is at hand. This preprocessing would be done at the symbolic model level, rather than on a particular instance.

In addition to *pre*-solve processing, one could develop an interrupt capability in the interface between modeling and solving routines such that model-driven directives could be applied, such as symmetry exclusion constraints for the LPF Problem. Moreover, local branch-and-bound information could be used to selectively generate RLT restrictions.

Another avenue for future work is to adapt the state-of-the-art solution strategies that have been developed around the RLT-methodology for the *Quadratic Assignment Problem* [4] and *Quadratic Knapsack Problem* [10] to these QBPs. These successful implementations of the RLT have been achieved by exploiting the block-diagonal structure found within the RLT formulations. While it is unlikely that such tailored algorithms would directly compete with current methods used in practice, especially because heuristics seem to produce good answers quickly, they have the potential to confirm optimality, find an improved solution, and/or add insight to the solution(s) found.

An investigation into extensions of the QBP models will result in a more comprehensive analysis of our fundamental question. The MSA problem can be extended to find a *best local alignment* by dropping the constraint that every character be assigned and add a contiguity assignment constraint. The LPF problem can be extended to allow any *collection of properties* to score neighbors, besides hydrophobicity, such as their charge, by indexing $\mathcal{H}$ for each property. The CMO problem can be extended to use edge values that measure closeness or bond strength to measure the "goodness" of an edge-pair association, making CMO a *weighted-overlap* problem. The RoA problem can be extended by adding *conflict constraints* that disallow certain combinations of rotamer assignments. Such model enhancements generally invalidate current heuristics that have been tailored to the original form, but we can simply revise the QBP models.

In conclusion, our QBP approach seems viable for both the the RoA and CMO problems, while we had limited success for the MSA and LPF Problems. Our results show that the RLT is well-suited for the CMO Problem, while both the RLT and Glover's linearization work well for the RoA Problem. The RLT formulation appears to be a good choice when the size of the QBP model is not too large, whereas Glover's linearization seems to be a well-rounded approach, performing well under most circumstances. We emphasize the potential for the convex reformulation approach, which has the immediate advantage that it can be submitted directly to a solver without incurring auxiliary variables and constraints. Finally, we hope that our formulations will provide operations researchers who specialize in QBP an opportunity to extend current solution approaches.

Acknowledgments

We thank César Rego for running his heuristic on our proteins for our hot start on the LPF. We thank Giuseppe Lancia for providing the contact map data files and his solutions to the CMO problems, which we used for hot starts. We thank William Hart for providing his code to obtain hot starts for our RoA problems, and we thank Diana Roe for the energy data. We thank Robert Fourer for his frequent help with AMPL, Ed Klotz for his help with CPLEX quadratic convexification, and David Gay's prompt attention to fixing a problem in the AMPL interface. We thank Warren Adams and Hanif Sherali for enlightening discussions about RLT. We thank Fred Glover for reading an earlier version of this paper and suggesting several improvements, and for his encouragement with our approach. We also thank Warren Adams, David Gay, and Wei Xie for noting some changes, which we implemented. Last, but not least, we thank two referees who raised questions that provoked us to extend and clarify some things, notably the inclusion of the RLT linearizations.

References

[1] W.P. Adams and R.J. Forrester. A simple recipe for concise mixed 0-1 linearizations. *Operations Research Letters*, 33(1):55–61, 2005.

[2] W.P. Adams and R.J. Forrester. Linear forms of nonlinear expressions: New insights on old ideas. *Operations Research Letters*, 35(4):510–518, 2007.

[3] W.P. Adams, R.J. Forrester, and F. Glover. Comparisons and enhancement strategies for linearizing mixed 0-1 quadratic programs. *Discrete Optimization*, 1(2):99–120, 2004.

[4] W.P. Adams, M. Guignard, P.M. Hahn, and W.L. Hightower. A level-2 reformulation-linearization technique bound for the quadratic assignment problem. *European Journal of Operational Research*, 180(3):983–996, 2007.

[5] W.P. Adams and H.D. Sherali. A tight linearization and an algorithm for zero-one quadaratic programming problems. *Management Science*, 32(10):1274–1290, 1986.

[6] R. Backofen and S. Will. Excluding symmetries in constraint-based search. *Constraints*, 7(3–4):333–349, 2002.

[7] H.M. Berman, K. Henrick, and H. Nakamura. Announcing the worldwide protein data bank. *Nature Structural Biology*, 10(12):980, 2003. PDB is at http://www.pdb.org/.

[8] S. Billionnet, S. Elloumi, and M-C. Plateau. Convex quadratic programming for exact solution of 0-1 quadratic programs. Technical Report 856, Laboratoire CEDRIC, Institut d'Informatique d'Entreprise, Paris, FR, 2005.

[9] J. Błażewicz, P. Formanowicz, and M. Kasprzak. Selected combinatorial problems of computational biology. *European Journal of Operational Research*, 161(3):585–597, 2005.

[10] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137, 1999.

[11] N. Castells-Brooke. *Beginner's Guide to Molecular Biology*. Rothamsted Research, http://www.rothamsted.ac.uk/notebook/courses/guide/, 2004.

[12] B. Chazelle, C.L. Kingsford, and M. Singh. A semidefinite approach to side-chain positioning with new rounding schemes. *INFORMS Journal on Computing*, 16(4):380–392, 2004.

[13] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. In M.O. Dayhoff, editor, *Atlas of Protein Science and Structure*, volume 5, Supplement 3, pages 345–352, Silver Spring, MD, 1978. National Biomedical Research Foundation.

[14] K.A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.

[15] K.A. Dill, S. Bromberg, K. Yue, K.M. Fiebig, D.P. Yee, P.D. Thomas, and H.S. Chan. Principles of protein folding — a perspective from simple exact models. *Protein Science*, 4(4):561–602, 1999.

[16] R.F. Doolittle. *Of Urfs and Orfs: A Primer on How to Analyze Derived Amino Acid Sequences*. University Science Books, Mill Valley, CA, 1987.

[17] G. Fuellen. A gentle guide to multiple alignment. Course note, Universität Bielefeld, Virtual School of Natural Sciences BioComputing Division (VSNS-BCD), 1997. Available at http://www.techfak.uni-bielefeld.de/bcd/Curric/MulAli/mulali.html.

[18] H.K. Fung, S. Rao, C.A. Floudas, O. Prokopyev, P.M. Pardalos, and F. Rendl. Computational comparison studies of quadratic assignment like formulations for the in silico sequence selection problem in de novo protein design. *Journal of Combinaotrial Optimization*, 10:41–60, 2005.

[19] H.K. Fung, M.S. Taylor, and C.A. Floudas. Novel formulations for the sequence selection problem in de novo protein design with flexible templates. *Optimization Methods and Sofware*, 22(1):51–71, 2007.

[20] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[21] R.F. Goldstein. Efficient rotamer elimination applied to protein side-chains and related spin-glasses. *Biophysical Journal*, 66(5):1335–1340, 1994.

[22] D.B. Gordon, G.K. Hom, S.L. Mayo, and N.A. Pierce. Exact rotamer optimization for protein design. *Journal of Computational Chemistry*, 24(2):232–243, 2003.

[23] D.B. Gordon and S.L. Mayo. Radical performance enhancements for combinatorial optimization algorithms based on the dead-end elimination theorem. *Journal of Computational Chemistry*, 19(13):1505–1514, 1998.

[24] H.J. Greenberg, W.E. Hart, and G. Lancia. Opportunities for combinatorial optimization in computational biology. *INFORMS Journal on Computing*, 16(3):211–231, 2004.

[25] P.L. Hammer and A.A. Rubin. Some remarks on quadratic programming with 0-1 variables. *R.I.R.O.*, 4:67–79, 1970.

Received 9 January 2007; revised 19 March 2007; accepted 20 June 2007

[26] W.E. Hart and S. Istrail. Fast protein folding in the hydrophobic-hydrophilic model within three-eights of optimal. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 157–168, New York, NY, 1995. ACM Special Interest Group on Algorithms and Computation Theory, ACM Press.

[27] W.E. Hart and S. Istrail. Lattice and off-lattice side chain models of protein folding (extended abstract): linear time structure prediction better than 86optimal. In *Proceedings of the first annual international conference on Computational molecular biology*, pages 137–146, New York, NY, 1997. ACM Press.

[28] A. Holder, editor. *Mathematical Programming Glossary*. INFORMS Computing Society, http://glossary.computing.society.informs.org, 2006–07.

[29] V. Ingram. *Biology Hypertextbook*. MIT, http://web.mit.edu/esgbio/www/, 2006.

[30] G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optima1 PDB structure alignments: A branch-and-cut algorithm for the maximum contact map overlap problem. In *Proceedings of the Fifth Annual International Conference on Computational Biology*, pages 143–202, New York, NY, 2001. ACM Press.

[31] S.B. Needleman and C.D. Wunch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:444–453, 1970.

[32] C. Rego, H. Lee, and F. Glover. A filter-and-fan approach to the 2D lattice model of the protein folding problem. Technical report, University of Mississippi, University, MS, 2006. Available at http://faculty.bus.olemiss.edu/crego/Papers.html.

[33] J.R. Riordan, J.M. Rommens, B. Kerem, N. Alon, R. Rozmahel, Z. Grzelczak, J. Zielenski, S. Lok, N. Plavsic, and J.L. Chou. Identification of the cystic fibrosis gene: cloning and characterization of complementary DNA. 245(4922):1066–1073, 1989.

[34] H.D. Sherali and W.P. Adams. A hierarchy of relaxations between the continuous and convex hull representation for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[35] M.S. Waterman. Efficient sequence alignment algorithms. *Journal of Theoretical Biology*, 108(3):333–337, 1984.

[36] W. Xie and N.V. Sahinidis. Residue-rotamer-reduction algorithm for the protein side-chain conformation problem. *Bioinformatics*, 22(2):188–194, 2006.

[37] W. Xie and N.V. Sahinidis. A reduction-based exact algorithm for the contact map overlap problem. *Journal of Computational Biology*, 14(5):637–654, 2007.