Improving Mathematical Programming Approaches for Motif Finding

Carl Kingsford, Elena Zaslavsky, and Mona Singh

Department of Computer Science and Lewis-Sigler Institute for Integrative Genomics,
Princeton University, Princeton, NJ 08544
{carlk,elenaz,msingh}@cs.princeton.edu

Abstract. The motif finding problem is to locate a collection of mutually similar subsequences within a given set of DNA sequences. This is an important problem, as such shared motifs often correspond to regulatory elements. We study a combinatorial framework for the motif finding problem, where the goal is to find a minimum (or maximum) weighted clique in a k-partite graph. Previous approaches to find these cliques have relied on graph pruning and divide-and-conquer techniques. Recently, it has been shown that mathematical programming is a promising approach for motif finding. Here, we describe a novel integer linear programming formulation for the problem. A key observation driving our formulation is that the weights on the edges in the graph come from a small set of possibilities. We show that our new formulation leads to a method that is highly effective in practice on instances arising from biological sequence data. We are able to solve these problems to optimality often many times faster than the existing mathematical programming approach.

1 Introduction

A central challenge in post-genomic biology is to reconstruct the regulatory network of an organism. A key step in this process is the discovery of regulatory elements. A commonly studied paradigm starts with a set of DNA sequences that contain binding sites for a common regulatory protein, and then finds shared (or similar) subsequences in each. These subsequences, or *motifs*, are putative binding sites for the same factor. The effectiveness of identifying regulatory elements in this manner has been demonstrated when considering sets of sequences identified via shared co-expression [29], orthology [6, 12], and genome-wide location analysis [16]).

From a computational point of view, the motif finding problem can be formulated in different ways, and while many methods work reasonably well, a recent comprehensive study by [31] shows that no single motif finding method exhibits a high absolute measure of correctness. Broadly speaking, the methods are either probabilistic or combinatorial. Probabilistic approaches estimate parameters of a motif model using maximum likelihood or maximum *a posterior* estimation to find the parameters of these models [15, 4, 14, 17, 9]. Combinatorial approaches either enumerate through all allowed motifs (e.g., [30, 27, 18, 32,

8, 21]), or attempt to maximize some measure based on sequence similarity, or minimize some measure based on distance (e.g., [11, 24, 22, 28]).

We take the combinatorial approach and formulate the motif finding problem as that of finding the best gapless local multiple sequence alignment using the sum-of-pairs (SP) scoring scheme. The SP-score is one of many reasonable schemes for assessing motif conservation [20, 26]. The combinatorial problem is equivalent to that of finding a minimum weight clique of size p in a p-partite graph (e.g. [23, 22, 28]).

For general notions of distance, this problem is NP-hard to approximate within any reasonable factor [5]. In the motif finding setting, where the distances obey the triangle inequality, the problem remains NP-hard [1]. While constant-factor approximation algorithms exist [10, 3], the ability to find the optimal solution in practice is preferable.

Our approach follows that of [33], which introduced the integer linear programming (ILP) formulation of the motif finding problem. Their testing on identifying known DNA binding sites of *E. coli* transcription factors [25] shows that the approach performs well for motif finding, identifying either known motifs or motifs of higher conservation. However, a difficulty mentioned in [33] is the size of the integer linear programs, which can have millions of variables for interesting biological problems. In that work, the authors tackle the ILPs by preprocessing with graph pruning and decomposition techniques. Here, we take an alternate direction and propose a novel, more compact ILP that cleverly utilizes the discrete nature of the distance metric imposed on pairs of subsequences. We present a class of constraints to make the linear programming relaxation of the new formulation provably as tight as that given in [33]. Instead of introducing all of these additional constraints, we describe and test a heuristic approach to solve the LP relaxation of our novel ILP formulation that, in all observed cases, finds a solution of the same objective value as the LP relaxation of [33], often an order of magnitude faster. Moreover, we show that in practice, the LP relaxations for both of the ILP formulations often have integral optimal solutions, making solving the LP relaxations sufficient for solving the original ILP. Even if this were not the case, the ability to find faster solutions to the relaxations may translate into significant speed-ups in branch-and-bound approaches for ILP solving.

2 Formal Problem Specification

In the motif finding problem, we are given p sequences, which are assumed without loss of generality to each have length N', and a motif length ℓ. The goal is to find a substring s_i of length l in each sequence i, such that the sum of the pairwise distances between the substrings (i.e., $\sum_{i<j} \mathbf{distance}(s_i, s_j)$) is minimized. The distance between substrings may be defined in several ways. The simplest measure, and the one we restrict ourselves to in this paper is the Hamming distance.

It is convenient to rephrase the motif finding problem in graph theoretic terms [23]. For a problem with p sequences, we define a complete, weighted p-

partite graph, with a part V_i for each sequence. In V_i, there is a node for every possible window of length ℓ in sequence i. Thus there are $N := N' - \ell + 1$ nodes in each V_i, and the vertex set $V = V_1 \cup \cdots \cup V_p$ has size Np. For every pair u and v in different parts there is an edge $(u, v) \in E$. Letting $\mathbf{seq}(u)$ denote the subsequence corresponding to node u, the weight w_{uv} on edge (u, v) equals $\mathbf{distance}(\mathbf{seq}(u), \mathbf{seq}(v))$. The goal in motif finding is to choose a node from each part so as to minimize the weight of the induced subgraph. We note that the combinatorial formulation of the "subtle motifs" problem is similar [22], though in that case edges exist only between nodes corresponding to subsequences that are within a certain distance of one another. The approach we outline below can be extended to that context as well.

3 Integer and Linear Programming Formulations

3.1 Original integer linear programming formulation

[33] introduced the following integer linear programming (ILP) formulation of the motif finding problem. In this ILP formulation, there is a variable X_u for each node u in the graph described above. The variable X_u is set to 1 if node u is chosen, and 0 otherwise. Additionally, there is one variable X_{uv} for each edge in the graph (X_{uv} is the same as X_{vu}). These edge variables are set to 1 if both end points of the edge are chosen. In the integer programming setting all variables are constrained to take values from $\{0, 1\}$. The following ILP is easily seen to model the above graph problem:

$$\begin{array}{ll} \text{Minimize} & \sum_{\{u,v\}\in E} w_{uv} \cdot X_{uv} \\ \\ \text{subject to} & \\ \quad \sum_{u\in V_i} X_u = 1 & \text{for } i = 1, \ldots, p \\ \quad \sum_{u\in V_i} X_{uv} = X_v & \text{for } i = 1, \ldots, p \text{ and } v \in V \setminus V_j \end{array} \tag{IP1}$$

The first set of constraints ensures that one node is chosen from each part, and the second set requires that an edge is chosen if its end points are. This ILP is the same as the ILP formulation for side-chain positioning presented in [13], and similar to the ILP in [2].

3.2 New integer linear programming formulation

Since the alphabet and the length of the sequences are finite, there are only a finite number of possible pairwise distances. For example, in the case of Hamming distances, edge weights can only take on $\ell+1$ different values. We take advantage of the small number of possible weights and the fact that the edge variables of IP1 are only used to ensure that if two nodes u and v are chosen in the optimal solution then w_{uv} is added to the cost of the clique. We introduce a second ILP in which we no longer have edge variables X_{uv}. Instead, in addition to the node variables X_u, we have a variable Y_{ujc} for each node u, each position j such that u

is not in V_j, and each possible edge weight c. These Y variables model groupings of the edges by cost into *cost bins*. The intuition is that Y_{ujc} is 1 if node u and some node $v \in V_j$ are chosen s. t. $w_{uv} = c$. Formally, let D be the set of possible edge weights and let $W = \{(u, j, c) : c \in D, u \in V, j \in 1, \ldots p \text{ and } u \notin V_j\}$ be the set of triples over which the Y_{ujc} variables are indexed. Then the following ILP models the motif-finding graph problem:

$$\text{Minimize} \quad \sum_{(u,j,c)\in W:\text{part}(u)<j} c \cdot Y_{ujc}$$

subject to

$$\begin{array}{lll} \sum_{u\in V_i} X_u = 1 & \text{for } i = 1, \ldots, p & \text{(IP2a)} \\ \sum_{c\in D} Y_{ujc} = X_u & \text{for } j \in 1, \ldots, p \text{ and } u \in V \setminus V_j & \text{(IP2b)} \\ \sum_{v\in V_j:w_{uv}=c} Y_{vic} \geq Y_{ujc} & \text{for } (u, j, c) \in W \text{ s.t. part}(u) < j & \text{(IP2c)} \end{array}$$

(IP2)

As in the previous formulation, the first set of constraints forces a single node to be chosen in each part. The second set of constraints says that if a node u is chosen, for each j, one of its "adjacent" cost bins must also be chosen (Fig. 1). The third set of constraints ensures that Y_{ujc} can be chosen only if some node $v \in V_j$ is also chosen such that $w_{uv} = c$. We discard variables Y_{ujc} if there is no $v \in V_j$ such that $w_{uv} = c$. Fig. 1 gives a schematic drawing of these constraints.

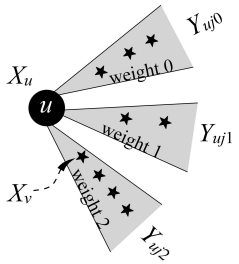


Fig. 1. Schematic of IP2. Adjacent to each node u there are at most $|D|$ cost bins, each associated with a variable Y_{ujc}. Associated with each cost c are the nodes $v \in V_j$ for which $w_{uv} = c$ (represented in the figure by stars). Constraints (IP2b) say that we must spread a total of X_u weight apportioned over the bins, while constraints (IP2c) limit us to choosing cost bin variables where there is some node $v \in V_j$ chosen such that $w_{uv} = c$.

It is straightforward to see that IP2 correctly models the motif-finding problem if the variables are $\in \{0, 1\}$. For any choice of p-clique $\{u_1, \ldots, u_p\}$ of weight $\mu = \sum_{i<j} w_{u_i u_j}$, a solution of cost γ to IP2 can be found by taking $X_{u_i} = 1$ for $i = 1 \ldots, p$, and taking $Y_{u_i jc} = 1$ for all $1 \leq j \leq p$ such that $w_{u_i u_j} = c$. This solution is easily seen to be feasible, and between any pair of positions i, j it contributes cost $w_{u_i u_j}$; therefore, the total cost is γ. On the other hand, consider any solution (X, Y) to IP2 of objective value γ. Consider the clique formed by the nodes u such that $X_u = 1$. Between every two positions $i < j$, the constraints (IP2a) and (IP2b) imply that exactly one one Y_{ujc} and one Y_{vid} are set to 1 for some $u \in V_i$ and $v \in V_j$ and costs c, d. Constraint (IP2c) corresponding to (u, j, c) with Y_{ujc} on its right-hand side can only be satisfied if the sum on its left-hand side is 1, which implies $c = d = w_{uv}$. Thus, a clique of weight γ exists in the motif-finding graph problem.

3.3 Advantages of IP2

In practice, IP2 has many fewer variables than IP1. IP1 has $Np(N(p-1)/2+1)$ variables and $p + Np(p-1)$ constraints. If none of the Y_{ujc} variables can be thrown out, IP2 has $Np((p-1)d+1)$ variables and $p + Np(p-1)(d/2+1)$ constraints, where $d = |D|$, the number of kinds of weights. If $d < N/2$, the second IP will have fewer variables. In practice, d is expected to be much smaller than N, and while N could reasonably be expected to grow large as longer and longer sequences become practical to study, d is constrained by the geometry of transcription factor binding and will remain small. Also, in practice, it is likely that many Y_{ujc} variables are removed because $\mathbf{seq}(u)$ does not have matches of every possible weight in each of the other sequences. IP2, on the other hand, will have $O(d)$ times more constraints than IP1.

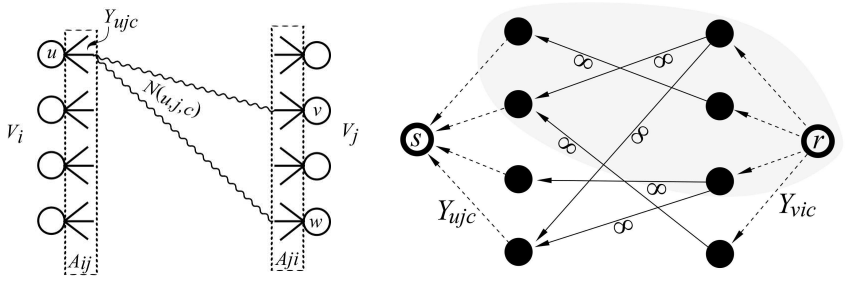
While the space requirement for the simplex algorithm is related to the number of constraints and variables, running time is not necessarily directly related. Smaller integer programs with weaker LP relaxations are often less useful for branch-and-bound approaches to IP solving. Thus, we seek the tightest, smallest IP possible. In the end, experiments must be performed to gauge the efficacy of various formulations on practical problems. We present experiments below which suggest IP2 can be more than an order of magnitude faster than IP1.

The recasting of the problem such that the number of edge weights becomes the determining factor of problem size suggests several avenues for practical performance enhancements. For example, it is often the case that one is interested in an optimal solution only if it is of high enough quality, meaning that no motif instance in the solution is further than α away from any other (the *diameter* of the solution is $\leq \alpha$). If this is the case, edges of weight $> \alpha$ can be deleted. Such a requirement reduces d and makes IP2 still smaller. In many applications, even if large diameter solutions are acceptable, there is an expectation that the diameter is likely small, and a solution with a small diameter may be preferred to one with a lower sum-of-pairs score but more outliers. In such a case, the IP2 formulation may allow one to check for low diameter solutions quickly.

3.4 Linear Programming Relaxation

The typical approach to solving an ILP is to solve the linear program derived from the ILP by dropping the requirement that the variables be in $\{0, 1\}$, and instead requiring only that the variables lie in the continuous range $[0, 1]$. This modified problem is called the linear programming (LP) relaxation. Efficient algorithms are known for solving linear programs. In the case of minimization, an ILP formulation is *weaker* than another if the corresponding LP relaxation of the first admits a solution of lower objective value than is possible with the second (*stronger* formulation is defined accordingly). Weaker relaxations are often less useful in solving the corresponding ILP.

The LP relaxation of IP1, which we refer to as LP1, is stronger than the LP relaxation of IP2 as stated. In this section, we present a fairly natural (though exponential) class of constraints that, if added to the LP relaxation of IP2, makes



(a) (b)

Fig. 2. (a) $\mathcal{N}(u, j, c)$ is shown assuming that v and w are the only nodes in V_j that have cost c with u. (b) Graph used to show (1) are sufficient.

the two formulations equivalent. We refer to this fully constrained relaxation of IP2 as LP2. In the subsequent sections, we provide a separation algorithm and also show that in practice we can focus on just two types of these constraints in LP2, and we are able to solve the original ILP iteratively by adding cutting planes corresponding to violated constraints of these types.

Additional constraints. Focus on a single pair of positions i and j. In IP1 the edge variables between V_i and V_j explictly model the bipartite graph between those two positions. In IP2, however, the bipartite graph is only implicitly modeled by an understanding of which Y variables are compatible to be chosen together. We study this implicit representation by considering the bipartite *compatibility* graph $\mathcal{C}_{ij}$ between two positions i and j. Intuitively, we have a node in this compatibility graph for each Y_{ujc} and Y_{vic}, and there is an edge between the nodes corresponding to Y_{ujc} and Y_{vic} if $w_{uv} = c$. These two Y variables are compatible in that they can both be set to 1 in IP2. More formally, $\mathcal{C}_{ij} = (A_{ij}, A_{ji}, F)$, where $A_{ij} = \{(u, j, c) : u \in V_i, c \in D\}$ is the set of indices of Y variables adjacent to a node in V_i, going to position j, and A_{ji} is defined analogously, going in the opposite direction. The edge set F is defined in terms of the neighbors of a triple (u, j, c). Let $\mathcal{N}(u, j, c) = \{(v, i, c) : u \in V_i, (v, i, c) \in A_{ji}$ and $w_{uv} = c\}$ be the *neighbors* of (u, j, c). They are the indices of the Y_{vic} variables adjacent to position j going to position i so that the edge $\{u, v\}$ has weight c. There is an edge in F going between (u, j, c) and each of its neighbors. We call c the *cost* of triple (u, j, c). All this notation is summarized in Fig. 2(a).

In any feasible integral solution, if $Y_{ujc} = 1$, then some Y_{vic} for which $(v, i, c) \in \mathcal{N}(u, j, c)$ must also be 1. Extending this insight to subsets of the Y_{ujc} variables yields a class of constraints which will ensure that the resulting linear programming formulation is as tight as LP1. If Q_{ij} is a subset of A_{ij}, then let $\mathcal{N}(Q_{ij}) = \bigcup_{(u,j,c)\in Q_{ij}} \mathcal{N}(u, j, c)$ be the set of indices that are neighbors to any vertex in Q_{ij}. If $Q_{ij} \subseteq A_{ij}$ then $\mathcal{N}(Q_{ij}) \subseteq A_{ji}$. The following constraint is

true in IP2 for any such Q_{ij}:

$$\sum_{(u,j,c)\in Q_{ij}} Y_{uic} \leq \sum_{(v,i,c)\in \mathcal{N}(Q_{ij})} Y_{vic}\,. \tag{1}$$

That is, choose any set of Y_{ujc} variables adjacent to position i that go to position j. The sum of the Y variables for their neighbors must be greater than or equal to the sum of the variables originally chosen. Notice that the third set of constraints in IP2 are of the form (1), taking Q_{ij} to be the singleton set $\{(u,i,c)\}$.

Theorem 1. *If for every pair $i < j$, constraints of the form* (1) *are added to IP2 for each $Q \subseteq A_{ij}$ such that all triples in Q are of the same cost then the resulting LP relaxation LP2 is as strong as the relaxation LP1 of IP1.*

Proof. For any feasible solution for LP2, we will show that there is a feasible solution for LP1 with the same objective value, thereby demonstrating that the optimal solution to LP2 is no smaller than the optimal solution to LP1. In particular, fix a solution (X, Y) to the LP2 with objective value γ. We need to show that for any feasible distribution of weights on the Y variables a solution to LP1 can be found with objective value γ.

In order to reconstruct a solution $\hat{X}$ for LP1 of objective value γ, we will set $\hat{X}_u = X_u$, using the values of the node variables X_u in the optimal solution to LP2. We must assign values to $\hat{X}_{uv}$ to complete the solution. Recall the compatibility graph $\mathcal{C}_{ij}$ described above. Because all edges in $\mathcal{C}_{ij}$ are between nodes of the same cost, $\mathcal{C}_{ij}$ is really $|D|$ disjoint bipartite graphs $\mathcal{C}^c_{ij}$, one for each cost. Let $A^c_{ij} \cup A^c_{ji}$ be the node set for the subgraph $\mathcal{C}^c_{ij}$ for cost c. Each edge in a subgraph $\mathcal{C}^c_{ij}$ corresponds to one edge in the graph G underlying LP1. Conversely, each edge in G corresponds to exactly one edge in one of the $\mathcal{C}^c_{ij}$ graphs (if edge $\{u, v\}$ has cost c_1, it corresponds to an edge in $\mathcal{C}^{c_1}_{ij}$). We will thus proceed by assigning values to the edges in the various $\mathcal{C}^c_{ij}$, and this will yield values for the $\hat{X}_{uv}$.

If $y(A) := \sum_{(u,j,c)\in A} Y_{ujc}$, by the first two sets of constraints in LP2, $y(A_{ij}) = y(A_{ji}) = 1$. Since the constraints (1) are included with $Q = A^c_{ij}$ for each cost c, by the pigeonhole principle, $y(A^c_{ij}) = y(A^c_{ji})$ for every cost c. Thus, for each subgraph $\mathcal{C}^c_{ij}$, the weight placed on the left half equals the weight placed on the right half. We will consider each induced subgraph $\mathcal{C}^c_{ij}$ separately.

We modify $\mathcal{C}^c_{ij}$ as follows to make it a directed, capacitated graph. Direct the edges of $\mathcal{C}^c_{ij}$ so that they go from A^c_{ji} to A^c_{ij}, and set the capacities of these edges to be infinite. Add two dummy nodes $\{r, s\}$ and edges directed from r to each node in A^c_{ji} and edges from each node in A^c_{ij} to s. Every edge adjacent to r and s is also adjacent to some node representing a Y variable. Put capacities on these edges equal to the value of the Y variable to which they are adjacent (see Fig. 2(b)).

The desired solution to LP1 can be found if the weight of the nodes (Y variables) in each compatibility subgraph can be spread over the edges. In other words, a solution to LP1 of weight γ can be found if, for each c, there is a flow

of weight $y(A^c_{ij})$ from r to s in the above constructed graph. The assignment to $\hat{X}_{uv}$ will be the flow crossing the corresponding edge in the $\mathcal{C}^c_{ij}$ of appropriate cost. Below we show that the set of constraints described in the theorem ensure that the minimum cut in the constructed graph is $\geq y(A^c_{ij})$, and thus that there is a flow of the required weight. The proof of this fact can be found in [7] on page 54-55, and is reproduced in our notation in the following lemma.

Lemma 1. *The minimum cut of the flow graph described in the proof of Theorem 1 (and shown in Fig. 2(b)) is $y(A^c_{ij})$.*

Proof. Recall that the capacities of the edges leaving r are Y_{vic} and those entering s are Y_{ujc}, and that the total capacity leaving r equals the total entering s, and this total capacity equals $y(A^c_{ij})$. We want to show that the minimum $r-s$ cut in this graph is $\geq y(A^c_{ij})$.

Consider an $r-s$ cut $\{r\} \cup A \cup B$ where $A \subseteq A^c_{ji}$ and $B \subseteq A^c_{ij}$. (Such a cut is shaded in Fig. 2(b).) If any edges go between A and $A^c_{ij} \setminus B$ then the capacity is infinite, and we are done. Otherwise the value of the cut is the sum of the capacities of the edges leaving r and going to $A^c_{ji} \setminus A$ plus the sum of the capacities of the edges entering s from a node in B. This value is $> y(A^c_{ij})$ if and only if

$$y(A^c_{ji} \setminus A) \geq y(A^c_{ij} \setminus B)\,. \tag{2}$$

Assume for a moment that all nodes in $A^c_{ji} \setminus A$ have a neighbor in $A^c_{ij} \setminus B$. Then $\mathcal{N}(A^c_{ij} \setminus B) = A^c_{ji} \setminus A$ because there are no edges between A and $A^c_{ij} \setminus B$. Thus, (2) is satisfied if and only if

$$y(Q) \leq y(\mathcal{N}(Q))\,, \tag{3}$$

for any $Q \subseteq A^c_{ij}$, where Q is now standing in for $A^c_{ij} \setminus B$ for various choices of B in various cuts. Eqn. (3) equivalent to (1).

If there had been a node in $A^c_{ji} \setminus A$ that did not have a neighbor in $A^c_{ij} \setminus B$ then we could have added that node to A to make A', and (2) will hold for A if and only if it holds for A'.

It is also clear that linear relaxation LP2 described in Theorem 1 is no stronger than LP1 as any solution to LP1 can be converted to a solution of LP2 by putting the weight on edge variables X_{uv} onto Y_{ujc} and Y_{vic}, where $w_{uv} = c$. This solution to LP2 will satisfy all the constraints in the theorem. Thus, we have shown LP1 and LP2 to be equivalent. □

To solve LP2, a linear program with an exponential number of constraints, we provide a separation algorithm, which finds a violated constraint, if one exists, in polynomial time.

Theorem 2. *There is an polynomial-time algorithm that can find a violated constraint in LP2 or report that none exists.*

Proof. Because each constraint in (1) involves variables of a single cost, if (1) is violated for some set Q, then Q is a subset of an A^c_{ji} for some i, j, c, and so we

can consider each subgraph $\mathcal{C}_{ij}^c$ independently. The proof of Theorem 1 shows that there is a violated constraint of the form (1) between i, j involving variables of cost c if and only if the maximum flow in C_{ij}^c is less than $y(A_{ij}^c)$. Thus, the minimum cut can be found for each triple i, j, c, and, if a triple i, j, c is found where the minimum cut is less than $w(A_{ij}^c)$, one knows that a violated constraint exists between positions i and j with $Q \subset A_{ij}^c$.

The minimum cut can then be examined to determine the violated constraint explicitly. Let $\{r\} \cup A \cup B$ be the minimum $r - s$ cut in C_{ij}^c, with $A \subseteq A_{ji}^c$ and $B \subseteq A_{ij}^c$ (following the notation of Lemma 1). Such a cut is shaded in Figure 2. Let m be the capacity of this cut, and assume, because we are considering a triple i, j, c that was identified as having a violated constraint, that $w(A_{ij}^c) > m$. For ease of notation let $\bar{A} = A_{ij}^c \setminus A$. Because $m < \infty$ there are no edges going from A to $A_{ji}^c \setminus B$, and hence two things hold: (1) $m = y(B) + y(\bar{A})$ and (2) $\mathcal{N}(A) \subseteq B$, and therefore $y(\mathcal{N}(A)) \leq y(B)$. Chaining these facts together, we have

$$y(A) = y(A_{ij}^c) - y(\bar{A}) > m - y(\bar{A}) = y(B) \geq y(N(A)),$$

Thus, the set A is a set for which the constraint of the form (1) is violated. □

In practice not all of these constraints are necessary. Some particular choices of Q_{ij} yield constraints that are intuitively very useful and are usually enough in practice. The constraints with the largest Q_{ij}, that is $Q_{ij} = A_{ij}$, were used in the proof of Theorem 1, and we have found them to be useful in practice. In fact, for this Q_{ij} inequality (1) is an equality. LP2 already includes all the constraints with $Q_{ij} = \{Y_{ujc}\} \subset A_{ij}^c$. Rather than including constraints with $1 < |Q_{ij}| < |A_{ij}^c|$, we include the constraints with i and j reversed, taking $Q_{ji} = \{Y_{vic}\} \subseteq A_{ji}^c$ for $v \in V_j$, which can be seen to be weaker versions of constraints (1) with larger Q_{ij} sets. More detail about our approach to and experiences with real problems can be found in Section 4. Examples can be constructed for which these constraints are insufficient to make LP2 as tight as LP1. For example, Fig. 3 gives a graph $\mathcal{C}_{ij}^c$ for a single color for which all the constraints (1) hold but for which no solution of IP1 can be constructed. However, we have not encountered such pathological cases in practice, and so we do not explore adding constraints with $|Q| > 1$.

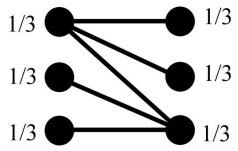


Fig. 3. Example graph $\mathcal{C}_{ij}^c$ where the added constraints are insufficient. All the constraints with $|Q| = 1$ or $|Q| = |A_{ij}|$ are satisfied, but the weights on one side cannot be matched up with weights on the other.

4 Computational Results

4.1 Methodology

We have found the following methodology to work well in practice. We first solve the LP relaxation of IP2. If the solution is integral, we are finished. Otherwise,

we add any violated constraint of the form (1) with i and j reversed and with $|Q_{ji}| = 1$, and resolve. We use the optimal basis of the previous iteration as a starting point for the next, setting the dual variables for the added constraints to be basic.

Because the variant of the simplex algorithm can make a large difference in running time in practice, LP1 was solved using two different variants. In the first (`primal dualopt`), the primal problem was solved using the dual simplex algorithm. In the second (`dual primalopt`), the dual problem was solved using the primal simplex algorithm. While, in theory, these two variants should perform similarly, in practice running times can differ significantly. Applying the dual simplex method to the dual problem or the primal simplex to the primal problem are not expected to perform as well, and small scale testing confirms this intuition. LP2 was always solved using the dual simplex method applied to the primal problem to make adding constraints faster.

The linear and integer programs were specified with Ampl and solved using CPLEX 7.1. All experiments were run on a public 1.2 GHz SPARC workstation shared by many researchers, using a single processor. All the timings reported are in CPU seconds on this machine. For any run, any problem taking longer than 5 hours was aborted.

4.2 Test Sets

We present results on identifying the binding sites of 50 transcription factor families. We construct our data set from the data of [25, 19] in a fashion similar to [20]. In short, we remove all sites for sigma-factors, duplicate sites, as well as those that could not be unambiguously located in the genome. For each transcription factor under consideration, we extract the proteins it is regulating, and gather at least 300 base pairs of genomic sequence upstream of ther transcription start sites. In those cases where the binding site is located further upstream, we extend the sequence to include the binding site. The window size for each family was chosen based on the length of the consensus binding site size, determined from other biological studies. The families, their sizes and the length of the binding site are shown in Table 1.

4.3 Performance of the LP relaxations

We solved LP1 and LP2 relaxations for the 50 problems in Table 1. The running times of LP2 are shown in Fig. 4(b). Generally, the initial set of constraints is sufficient to get a tight solution. Six problems required adding constraints to LP2 in order to make it as tight as LP1. The problems flhCD, torR, and hu required 2 cutting plane iterations, ompR required 3, oxyR required 4, and nagC required 5. Running times reported in Fig. 4(b) are the sum of the solve times of all cutting plane iterations.

Of the problems solvable in < 5 hours, only 3 were not integral. This is somewhat surprising. Of course, there is much structure to real problems, which may make them less susceptible to the worst-case analysis. The success of the

TF	ℓ	p	n	TF	ℓ	p	n	TF	ℓ	p	n	TF	ℓ	p	n
ada	31	3	810	fhlA	27	2	731	ilvY	27	2	1079	ntrC	17	5	1516
araC	48	6	1715	fis	35	18	5371	lacI	21	2	560	ompR	20	9	3057
arcA	15	13	4790	flhCD	31	3	810	lexA	20	19	5554	oxyR	39	4	1048
argR	18	17	5960	fnr	22	12	3705	lrp	25	14	4090	pdhR	17	2	568
carP	25	2	552	fruR	16	11	4082	malT	10	10	3410	phoB	22	14	4618
cpxR	15	9	2614	fur	18	9	3182	marR	24	2	813	purR	26	20	5856
cspA	20	4	1410	galR	16	7	2188	melR	18	2	717	rhaS	50	2	502
cynR	21	2	854	gcvA	20	4	1234	metJ	16	15	5754	soxS	35	13	4004
cysB	40	3	783	glpR	20	11	3829	metR	15	8	3312	torR	10	4	2198
cytR	18	5	1695	hipB	30	4	1084	modE	24	3	934	trpR	24	4	1108
dnaA	15	8	2381	hns	11	5	1485	nagC	23	6	1870	tus	23	5	1390
fadR	17	7	2122	hu	16	2	571	narL	16	10	3301	tyrR	22	17	5258
farR	10	3	873	iclR	15	2	588								

Table 1. Sizes for the 50 problems considered: number of sequences (p), motif length (ℓ), and total number of nodes in the underlying graph (n).

LP relaxations in finding integral solutions suggests that handling non-integral cases may not be as pressing a problem as one would think.

We compared the running time of LP2 with that of LP1 by taking as the running time of LP1 that of the simplex variant that performed the best. In order words, we take the speed-up factor to be:

$$\frac{\min(\texttt{primal dualopt}, \texttt{dual primalopt})}{\text{LP2}} \tag{4}$$

This gives LP1 the benefit of the doubt. In practice, always achiving the runtime used in the numerator would require running each variant in parallel using 2 processors. The speed-up factors for these problems are shown in Fig. 4(a). For 10 problems, neither simplex variant completed in < 5 hours when applied to LP1, whereas LP2 did. For these problems, the numerator of (4) was taken to be 5 hours. This gives a lower bound on the speed up. For one problem, cytR, the reverse is true and LP2 could not finish within 5 hours, while both simplex variants successfully solved it using LP1. For this problem, the denominator was taken to be 5 hours, and (4) gives an upper bound. For 5 problems, no method found a solution in < 5 hours; these are omitted from Fig. 4. Only cytR was slower using LP2, and an order of magnitude increase in speed is common when using LP2 compared with LP1.

As expected, the size of the constraint matrix (defined as the number of constraints times the number of variables) is often be smaller for LP2. Fig. 4(c) plots (size for LP2)/(size for LP1). While, in 5 cases the matrix for LP2 is larger, in many cases it is $< 50\%$ the size of the matrix for LP1. A smaller constraint matrix can often lead to faster iterations.

We also compared the motifs found by our approach to the set of known transcription factor binding sites. In all cases, we found motifs that are at least

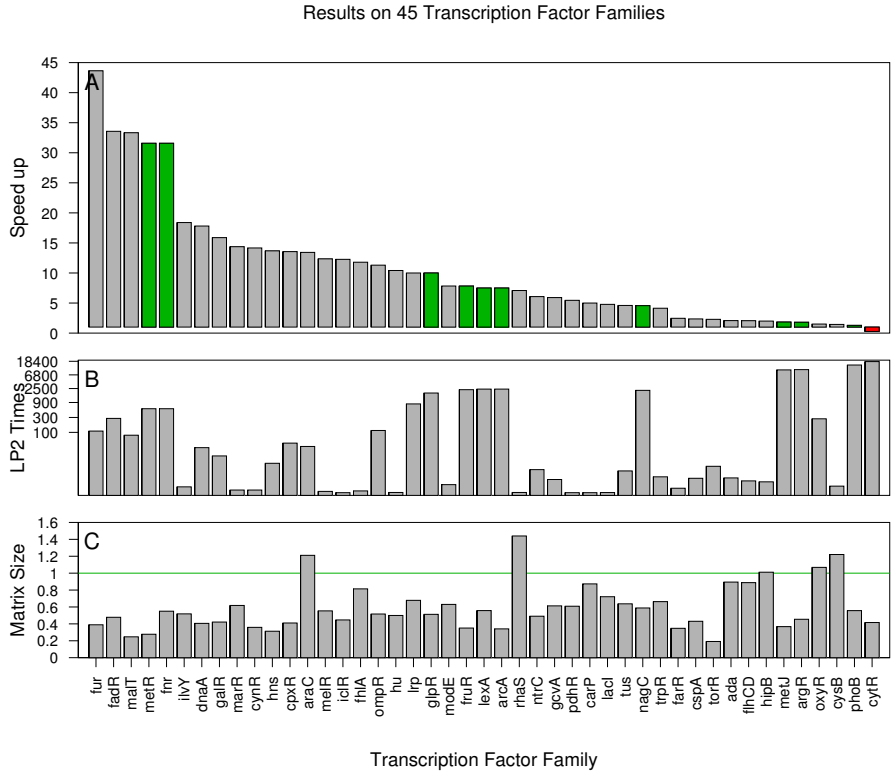


Fig. 4. (a) Speed-up factor of LP2 over LP1 as defined in 4. Shaded bars correspond to problems for which LP1 did not finish in < 5 hours. The doublely shaded bar (far right) marks the problem for which LP2 did not finish in < 5 hours, but LP1 did. (b) Running times in seconds for LP2. The y-axis is in log scale. (c) Matrix size for LP2 divided by that for LP1.

as well conserved as the actual binding sites (measured by average information content). Since our test data are real genomic sequences, co-regulated genes may in fact have multiple shared binding sites for different transcription factors.

5 Conclusions

In this paper, we introduced a novel ILP formulation of the motif finding problem that works well in practice. In particular, it is significantly faster in finding optimal solutions to motif finding instances than a previous ILP formulation introduced by [33]. We note that a variety of graph pruning and decomposition techniques have been introduced for motif finding (e.g., [23, 22]), and it is likely that, in conjunction with those techniques, our formulation will be able to tackle problems of significantly larger sizes.

There are many interesting avenues for future work. While the underlying graph problem is essentially identical to that of [5, 13], one central difference is that when minimizing distance in the motif finding application based on nucleotide matches and mismatches, the triangle inequality is satisfied. The current ILP formulations do not exploit this, and as a result, work in its absence. Another feature commonly present in motif finding that is not used here is that the edge weights in the graph are not independent, as each node represents a subsequence from a window sliding along the DNA. Incorporating either the triangle inequality or the correlation between edge weights into the ILP or its analysis may lead to further advances in computational methods for motif finding. Finally, it would also be useful to extend the basic formulation presented here to other motif finding applications, for example, to find multiple co-occuring or repeated motifs.

References

1. Akutsu, T., Arimura, H., and Shimozono, S. On approximation algorithms for local multiple alignment. RECOMB, 2000, pp. 1–7.
2. Althaus,E., Kohlbacher,O., Lenhof,H.-P. and Müller,P. A combinatorial approach to protein docking with flexible sidechains. RECOMB, 2000, pp. 15–24.
3. Bafna, V., Lawler, E., Pevzner P. A. Approximation algorithms for multiple alignment. Theoretical Computer Science, 182:233–244, 1997.
4. Bailey, T. and Elkan, C. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. Machine Learning, 21:51–80, 1995.
5. Chazelle, B., Kingsford, C., and Singh M. A semidefinite programming approach to side-chain positioning with new rounding strategies, INFORMS J. on Computing, 16:380–392, 2004.
6. Cliften, P., Sundarsanam P., Desikan, A., Fulton, L., Fulton, B., Majors, J. *et al.*. Finding functional features in Saccharomyces genomes by phylogenetic footprinting. Science, 301:71–76, 2003.
7. Cook, W.J., Cunningham, W.H., Pulleyblank, W.R., Schrijver, A. Combinatorial Optimization. Wiley-Interscience, New York, 1997.
8. Eskin, E. and Pevzner, P. Finding composite regulatory patterns in DNA sequences. Bioinformatics (Supplement 1), 18:S354–S363, 2002.
9. Frith, M.C., Hansen, U., Spouge, J.L. and Weng Z. Finding functional sequence elements by multiple local alignment. Nucleic Acids Res. 32:189–200, 2004.
10. Gusfield, D. Efficient methods for multiple sequence alignment with guaranteed error bounds. Bull. Math. Biol. 55:141–154, 1993.
11. Hertz, G. and Stormo, G. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. Bioinformatics 15:563–577, 1999.
12. Kellis, M. Patterson, N., Endrizzi, M., Birren, B. and Lander E. Sequencing and comparison of yeast species to identify genes and regulatory elements. Nature, 423: 241–254, 2003.
13. Kingsford, C.L., Chazelle, B. and Singh, M. Solving and analyzing side-chain positioning problems using linear and integer programming. Bioinformatics 21:1028–1039, 2005.
14. Lawrence, C., Altschul, S., Boguski, M., Liu, J., Neuwald, A., and Wootton, J. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. Science 262:208–214, 1993.

15. Lawrence, C. and Reilly, A. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. Proteins: Structure, Function, and Genetics, 7:41–51, 1990.
16. Lee, T.I., Rinaldi, N.J., Robert, F., Odom, D.T., Bar-Joseph, Z., Gerber, *et al.*Transcriptional regulatory network Saccharomyces cerevisiae. Science 298:799–804, 2002.
17. Liu, X., Brutlag, D.L., Liu, J.S. BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. PSB, 2001, pp. 127–138.
18. Marsan L., Sagot M.F. Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification. J. Comp. Bio. 7:345-362, 2000.
19. McGuire, A., Hughes, J., and Church, G. Conservation of DNA regulatory motifs and discovery of new motifs in microbial genomes. Genome Res. 10:744–757, 2000.
20. Osada, R., Zaslavsky, E., and Singh, M. Comparative analysis of methods for representing and searching for transcription factor binding sites. Bioinformatics 20:3516–3525, 2004.
21. Pavesi, G., Mereghetti, P., Mauri, G. and Pesole. G. Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. Nucl. Acids Res. 32: W199–W203, 2004.
22. Pevzner, P. and Sze, S. Combinatorial approaches to finding subtle signals in DNA sequences. ISMB, 2000, pp. 269–278.
23. Reinert, K., Lenhof, H.P., Mutzel, P., Mehlhorn, K., and Kececioglu, J. A branch-and-cut algorithm for multiple sequence alignment. RECOMB, 1997, pp. 241–249.
24. Rigoutsos, I., and Floratos, A. Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm. Bioinformatics 14:55–67, 1998.
25. Robison, K. and McGuire, A. M. and Church, G. M. A comprehensive library of DNA-binding site matrices for 55 proteins applied to the complete *Escherichia coli* K-12 Genome. J. Mol. Biol. 284:241–254, 1998.
26. Schuler, G., Altschul, S., and Lipman, D. A workbench for multiple alignment construction and analysis. Proteins 9(3):180–190, 1991.
27. Sinha, S. and Tompa, M. YMF: A program for discovery of novel transcription factor binding sites by statistical overrepresentation. Nucl. Acids Res. 31:3586-3588, 2003.
28. Sze, S.-H., Lu, S. and Chen, J. Integrating sample-driven and pattern-driven approaches in motif finding. WABI, 2004, pp. 438–449.
29. Tavazoie, S., Hughes, J. D, Campbell, M. J., Cho, R. J., Church, G.M. Systematic determination of genetic network architecture. Nature Genetics 22(3):281–285, 1999.
30. Tompa, M. An exact method for finding short motifs in sequences, with application to the ribosome binding site problem. ISMB, 1999, pp. 262–271.
31. Tompa, M., Li, N., Bailey, T. L., Church, G.M., De Moor, B., Eskin, E., *et al.*Assessing computational tools for the discovery of transcription factor binding sites. Nature Biotech. 23:137–144, 2005.
32. van Helden, J., Rios, A.F. and Collado-Vides, J. Discovering regulatory elements in non-coding sequences by analysis of spaced dyads. Nucleic Acids Res. 28:1808–1818, 2000.
33. Zaslavsky, E. and Singh, M. Combinatorial Optimization Approaches to Motif Finding. Manuscript, submitted for publication, 2005.