A branch-and-cut approach to physical mapping of chromosomes by unique end-probes

Thomas Christof* Michael Jünger† John Kececioglu‡
Petra Mutzel§ Gerhard Reinelt*

April 3, 1997

Abstract A fundamental problem in computational biology is the construction of physical maps of chromosomes from hybridization experiments between unique probes and clones of chromosome fragments in the presence of error. Alizadeh, Karp, Weisser and Zweig (*Algorithmica* 13:1/2, 52–76, 1995) first considered a maximum-likelihood model of the problem that is equivalent to finding an ordering of the probes that minimizes a weighted sum of errors, and developed several effective heuristics. We show that by exploiting information about the end-probes of clones, this model can be formulated as a weighted Betweenness Problem. This affords the significant advantage of allowing the well-developed tools of integer linear-programming and branch-and-cut algorithms to be brought to bear on physical mapping, enabling us for the first time to solve small mapping instances to optimality even in the presence of high error. We also show that by combining the optimal solution of many small overlapping Betweenness Problems, one can effectively screen errors from larger instances, and solve the edited instance to optimality as a Hamming-Distance Traveling Salesman Problem. This suggests a new approach, a Betweenness-Traveling Salesman hybrid, for constructing physical maps.

Keywords Computational biology, physical mapping of chromosomes, betweenness problem, linear ordering problem, branch-and-cut

1 Introduction

Each human chromosome is a linear sequence of roughly 10^8 bases. To aid manipulation of DNA of this scale in the laboratory, and to prepare it for sequencing, *physical maps* of chromosomes are constructed that give the location along the molecule of important features, such as the location of clones of DNA fragments.

*Institut für Angewandte Mathematik, Universität Heidelberg, Germany.
Email: thomas.christof@iwr.uni-heidelberg.de.
†Institut für Informatik, Universität zu Köln, Germany. Email: mjuenger@informatik.uni-koeln.de.
‡Department of Computer Science, University of Georgia, Athens, GA 30602, USA.
Email: kece@cs.uga.edu.
§Max-Planck-Institut für Informatik, Saarbrücken, Germany. Email: mutzel@mpi-sb.mpg.de.

In this paper, we consider the construction of physical maps by a protocol known as *STS-content mapping*. In this strategy, which is widely used in the Human Genome Project, each *clone* corresponds to an interval of the chromosome, and each *probe* corresponds to a unique point on the chromosome. While the position of clones and probes along the chromosome is unknown, it can be determined whether a clone contains a probe by a test called a *hybridization* experiment. (The experiment tests whether the probe DNA bonds, or *hybridizes*, with the clone DNA.) As hybridization experiments are inevitably imperfect, the resulting clone-probe incident data contains errors. In a *false positive* error, the experiment reports that a clone contains a probe when it does not, while in a *false negative* error, the experiment reports that a clone does not contain a probe when it does. The goal is to recover the probe or clone ordering from such hybridization data.

We consider the problem of STS-content mapping with false positive and false negative errors. In practice, an additional type of error, called *chimerism*, occurs. A chimeric clone is a clone that does not sample a single interval of the chromosome, but contains two or more unrelated fragments of DNA. While we concentrate on handling false positive and negative errors, our approach can also be extended to data with chimeric clones, as indicated in the final section.

In practice, the set of probes is obtained by extracting DNA from the ends of clones. Usually probes are obtained from a subset of the clones, and are not consistently extracted from both ends. For any given clone, however, the probes that were extracted from its ends are known, and can be identified.

In this paper, we assume that probes are extracted from *both* ends of *every* clone. (We do not assume any information about which end a probe came from.) We show that by consistently selecting probes from both ends of a clone, the problem of reconstructing the probe order in the presence of false positives and negatives can be successfully tackled by integer linear programming. Our computational results suggest that this change to the experimental protocol could significantly improve the quality of the physical maps that are constructed, while tolerating a much higher experimental error rate. Furthermore, our integer linear programming formulation, which is based on the Weighted Betweenness Problem, can handle partial-order information on probes, and is actually simplified by such information. Our approach can also be extended to general probe-clone hybridization data where clones do not have probes extracted from both ends, as indicated in the final section.

1.1 Related work

We use the following mathematical description of the problem, also used by Alizadeh, Karp, Weisser and Zweig [AKWZ94] and Greenberg and Istrail [GI95]. The chromosome is mapped by n probes, $P_1, \ldots, P_n$, and m clones, $C_1, \ldots, C_m$. The outcome of the probe-clone hybridization experiments is given by an m by n 0-1 matrix A whose rows correspond to clones and whose columns correspond to probes. An entry a_{ij} is 1 if probe P_j hybridizes to clone C_i, and 0 otherwise. Entry $a_{ij} = 1$ is a false positive if the entry should be 0, while entry $a_{ij} = 0$ is a false negative if it should be 1.

In the absence of error and knowledge of end-probes, the problem of reconstructing the probe order is equivalent to the Consecutive Ones Problem: find a permutation π of the columns of matrix A so that in the reordered matrix A^π, the ones in every row are consecutive. Using the PQ-tree of Booth and Lueker [BL76], a representation of all permutations π that have the consecutive ones property can be computed in time linear in the number of ones in matrix A. In the presence of error, however, this approach breaks down.

Biologists and computer scientists have tackled the problem in several ways. Typically, maximum-likelihood functions are suggested and their solution is attempted by local search (see [AKWZ94, GHL$^+$93]). Another approach is to approximate the maximum-likelihood function by a well-studied combinatorial problem, such as the Hamming-Distance Traveling Salesman Problem ([AKWZ94, AKNW95, GI95]). Methods for filtering the data have also been offered as an attempt to remove typical errors such as false positives or chimeric clones [CDGH95, GLM94]. Further pointers to the literature are given in [MLV97]. It is worth emphasizing that all these approaches are particularly sensitive to false-positive errors, and are successful only for relatively low false-positive rates.

Alizadeh, Karp, Weisser, and Zweig [AKWZ94] first introduced the maximum-likelihood model that we consider. The idea is to find a corrected matrix B that maximizes $p(B|A)$, the probability that B is the true hybridization matrix, given the observed matrix A. Using Bayes' Theorem, they show

$$\underset{B}{\operatorname{argmax}}\{p(B|A)\} \quad = \quad \underset{B}{\operatorname{argmax}}\left\{\left(\frac{p_\chi}{1-p_\chi}\right)^{n_\chi(A|B)}\left(\frac{p_\rho}{1-p_\eta}\right)^{n_\rho(A|B)}\left(\frac{p_\eta}{1-p_\rho}\right)^{n_\eta(A|B)}\right\},$$

where $n_\chi(A|B)$ is the number of rows of A that are declared chimeric with respect to B, $n_\rho(A|B)$ is the number of false positives in A with respect to B, $n_\eta(A|B)$ is the number of false negatives in A with respect to B, p_χ is the probability that a clone is chimeric, p_ρ is the probability that an entry is a false positive, and p_η is the probability that an entry is a false negative. Choosing the constants $c_\chi = -\ln\frac{p_\chi}{1-p_\chi}$, $c_\rho = -\ln\frac{p_\rho}{1-p_\eta}$, and $c_\eta = -\ln\frac{p_\eta}{1-p_\rho}$ gives

$$\underset{B}{\operatorname{argmax}}\Big\{p(B|A)\Big\} \quad = \quad \underset{B}{\operatorname{argmin}}\Big\{c_\chi\, n_\chi(A|B) \;+\; c_\rho\, n_\rho(A|B) \;+\; c_\eta\, n_\eta(A|B)\Big\}.$$

The problem then is to find a probe ordering π that minimizes the linear objective function

$$f(\pi) \quad = \quad \min_B \Big\{c_\chi\, n_\chi(A^\pi|B) \;+\; c_\rho\, n_\rho(A^\pi|B) \;+\; c_\eta\, n_\eta(A^\pi|B)\Big\},$$

where A^π is A with columns reordered by π. Evaluating f for a given π requires finding the best matrix B; Jain and Myers [JM95] show how this can be done efficiently using dynamic programming. Given the NP-hardness of minimizing $f(\pi)$ [Boo75], Alizadeh et al. attack the problem using local search.

1.2 Plan of the paper

We model mapping with end-probes, in the absence of chimeric clones, as the *Weighted Betweenness Problem*. Given a collection of betweenness and nonbetweenness constraints on a set of elements to be linearly ordered, the Weighted Betweenness Problem asks for an ordering π of the elements that minimizes a weighted sum of the constraints violated by π (see Section 2). We then give an integer linear programming formulation of Weighted Betweenness based on the Linear Ordering Problem. Section 3 describes a branch-and-cut approach to solving Weighted Betweenness. A straightforward application of this approach showed only limited success; Section 4 discusses the necessary improvements that lead to better linear programming formulations. Our experiments on generated data verify that, under the maximum-likelihood objective, the correct order is an optimal solution, or very close to an optimal solution. Conversely, our exact solution of Weighted Betweenness gives, in most cases, the correct probe ordering (see Section 5). Section 6 describes our plans for further research, and indicates how our approach can be extended to handle chimerism as well as the absence of end-probe information.

2 Reducing physical mapping with end-probes to integer linear programming

We formulate the problem of minimizing $f(\pi)$ in the absence of chimerism in terms of the *Weighted Betweenness Problem.* Consider the r-th row of A corresponding to clone C_r. Let columns i and k correspond to the end-probes P_i and P_k of clone C_r. Consider a probe P_j, other than P_i or P_k, that hybridizes to clone C_r, i.e. where $a_{rj} = 1$. If this entry a_{rj} is correct, then in the correct order π of the probes, column j should be between columns i and k. We denote this requirment by the triple (i, j, k), which we call a *betweenness constraint.* Since we do not know the relative order of end-probes, both the ordering $i \cdots j \cdots k$ and $k \cdots j \cdots i$ are consistent with betweenness constraint (i, j, k).

Now consider a probe P_j, again different from P_i and P_k, that did *not* hybridize with clone C_r, i.e. where $a_{rj} = 0$. If entry a_{rj} is correct, then in the correct order π of the probes, column j should *not* be between columns i and k, which we denote by the triple $\overline{(i, j, k)}$, and call a *nonbetweenness constraint.* Both the orderings $k \cdots i \cdots j$, $i \cdots k \cdots j$, and their reverse, are all consistent with the constraint $\overline{(i, j, k)}$. For an instance of the problem, we denote the set of betweenness constraints by $\mathcal{B}$, and the set of nonbetweenness constraints by $\overline{\mathcal{B}}$. Notice that any triple i, j, k occurs only once in $\mathcal{B}$ or $\overline{\mathcal{B}}$.

The key advantage of having end-probe information is that we can express the number of false positives and false negatives in a probe ordering π, which are *global* properties of π usually requiring dynamic programming to compute [JM95, AKWZ94], in terms of *local* betweenness and nonbetweenness constraints. The false positives in row r are exactly those columns j for which $a_{rj} = 1$ but constraint (i, j, k) is violated by π, where P_i and P_k are the end-probes of C_r; each such j has a cost of c_ρ in the objective function $f(\pi)$. The false negatives in row r are exactly those columns j for which $a_{rj} = 0$ but constraint $\overline{(i, j, k)}$ is violated; each such j has a cost of c_η. Thus the problem is equivalent to finding a π that minimizes the weighted sum of the constraints that it violates. We note that this formulation is slightly more general than the classical Betweenness Problem [Opa79], in that we have both betweenness and nonbetweenness constraints, and we are optimizing a weighted sum of violations.

Opatrny [Opa79] has shown that simply deciding whether a set of elements can be linearly ordered to satisfy a collection of betweenness constraints is NP-complete. Chor and Sudan [CS95] present an approximation algorithm for the classical Betweenness Problem that either finds a feasible solution, or finds a linear order that satisfies at least one-half of the constraints. They do not consider nonbetweenness constraints or weighted constraints.

2.1 Variables

Our formulation of the Weighted Betweenness Problem can be expressed as an integer linear program by introducing linear ordering variables. For every ordered pair (i, j) of probes, we introduce a variable x_{ij} with the interpretation

$$x_{ij} = \begin{cases} 1 & \text{if } i \text{ precedes } j \text{ in the ordering } \pi^{-1}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus the probe order π is represented by the set of variables x_{ij}.

With every betweenness constraint (i, j, k) from set $\mathcal{B}$, we associate a variable b_{ijk} with

$$b_{ijk} = \begin{cases} 1 & \text{if constraint } (i, j, k) \text{ is met,} \\ 0 & \text{otherwise.} \end{cases}$$

Thus $1 - b_{ijk}$ counts whether betweenness constraint (i, j, k) is violated.

As it will turn out, it is more natural in the linear programming formulation to deal with betweenness constraints rather than nonbetweenness constraints; hence we will express nonbetweenness constraints in terms of betweenness constraints. With every nonbetweenness constraint $\overline{(i, j, k)}$ from set $\overline{\mathcal{B}}$, we associate a variable b_{ijk} with

$$b_{ijk} = \begin{cases} 1 & \text{if constraint } (i, j, k) \text{ is met,} \\ 0 & \text{otherwise.} \end{cases}$$

Thus b_{ijk} counts whether $\overline{(i, j, k)}$ is violated.

Note that this notation for b-variables is not ambiguous, as $\mathcal{B}$ and $\overline{\mathcal{B}}$ have no triples in common. In the following, when we write $b_{ijk} \in \mathcal{B}$, we mean that $(i, j, k) \in \mathcal{B}$; similarly, when we write $b_{ijk} \in \overline{\mathcal{B}}$, we mean $\overline{(i, j, k)} \in \overline{\mathcal{B}}$.

2.2 Constraints

To ensure that the variables x_{ij} encode a linear ordering π of the probes, the following conditions must be met:

$$0 \le x_{ij} \le 1 \quad \text{for} \quad 1 \le i \ne j \le n \tag{1}$$
$$x_{ij} + x_{ji} = 1 \quad \text{for} \quad 1 \le i \ne j \le n \tag{2}$$
$$x_{ij} + x_{jk} + x_{ki} \le 2 \quad \text{for} \quad 1 \le i \ne j \ne k \le n \tag{3}$$
$$x_{ij} \text{ integral} \quad \text{for} \quad 1 \le i \ne j \le n. \tag{4}$$

Conditions (2) and (3) enforce the antisymmetry and transitivity properties of a linear ordering. The inequalities in (3) are sometimes called the *3-dicycle inequalities.*

With these conditions, every feasible assignment of the x-variables corresponds to a linear ordering π of the probes. To ensure that the b_{ijk} count violations of the betweenness and nonbetweenness constraints, we add the following inequalities. To force a $b_{ijk} \in \mathcal{B}$ to be 0 if (i, j, k) is violated, we add

$$b_{ijk} \le x_{jk} + x_{ji} \tag{5}$$
$$b_{ijk} \le x_{ij} + x_{kj}. \tag{6}$$

Thus $b_{ijk} \in \mathcal{B}$ is 0 if $x_{ij} \ne x_{jk}$.

To force a $b_{ijk} \in \overline{\mathcal{B}}$ to be 1 if $\overline{(i, j, k)}$ is violated, we add

$$b_{ijk} \ge -x_{ij} - x_{jk} + 1 \tag{7}$$
$$b_{ijk} \ge x_{ij} + x_{jk} - 1. \tag{8}$$

Thus $b_{ijk} \in \overline{\mathcal{B}}$ is 1 if $x_{ij} = x_{jk}$.

Due to the form of the objective function (see below), we do not have to explicitly require integrality of the b_{ijk}. The coefficient on a $b_{ijk} \in \mathcal{B}$ in the objective function is negative, while the coefficient on a $b_{ijk} \in \overline{\mathcal{B}}$ is positive, so it suffices to require

$$0 \leq b_{ijk} \leq 1. \tag{9}$$

2.3 Objective

Finally, we seek an assignment of the variables that minimizes

$$c_\eta \sum_{b_{ijk} \in \overline{\mathcal{B}}} b_{ijk} \; + \; c_\rho \sum_{b_{ijk} \in \mathcal{B}} (1 - b_{ijk}) \;\; =$$

$$c_\eta \sum_{b_{ijk} \in \overline{\mathcal{B}}} b_{ijk} \; - \; c_\rho \sum_{b_{ijk} \in \mathcal{B}} b_{ijk} \; + \; c_\rho |\mathcal{B}|,$$

where c_η and c_ρ are the false negative and false positive costs from Section 1.1. Since $c_\rho|\mathcal{B}|$ is a constant for any particular problem instance, this is equivalent to minimizing the objective function

$$c_\eta \sum_{b_{ijk} \in \overline{\mathcal{B}}} b_{ijk} \; - \; c_\rho \sum_{b_{ijk} \in \mathcal{B}} b_{ijk}.$$

Constraints (1) to (9) guarantee that the solutions are precisely all possible linear orderings (represented by variables x_{ij}) and that the number of violated betweenness and nonbetweenness constraints are counted correctly (through variables b_{ijk}).

3 Solving the integer program by branch-and-cut

The *integer linear program* formulated in the previous section (which we call the ILP), consists of $\Theta(n^3)$ constraints in $\Theta(n^2)$ variables, where n is the number of probes (equal to twice the number of clones). Therefore, even for relatively small values of n, it is not feasible to generate this ILP completely and solve it by a commercial mixed-integer linear-programming package.

Instead, our algorithm for solving the ILP proceeds as follows. We first tackle the *canonical linear-programming relaxation* of the ILP (which we call the LP). This relaxation is obtained by dropping the integrality requirements (4).

To this end, note that by using the equation $x_{ij} + x_{ji} = 1$, we can define a standard form for the inequalities in which all coefficients of the linear ordering variables are nonnegative. We also use this equation to eliminate variables x_{ij} with $j > i$, which halves the number of such variables. This elimination converts the 3-dicycle inequalities (3) into the inequalities $x_{ij} + x_{jk} - x_{ik} \leq 1$ and $x_{ij} + x_{jk} - x_{ik} \geq 0$ where $i < j < k$.

Because of the large number of constraints, the LP is solved using a cutting plane approach. To this end we start with the linear program consisting only of constraints (1), (2), and (9), and solve it to optimality. If the optimal solution violates any of constraints (3), (5)–(8), we add the violated inequalities and reoptimize. This approach generates inequalities only when needed and leads to substantially smaller linear programs. When, after several iterations, no more inequalities are violated, we have found the optimal solution of the canonical relaxation. If this solution is integral, then we have solved the weighted betweenness problem and have provably found an optimal physical map. If in the solution some variables have fractional values, the same procedure is applied recursively to two subproblems, one in which a fractional variable x_{ij} is set to 1, and the other in which the variable is set to 0. This approach to solving integer programs is commonly called *branch-and-cut*.

Note that for every solution of the betweenness problem, corresponding to a permutation $\pi = (\pi_1 \pi_2 \cdots \pi_n)$, the reverse permutation $(\pi_n \cdots \pi_2 \pi_1)$ is also a solution for the same objective function. This allows us to fix one arbitrary variable x_{ij} at 1 before solving the initial relaxation. This is important in practice in order to break symmetries.

Our physical-mapping algorithm is based on the branch-and-cut algorithm of Grötschel, Jünger, and Reinelt [GJR84, GJR85] for the classical Linear Ordering Problem. Reimplementation was done using the software system ABACUS of Jünger, Reinelt, and Thienel [Thi95, JRT95], a general, object-oriented framework for implementing branch-and-cut algorithms. The object-oriented design of ABACUS in C++ proved to be extremely useful. Since Linear Ordering and Betweenness are strongly related, functions and data structures common to both could be implemented in common base-classes.

In practice, however, only small instances could be solved to optimality with this implementation. The gap between the solution values of the ILP and its relaxation is rather large, so that the weak bounds obtained from the relaxation did not permit early termination of the branch-and-cut procedure. The next section discusses how this problem was overcome.

4 Tightening the linear programming relaxation

We now describe how additional valid inequalities were obtained to improve the linear programming relaxation. To simplify the notation, we refer to all betweenness conditions through set $\mathcal{B}$ alone. (Every betweeness variable corresponds to a betweenness constraint in $\mathcal{B}$.)

Suppose then that we are given an instance of Weighted Betweenness described by a number n of probes and a set $\mathcal{B}$ of t triples. With every feasible solution to the problem, corresponding to a permutation $\pi = (\pi_1\ \pi_2\ \cdots\ \pi_n)$ of the probes, we associate 0-1 vectors $x^\pi \in \{0,1\}^{n(n-1)}$ and $b^\pi \in \{0,1\}^t$ with

$$x_{ij}^\pi = \begin{cases} 1 & \text{if} \quad \pi_i^{-1} < \pi_j^{-1} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$b_{ijk}^\pi = \begin{cases} 1 & \text{if} \quad \pi_i^{-1} < \pi_j^{-1} < \pi_k^{-1} \quad \text{or} \quad \pi_k^{-1} < \pi_j^{-1} < \pi_i^{-1} \\ 0 & \text{otherwise.} \end{cases}$$

The *polytope* $\mathcal{P}(n, \mathcal{B})$ associated with the instance of the betweenness problem is the convex hull of all posible vectors $\binom{x^\pi}{b^\pi}$,

$$\mathcal{P}(n,\mathcal{B}) = \operatorname{conv}\left(\left\{ \begin{pmatrix} x^\pi \\ b^\pi \end{pmatrix} \,\middle|\, \pi \text{ is a permutation of the probes} \right\}\right).$$

The vertices of $\mathcal{P}(n, \mathcal{B})$ are the 0-1 vectors, and correspond precisely to the feasible solutions of Weighted Betweenness.

The problem with the approach of the previous section is that the set of feasible solutions of the canonical linear programming relaxation is much larger than $\mathcal{P}(n, \mathcal{B})$. What is needed is a complete description of $\mathcal{P}(n, \mathcal{B})$ in terms of linear inequalities. The NP-hardness of Weighted Betweenness, however, makes it unlikely that such a complete linear description can be found in general and exploited algorithmically. We therefore concentrate on partial descriptions of $\mathcal{P}(n, \mathcal{B})$ that improve the linear programming relaxation.

4.1 Exploiting linear descriptions of small instances

A useful observation is that inequalities derived for small polytopes can be applied to larger polytopes. In particular, let $a^T x \leq b$ be a valid inequality for $\mathcal{P}(n, \mathcal{B})$, so that $\mathcal{P}(n, \mathcal{B}) \subseteq \{x \mid a^T x \leq b\}$. Then inequality $a^T x \leq b$ is also valid for any $\mathcal{P}(n', \mathcal{B}')$ with $n \leq n'$ and $\mathcal{B} \subseteq \mathcal{B}'$. Thus linear descriptions of small problem instances yield valid inequalities for larger problem instances. This technique is sometimes called *trivial lifting.*

We can compute the complete linear description of small problem instances by enumerating all feasible 0-1 vectors and applying the so-called double-description method to obtain the linear description of the convex hull of these vectors (see Christof and Reinelt [CR96]). We present here the linear inequalities for those polytopes that were useful in our branch-and-cut algorithm. The polytopes are associated with problem instances on 3 and 4 probes and certain combinations of betweenness constraints. We do not list the trivial inequalities $x_{ij} \leq 1$ and $x_{ij} \geq 0$, or the 3-dicycle inequalities $x_{ij} + x_{jk} + x_{ki} \leq 2$, as they define facets of the polytope only in special cases.

4.1.1 $\mathcal{P}(3, \{(1,2,3)\})$

$$
\begin{aligned}
x_{12} + x_{23} \quad & - b_{123} && \leq 1 && (10)\\
x_{21} + x_{32} \quad & - b_{123} && \leq 1 && (11)\\
x_{12} + x_{32} \quad & + b_{123} && \leq 2 && (12)\\
x_{21} + x_{23} \quad & + b_{123} && \leq 2 && (13)\\
x_{12} + x_{23} + 2x_{31} \quad & + b_{123} && \leq 3 && (14)\\
2x_{13} + x_{21} + x_{32} \quad & + b_{123} && \leq 3 && (15)
\end{aligned}
$$

Note that inequality (14) implies the 3-dicycle inequality $x_{12} + x_{23} + x_{31} \leq 2$, and also forces $b_{123} = 0$ if probe 2 is before 3 and probe 3 is before 1 in an ordering.

In the following, we do not list facets that are already given by $\mathcal{P}(3, \{(1,2,3)\})$.

4.1.2 $\mathcal{P}(4, \{(1,2,3),(1,4,3)\})$

$$
\begin{aligned}
x_{12} + 2x_{24} + x_{32} + x_{41} + x_{43} \quad & + b_{123} - b_{143} && \leq 4 && (16)\\
x_{12} + 2x_{24} + x_{32} + x_{41} + x_{43} \quad & - b_{123} + b_{143} && \leq 4 && (17)\\
x_{14} + x_{21} + x_{23} + x_{34} + 2x_{42} \quad & + b_{123} - b_{143} && \leq 4 && (18)\\
x_{14} + x_{21} + x_{23} + x_{34} + 2x_{42} \quad & - b_{123} + b_{143} && \leq 4 && (19)
\end{aligned}
$$

4.1.3 $\mathcal{P}(4, \{(1,2,3),(2,1,4)\})$

$$
x_{14} + x_{32} + 2x_{43} \quad + b_{123} + b_{214} \quad \leq 4 \qquad (20)
$$

$$\begin{aligned}
x_{23} + 2x_{34} + x_{41} \quad &+ b_{123} + b_{214} &\le 4 \qquad (21)\\
x_{14} + x_{23} + 2x_{34} + 2x_{42} \quad &+ b_{123} + b_{214} &\le 5 \qquad (22)\\
x_{14} + x_{23} + 2x_{31} + 2x_{43} \quad &+ b_{123} + b_{214} &\le 5 \qquad (23)\\
2x_{13} + x_{32} + 2x_{34} + x_{41} \quad &+ b_{123} + b_{214} &\le 5 \qquad (24)\\
2x_{24} + x_{32} + x_{41} + 2x_{43} \quad &+ b_{123} + b_{214} &\le 5 \qquad (25)
\end{aligned}$$

4.1.4 $\mathcal{P}(4, \{(1,2,3),(2,1,4),(1,4,3)\})$

$$\begin{aligned}
x_{32} + x_{43} \quad &+ b_{123} + b_{214} + b_{143} &\le 3 \qquad (26)\\
x_{23} + x_{34} \quad &+ b_{123} + b_{214} + b_{143} &\le 3 \qquad (27)\\
x_{23} + 2x_{31} + x_{43} \quad &+ b_{123} + b_{214} + b_{143} &\le 4 \qquad (28)\\
2x_{13} + x_{32} + x_{34} \quad &+ b_{123} + b_{214} + b_{143} &\le 4 \qquad (29)\\
x_{12} + 2x_{24} + x_{32} + x_{41} + x_{43} \quad &+ b_{123} - b_{143} &\le 4 \qquad (30)\\
x_{12} + 2x_{24} + x_{32} + x_{41} + x_{43} \quad &- b_{123} + b_{143} &\le 4 \qquad (31)\\
x_{14} + x_{21} + x_{23} + x_{34} + 2x_{42} \quad &+ b_{123} - b_{143} &\le 4 \qquad (32)\\
x_{14} + x_{21} + x_{23} + x_{34} + 2x_{42} \quad &- b_{123} + b_{143} &\le 4 \qquad (33)\\
x_{12} + 2x_{24} + 2x_{32} + x_{43} \quad &+ b_{214} + b_{143} &\le 5 \qquad (34)\\
x_{14} + x_{23} + 2x_{34} + 2x_{42} \quad &+ b_{123} + b_{214} &\le 5 \qquad (35)\\
x_{21} + 2x_{23} + x_{34} + 2x_{42} \quad &+ b_{214} + b_{143} &\le 5 \qquad (36)\\
2x_{24} + x_{32} + x_{41} + 2x_{43} \quad &+ b_{123} + b_{214} &\le 5 \qquad (37)
\end{aligned}$$

Note that inequalities (34)–(37) and (30)–(33) are inequalities (22)–(25) and (16)–(19) with different variable names.

4.1.5 $\mathcal{P}(4, \{(1,2,3),(2,1,4),(1,4,3),(2,3,4)\})$

$$\begin{aligned}
+b_{123} - b_{214} - b_{143} - b_{234} &\le 0 \qquad (38)\\
-b_{123} + b_{214} - b_{143} - b_{234} &\le 0 \qquad (39)\\
-b_{123} - b_{214} + b_{143} - b_{234} &\le 0 \qquad (40)\\
-b_{123} - b_{214} - b_{143} + b_{234} &\le 0 \qquad (41)\\
+b_{123} + b_{214} + b_{143} + b_{234} &\le 2 \qquad (42)
\end{aligned}$$

In addition, inequalities (28)–(29) and (16)–(19) define facets of this polytope.

4.2 Incorporating the new inequalities

The process of identifying an inequality from the complete linear description that is violated by a fractional solution of the current linear programming relaxation is called *separation.* Our separation routine for the above inequalities proceeds as follows.

To find an inequality of type (10)–(42) that is violated by the current solution of the linear program, we enumerate all legal assignments from the 3 or 4 probes of the inequality to the probes of the problem instance. Due to symmetries in the inequalities, we exercise some care to ensure that only combinatorially distinct violated inequalities are generated.

Use of these inequalities derived from low-dimensional polytopes lead to much tighter linear programming relaxations, and made it possible to obtain the computational results that we describe next.

5 Computational results

5.1 Generator

For our computational experiments we used a generator similar to the one suggested by Greenberg and Istrail [GI95]. The distinct advantage of using simulated data is that we know the correct order of the probes, and hence we can reliably measure the success of our method.

For testing our algorithm, we generated data where each clone gave rise to two end-probes, all clones were of the same length, and where clones were randomly distributed across the chromosome. We used a coverage varying from 3 to 5, a false negative rate of 10%, and a false positive rate varying from 0% to 5%. To generate a false positive or false negative, a coin was flipped at each entry with the given probability. Across experiments with varying coverage, the clone length was held constant.

5.2 Exact solution by branch-and-cut

For comparison, we also considered the common approach to physical mapping of solving a *Hamming-Distance Traveling Salesman Problem* (HDTSP) (see for example [AKWZ94] and [GI95]). The nodes of a HDTSP are the columns of our hybridization matrix A, and the distances $d_{ij} = d_{ji}$ are given by the Hamming distance between the corresponding columns, i.e., the number of rows of A with $a_{ri} \neq a_{rj}$. To obtain a Hamilton cycle problem, an artificial column is introduced in A with all entries equal to 0. The resulting symmetric Traveling Salesman Problems were solved to optimality using the ABACUS version [Thi95] of the branch-and-cut code of Jünger, Reinelt, and Thienel [JRT94].

Figures 1 to 4 show results from 150 problem instances. Within each plot, the horizontal axis indicates the false positive rate. The three plots in each figure are for coverage 3, 4, and 5.

We first compared the solution values of the Weighted Betweenness Problem and the HDTSP with the value of the known correct solution, to determine whether exact solution of the maximum likelihood model is worthwhile for recovering the true map. As shown in Figure 1, the values of the optimal solution of the betweenness problem and the correct solution differ in only a fraction of the instances, and then by quite small amounts. This suggests that the maximum-likelihood function may be useful in identifying the correct probe order, even on instances with high false positive rates. In contrast, the optimal solution of the HDTSP is rarely an optimal solution for the original problem. Moreover, the discrepancy increases with higher false positive rates, while the betweenness solution remains stable.

As in Greenberg and Istrail [GI95], we define the adjacency-quality of a solution π as the ratio of the number of adjacencies of probes that are common to both π and the correct ordering π^*, divided by the number of adjacencies in π^*. Figure 2 shows that the quality of the solution of the betweenness problem is 1 in many cases, indicating that the optimal solution of the betweenness problem is exactly the correct ordering. Another measure of the quality of a solution is the average number of probes between two probes that are adjacent in the correct ordering. Figure 3 plots this measure. With respect to both measures, the quality

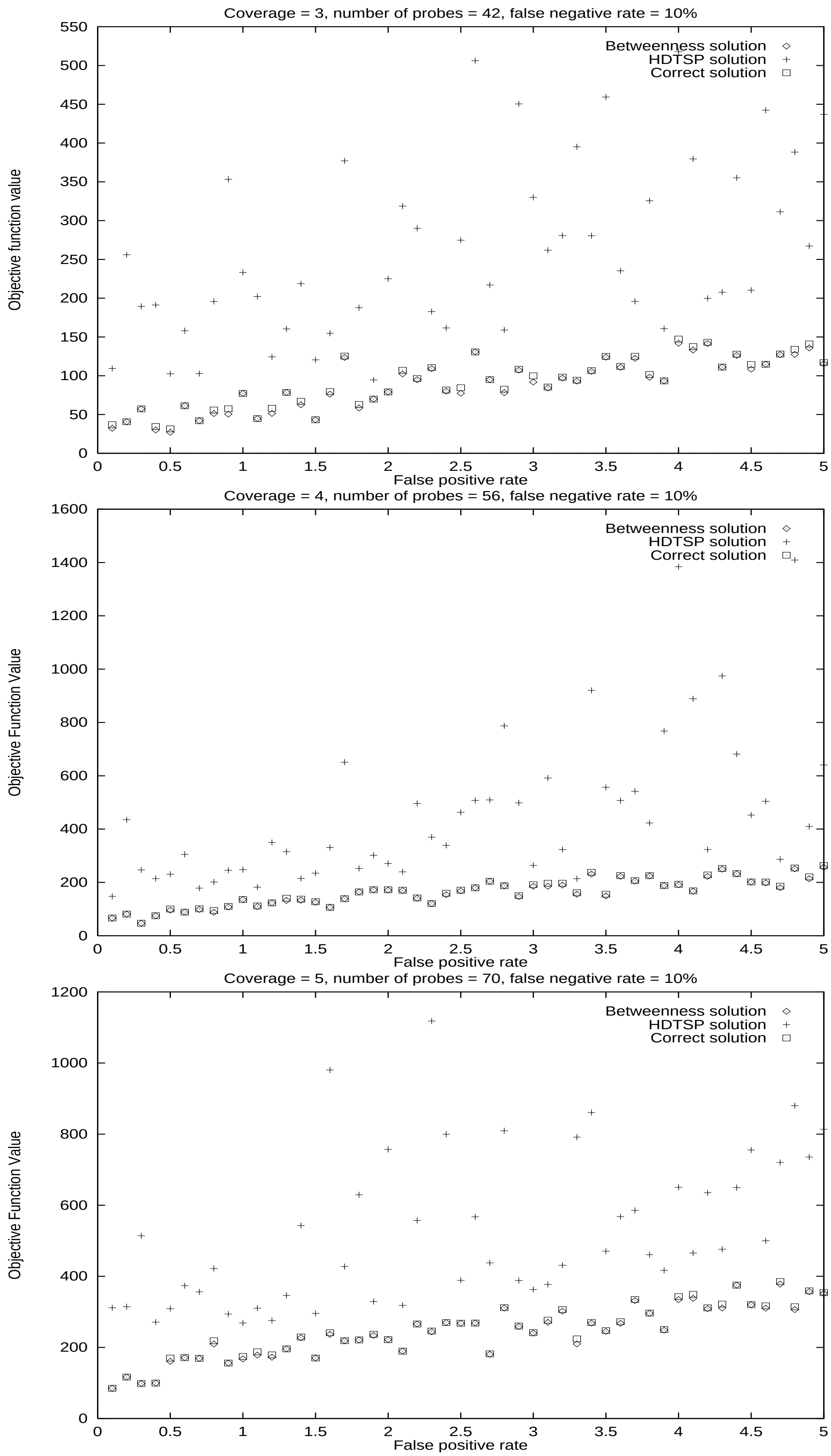

Figure 1 Optimal solution values.

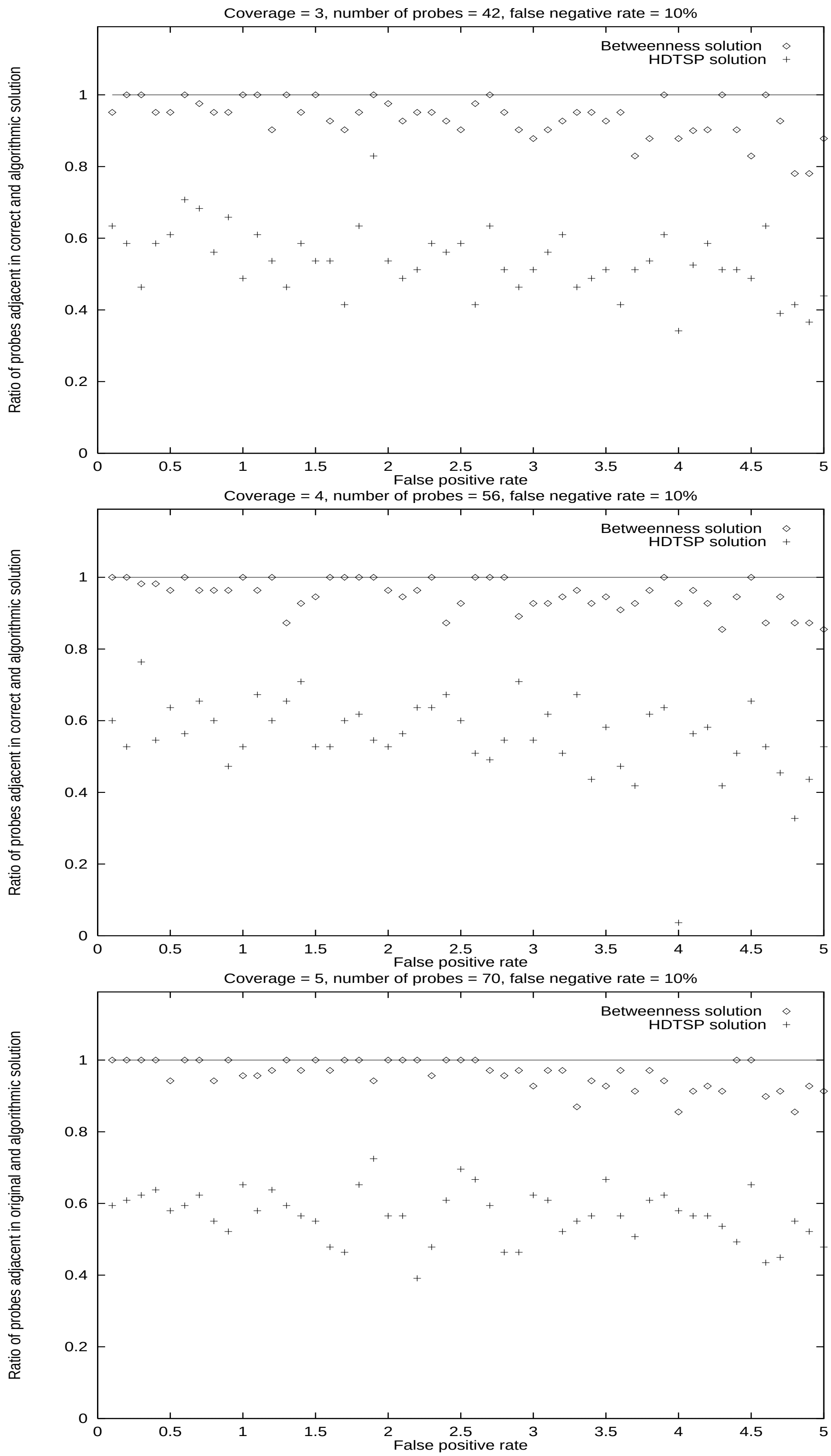

Figure 2 Adjacency quality.

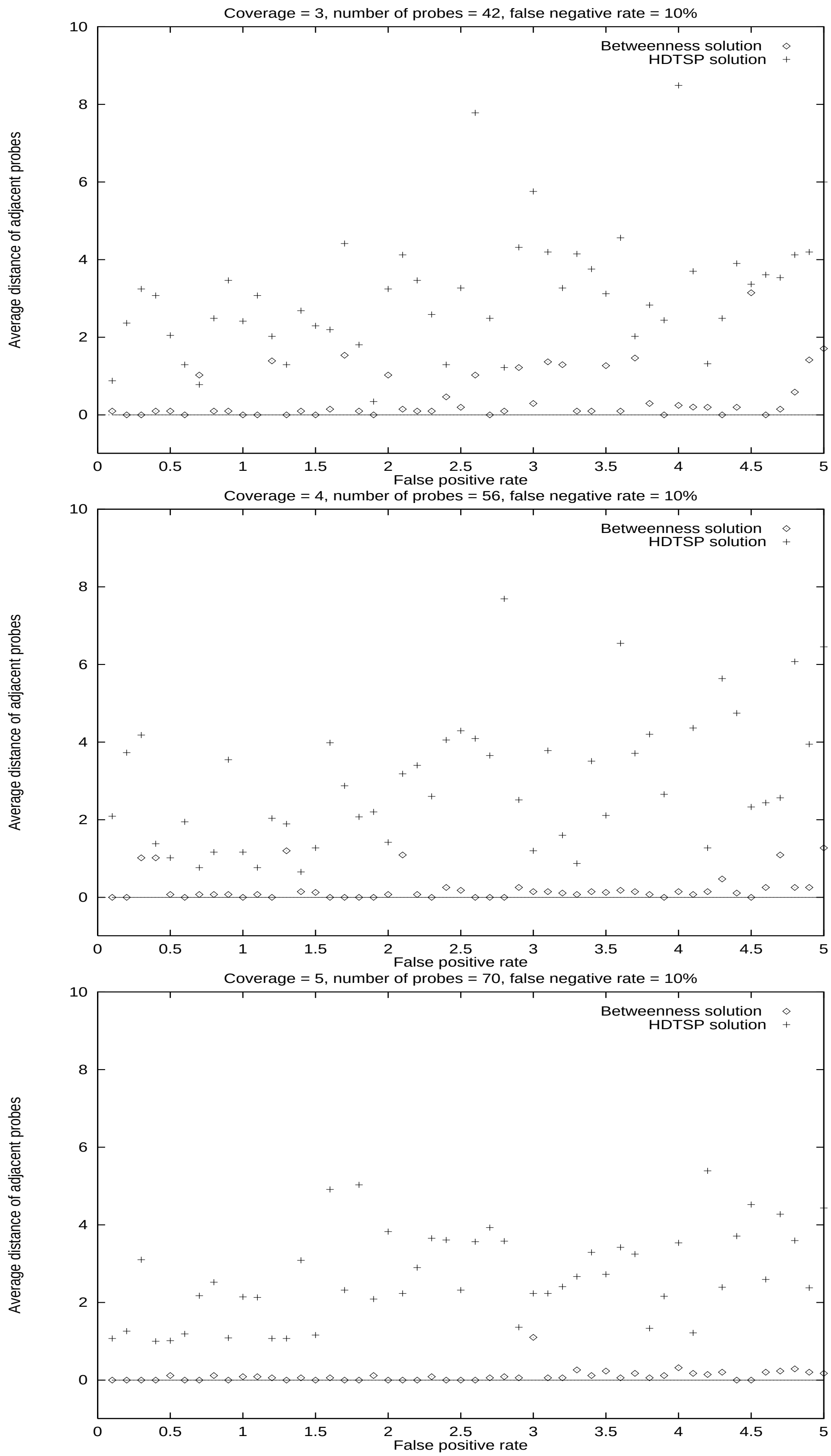

Figure 3 Distance quality.

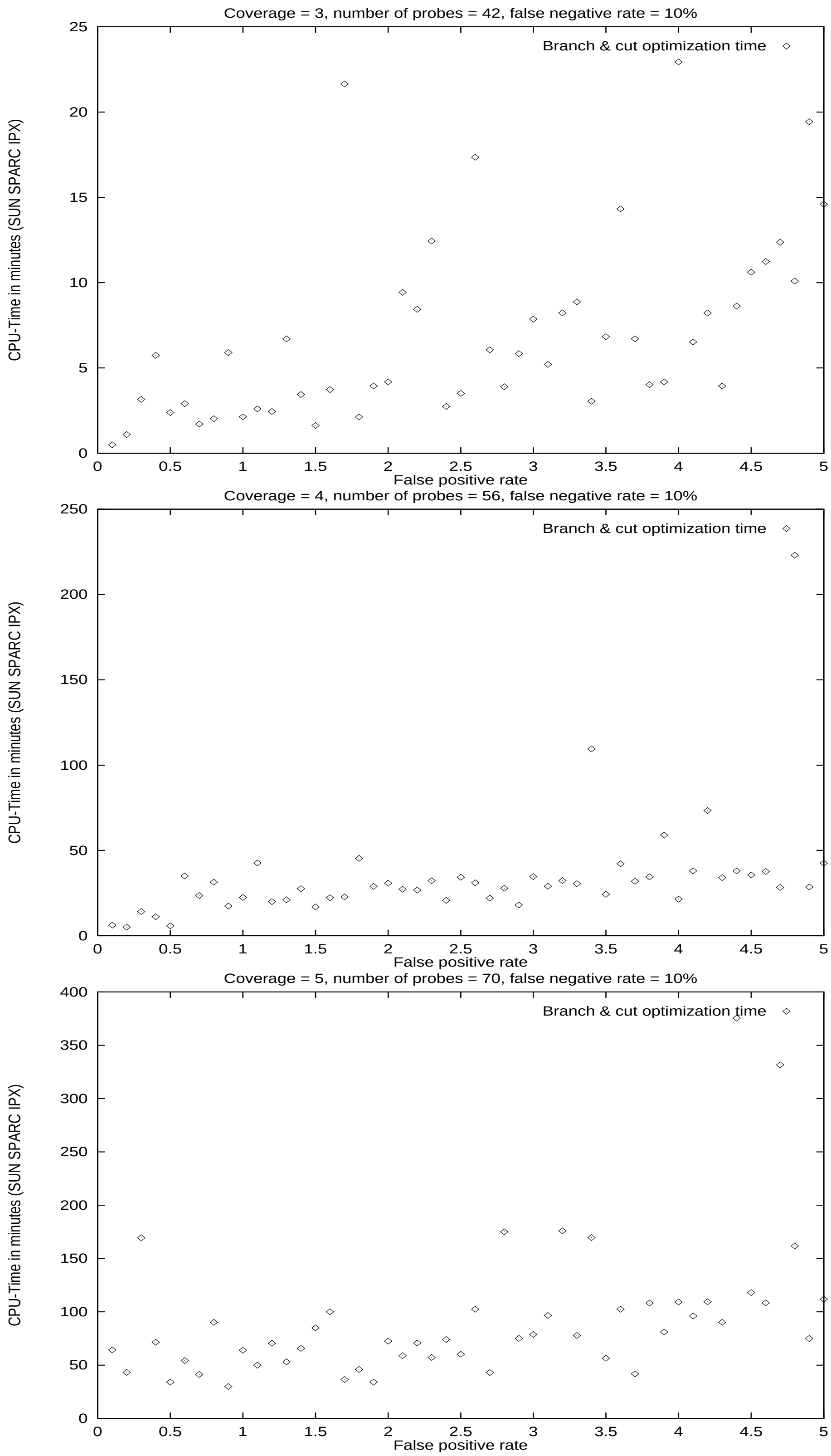

Figure 4 Computation time.

of the optimal solution of Weighted Betweenness is better in almost every instance than the quality of the optimal solution of the HDTSP.

Figure 4 shows the running times of the algorithm on a SUN SPARC IPX using CPLEX2.2 to solve the linear programs. As can be seen, the problems become harder to solve as the false positive ratio increases.

5.3 Heuristic solution by a hybrid approach

Unfortunately, we are currently able to solve to optimality only relatively small instances of Weighted Betweenness (up to 100 probes depending on the ratio of false positives to false negatives). For larger instances, however, the following *hybrid approach* appears promising. We use the exact solution of local betweenness problems to screen out errors in the full problem (most of which are false positives), and then solve a HDTSP on the edited full problem (which has been demonstrated to work well in the absence of false positives and negatives).

The intuition is that an entry classified as a false positive in an optimal solution of a local betweenness problem over a subset of the probes that are adjacent in the correct solution, is also likely to be a false positive in optimal solution of the full problem. We implement this idea as follows. Let s be the maximum number of probes in a local betweenness problem that we are willing to solve to optimality. (Here we take s to be 30.) For each probe P in the data, we generate a local betweenness subproblem over the $s/2$ probes that are nearest neighbours of P with respect to Hamming distance, together with all end-probes from the opposite ends of the clones for which the above $s/2$ probes are end-probes. We call an entry of the original matrix a false positive if its corresponding betweenness constraint is violated by the solution to every subproblem that involves it. After removing the presumed false positives, the matrix changes, so the complete procedure could be iterated several times.

Figure 5 compares the optimal HDTSP-solution with the corresponding optimal HDTSP-solution after running the screening procedure for an instance with coverage 4, 400 probes,

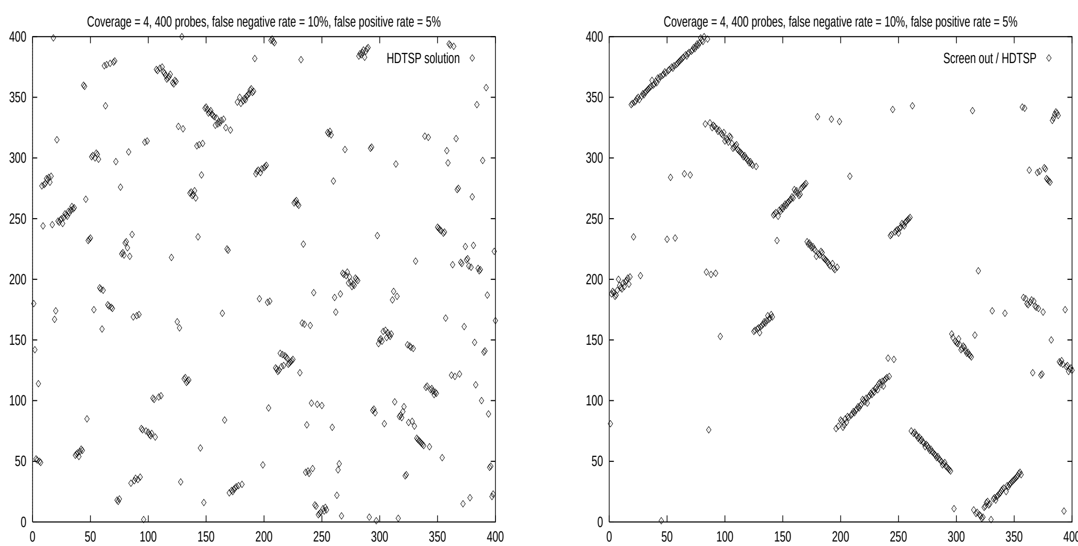

Figure 5 Exact solution of HDTSP without and with screening.

false positive rate 5%, and false negative rate 10%. (This false positive rate is more than an order of magnitude larger, and the coverage much lower, than what has been attempted previously [AKWZ94].) The original probe order is given as the identity permutation, and the computed order is plotted against the identity. A perfect solution would appear as a line of slope 1 or -1. Compared to the raw HDTSP-solution, the second solution contains longer line segments of ± 45 degrees, which suggests that a Betweenness-HDTSP hybrid may offer a superior approach. With screening we could reduce the number of false positives from 3902 to 109, although the number of false negatives increased from 179 to 418.

6 Conclusion and future directions

We have shown that maximum-likelihood physical mapping from clone-probe hybridization data by unique end-probes in the presence of false positive and false negative errors can be formulated as the Weighted Betweenness Problem. With this formulation, we can solve small instances to optimality using methods from polyhedral combinatorics. Computational experiments with the exact branch-and-cut algorithm support the validity of the maximum-likelihood model. Comparison with results obtained by solving the corresponding Traveling Salesman Problems to optimality suggests that exact optimization of the maximum-likelihood function through Weighted Betweenness may be a more robust approach for physical mapping than the widely-used transformation to a Hamming-Distance Traveling Salesman Problem.

We close by indicating how this approach can be extended in a natural way to data with missing end-probes and chimeric clones.

6.1 Data without end-probes

If end-probes are not given for a clone, we can add artificial probes for the clone that simulate its missing end-probes. If one end-probe of the clone is known, this adds one artificial end-probe, and if no end-probes are known, this adds two artificial end-probes. We can then generate betweenness and nonbetweenness constraints for the clone as before, taking care that the artificial end-probes only induce constraints for the given clone. Finding a linear order of all the probes (including the artificial ones) that minimizes the objective function will again minimize the weighted sum of false positive and false negative errors.

Remarkably, this simple modification allows us to handle arbitrary probe-clone hybridization data (while increasing the number of variables), and indicates that the betweenness formulation is more general than may appear.

6.2 Chimeric clones

We can also extend our model to handle chimeric clones as follows. For each clone with end-probes i and k, we add two artificial probes i' and k' that simulate the internal boundaries of the ends of its two chimeric fragments. (This addresses 2-chimerism, but can also be extended to 3-chimerism, and so on.) We ensure that these two artificial probes lie between the clone's end-probes by introducing betweenness conditions (i, i', k) and (i, k', k) and fixing the corresponding variables by setting

$$b_{ii'k} = 1 \tag{43}$$
$$b_{ik'k} = 1. \tag{44}$$

If a probe that does not hybridize to the clone falls between the two artificial probes, it should not be counted as a false negative, while if a probe that does hybridize to the clone falls between the artificial probes, it should be counted as a false positive. The first situation can be modeled by introducing nonbetweenness conditions $\overline{(i',j,k')}$ with weight $-c_\eta$ for all nonbetweenness conditions $b_{ijk} \in \overline{\mathcal{B}}$. The second situation can be achieved with nonbetweenness conditions $\overline{(i',j,k')}$ with weight c_ρ for all betweenness conditions $b_{ijk} \in \mathcal{B}$. We also introduce an additional variable d_{ik} for the clone with end-probes i and k that is 1 if and only if a probe falls between its two artificial internal probes. To force d_{ik} to equal 1 if a probe lies between the artificial probes, we add the inequalities

$$d_{ik} \;\; \geq \;\; b_{i'jk'} \quad \text{for all} \quad b_{ijk} \in \mathcal{B} \cup \overline{\mathcal{B}}. \tag{45}$$

The objective function to be minimized changes to

$$c_\rho \sum_{b_{ijk} \in \mathcal{B}} (1 - b_{ijk}) \; + \; c_\eta \sum_{b_{ijk} \in \overline{\mathcal{B}}} b_{ijk} \; +$$

$$c_\chi \sum_{d_{ik} \in \mathcal{C}} d_{ik} \; + \; c_\rho \sum_{b_{i'jk'} \in \mathcal{B}'} b_{i'jk'} \; - \; c_\eta \sum_{b_{i'jk'} \in \overline{\mathcal{B}'}} b_{i'jk'},$$

where $\mathcal{C}$ denotes the set of all clones and $\mathcal{B}' = \{b_{i'jk'} \mid \exists\, b_{ijk} \in \mathcal{B}\}$ and $\overline{\mathcal{B}'} = \{b_{i'jk'} \mid \exists\, b_{ijk} \in \overline{\mathcal{B}}\}$.

Lastly, we wish to emphasize that this new approach to physical mapping has still to stand the test of real-world problem instances.

References

[AKNW95] F. Alizadeh, R.M. Karp, L.A. Newberg, and D.K. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. *Algorithmica*, 13(1/2):52–76, 1995.

[AKWZ94] F. Alizadeh, R.M. Karp, D.K. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Disscrete Algorithms*, pages 489–500. ACM Press, 1994.

[BL76] K. Booth and G. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.

[Boo75] K.S. Booth. *PQ-Tree Algorithms.* PhD thesis, Univ. of California, Berkeley, 1975.

[CDGH95] R. Capra, J. Daues, W. Gillett, and L. Hanks. Fragment collapsing and splitting while assembling high-resolution restriction maps. *Journal of Computational Biology*, 2(2):185–205, 1995.

[CR96] T. Christof and G. Reinelt. Combinatorial optimization and small polytopes. *Top*, 4(1):1–64, 1996. Spanish Statistical and Operations Research Society.

[CS95] B. Chor and M. Sudan. A geometric approach to betweenness. In P. Spirakis, editor, *Algorithms – ESA '95*, volume 979 of *Lecture Notes in Computer Science*, pages 227–237. Springer, 1995.

[GHL+93] A. Grigoriev, J. Hoheisel, H. Lehrach, E. Maier, and R. Mott. Algorithms and software tools for ordering clone libraries: Application to the mapping of the genome of schizosaccharomyces pombe. *Nucleic Acids Res.*, 21:1965–1974, 1993.

[GI95] D.S. Greenberg and S. Istrail. Physical mapping by STS hybridization: Algorithmic strategies and the challenge of software evaluation. *Journal of Computational Biology*, 2(2):219–273, 1995.

[GJR84] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32:1195–1220, 1984.

[GJR85] M. Grötschel, M. Jünger, and G. Reinelt. Facets of the linear ordering polytope. *Mathematical Programming*, 33:43–60, 1985.

[GLM94] A. Grigoriev, H. Lehrach, and A. Mott. An algorithm to detect chimeric clones and random noise in genomic mapping. *Genomics*, 22:482–486, 1994.

[JM95] M. Jain and G. Myers. A note on scoring clones given a probe ordering. *Journal of Computational Biology*, 2(1):33–38, 1995.

[JRT94] M. Jünger, G. Reinelt, and S. Thienel. Optimal and provably good solutions for the symmetric traveling salesman problem. *Zeitschrift für Operations Research*, 40:183–217, 1994.

[JRT95] M. Jünger, G. Reinelt, and S. Thienel. Practical problem solving with cutting plane algorithms in combinatorial optimization. In L. Lovász W. Cook and P. Seymour, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 20: Combinatorial Optimization*. American Mathematical Society, 1995.

[MLV97] P. Mutzel, H.P. Lenhof, and M. Vingron. Computational molecular biology. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, chapter 23. 1997. to appear.

[Opa79] Opatrny. Total ordering problem. *SIAM Journal of Computing*, 8(1):111–114, 1979.

[Thi95] Stefan Thienel. *ABACUS A Branch-And-CUt System*. PhD thesis, Universität zu Köln, 1995.