

DECONVOLUTING BAC-GENE RELATIONSHIPS USING A PHYSICAL MAP*

Yonghui Wu

*Department of Computer Science and Engineering
University of California, Riverside, CA 92521, USA
yonghui@cs.ucr.edu*

Lan Liu

*Department of Computer Science and Engineering
University of California Riverside, CA 92521, USA
lliu@cs.ucr.edu*

Timothy J. Close

*Department of Botany and Plant Sciences
University of California, Riverside, CA 92521, USA
timothy.close@ucr.edu*

Stefano Lonardi[†]

*Department of Computer Science and Engineering
University of California, Riverside, CA 92521, USA
stelo@cs.ucr.edu*

Motivation: The deconvolution of the relationships between BAC clones and genes is a crucial step in the selective sequencing of the regions of interest in a genome. It usually requires combinatorial pooling of unique probes obtained from the genes (unigenes), and the screening of the BAC library using the pools in a hybridization experiment. Since several probes can hybridize to the same BAC, in order for the deconvolution to be achievable the pooling design has to be able to handle a large number of positives. As a consequence, smaller pools need to be designed which in turn increases the number of hybridization experiments possibly making the entire protocol unfeasible.

Results: We propose a new algorithm that is capable of producing high accuracy deconvolution even in the presence of a weak pooling design, i.e., when pools are rather large. The algorithm compensates for the decrease of information in the hybridization data by taking advantage of a physical map of the BAC clones. We show that the right combination of combinatorial pooling and our algorithm not only dramatically reduces the number of pools required, but also successfully deconvolutes the BAC-gene relationships with almost perfect accuracy.

*A preliminary version of this work was presented at the *LSS Computational Systems Bioinformatics Conference*, San Diego CA, and included in its *Proceedings* (2007).

[†]Corresponding author

Availability: Software available on request from the first author.

Keywords: BAC clone; unigene; deconvolution; group testing; combinatorial pooling; integer linear programming; linear programming; randomized rounding.

1. INTRODUCTION

While the number of fully sequenced organisms is growing rather rapidly, some organisms are unlikely to be sequenced in the immediate future due to the large size and highly repetitive content of their genomes. Many of these latter class of organisms are in the plant kingdom. For these cases, a feasible alternative strategy called *reduced representation sequencing* (see, e.g. [1]) entails focusing only on the *gene space*, that is the portion of the genome that is gene rich. It is well known that in higher organisms genes are not uniformly distributed throughout the genome but instead tend to cluster into gene-rich regions of the chromosomes (see, e.g. [11, 9]).

In many cases, however, even this latter strategy is too expensive or laborious. The next level of reduced sequencing requires a specific list of genes of interests (e.g., abiotic stress-related or pathogen responsive). The task then is to identify the portion of the genome that contains these genes of interest, e.g., by identifying the BAC clones^a carrying those genes, and then sequence solely these BAC clones.

Identification of the BACs containing a specific set of genes is called *deconvolution*. More precisely, the goal of BAC-gene deconvolution is to precisely assign each gene (unigene^b) to one or more BAC clones in the BAC library for that organism.

Another important reason to deconvolute BAC-gene relationships is to place BAC clones (and all the genes that they contain) on the genetic linkage map^c, if such map is available. This placement is possible when a gene within a BAC has been placed on the genetic linkage map.

The assignment of genes to BAC clones can be accomplished experimentally by performing hybridization between a characteristic short sequence in the gene called *probe*, and the entire BAC library. The gene is then assigned to the set of BAC clones that are positive for that probe. In this paper we assume that each probe has the property that it will hybridize only to the BAC clones containing the original gene from which it was obtained. In other words, we assume that each probe is a unique signature for each gene (unigene). The uniqueness of each probe can be established with certainty only if we have the complete knowledge of the full sequence of the genome of interest, but the problem of designing such probes is well-studied (see, e.g. [16, 17]). Because our assumption establishes a one-to-one

^aBACs are artificial chromosome vectors derived from bacteria used for cloning relatively large DNA fragments, typically in the range of 100K nucleotide bases.

^bA *unigene* is obtained by assembling one or more ESTs. A unigene (if correctly assembled) is a portion of a gene transcript.

^cA genetic linkage map is a linear map of the relative positions of genes along a chromosome, where distances are established by analyzing the frequency at which two gene loci become separated during chromosomal recombination.

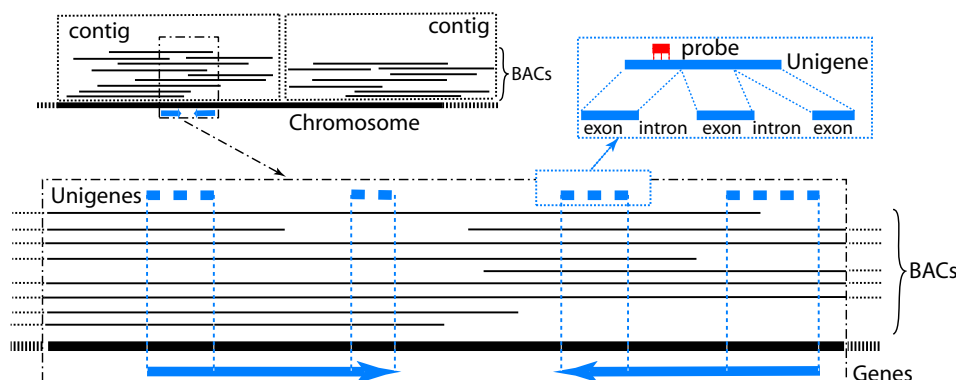


Fig. 1. An illustration of a chromosome, BAC clones, contigs, genes, unigenes and probes. The goal of deconvolution is to uniquely determine the set of BACs that each probe/unigene belongs to.

correspondence between probes and unigenes, in the context of this paper the terms “probes” and “unigenes” are equivalent and can be used interchangeably.

Figure 1 illustrates the BAC-unigene deconvolution problem. Each chromosome is “covered” by a set of BAC clones. The totality of the BAC clones constitutes the BAC library. A gene, represented by the arrow below the chromosome, can be covered by one or more unigenes. A typical unigene covers either the 3’ or the 5’ end of a gene but not the introns. Probes are designed from unigenes trying to avoid splicing sites, otherwise they would not hybridize to the corresponding BAC, which contains the introns as well.

Due to the large numbers of BACs and probes, usually in the order of tens of thousands, it is not feasible to carry out a separate hybridization for each probe/unigene. Group testing² is typically used to reduce the total number of hybridizations required. Here we assume that the probes are grouped into pools^d and that pools are used to screen the BAC library.

The hybridization experiments are then carried out for each BAC and probe pool pair. The readout of one such experiment is positive if the BAC under consideration happens to contain a gene that matches one or more probes in the pool, and as a consequence the hybridization will take place. In Computer Science terms, one can think of the hybridization process as some form of approximate string matching between the probe and the BAC.

So far the steps required for the deconvolution are (1) design a unique probe for each unigene, (2) group the probes into pools, and (3) screen the BAC library using the pools. The input to the deconvolution problem is the set of readouts of all these hybridizations between pool of probes and the BACs. Clearly the smaller the size of each pool, the greater is the number of experiments needed, and the “easier” is

^dOne can pool BACs, but this will not change the nature of the problem.

the deconvolution. Vice versa, if the size of each pool is too large, then too many BACs will be positives, and the deconvolution may not be achieved at all.

In order to study this trade-off, the notion of *decodability* needs to be introduced. We say that a particular pooling design is d -decodable if the deconvolution can be achieved in the presence of d or less positives. In this specific context, if all BACs happen to hybridize to at most d probes and the pooling is d -decodable, the subset of positive probes can be unambiguously determined from the readouts of the experiments. However, if a BAC hybridizes to more than d probes, it may not be possible to achieve the deconvolution. If the goal is to resolve the BAC-probe/unigene relationships exactly, the pools would have to be designed with a decodability greater than or equal to the maximum number of probes that a BAC might possibly contain.

As we will see later in the paper, in order for a pool design to have a high decodability the pools have to be rather small, which in turn can make the number of hybridization experiments prohibitively high. Our objective is to use a pooling design with low decodability (e.g., $d = 1$ or $d = 2$) which is fast and inexpensive and exploit additional information to achieve deconvolution. More specifically, we require some knowledge of overlap between the BAC clones as provided by a physical map^e of the genome of interest.

The rest of the paper is organized as follows. In Section 2 we define formally the problem, and we describe three algorithms for solving the deconvolution. The first one uses solely the results from the hybridization, the second exploits the physical map, and the third is a variation of the second when we have a “perfect” physical map. Although in practice it is unrealistic to assume a perfect map, this algorithm will be useful in the simulations to test the limits of our method. In that section we also formalize the deconvolution problem (with physical map) as a new optimization problem, and prove that it is NP-hard. The optimization problem is expressed in the form of an integer linear program, which is then relaxed to a linear program in order to be solved efficiently. To obtain the best integer solutions, several iterations of randomized rounding are applied to the solutions obtained from the linear program. Our randomized algorithm can be proved to achieve a constant approximation ratio.

In Section 3 we report experimental results on simulated hybridization data on the rice genome, and on real hybridization data on the genome of barley. The results show that our algorithm is capable of deconvoluting a much higher percentage of the BAC-gene relationships from the hybridization data than the naive basic approach. In particular, if given a high quality physical map, we can solve almost 100% of the assignments with almost perfect accuracy (if the pooling is well designed).

^eA *physical map* consists of a linearly ordered set of BAC clones encompassing the chromosomes. Physical maps can be generated by first digesting the BAC clones with restriction enzymes and then detecting overlaps between clones by matching the lengths of the fragments produced by the digestion.

2. METHODS

As mentioned above, the deconvolution process consists of two major steps. In the first, we use a low-decodability pooling design, screen the BAC library using these pools, and collect the hybridization data. Since the pooling has low-decodability, the data obtained by the first step will deconvolute the BAC-unigene relationships only partially. In the second step, we will exploit the physical map information to attempt to resolve the remaining ambiguities.

2.1. Problem Formulation

Let \mathbb{O} be the set of all the unique probes obtained from the unigenes and let \mathbb{B} be set of all BAC clones. A *pool* \mathbf{p} of probes is a simply a subset of \mathbb{O} , that is $\mathbf{p} \subset \mathbb{O}$. The collection of all pools is denoted by \mathbb{P} . The result of the hybridization of BAC b with pool \mathbf{p} is captured by the binary function $h(b, \mathbf{p})$. We have $h(b, \mathbf{p}) = 1$ if the readout is positive, $h(b, \mathbf{p}) = 0$ otherwise. If we assume no experimental errors, $h(b, \mathbf{p}) = 1$ if and only if there exists at least one probe from the pool \mathbf{p} that matches somewhere in BAC b .

Given the values of $h(b, \mathbf{p})$ for all pairs b, \mathbf{p} , the deconvolution problem is to establish an assignment between the probes in \mathbb{O} and the BAC clones in \mathbb{B} , in a such a way that it satisfies the value of h .

2.2. Basic Deconvolution

The basic deconvolution is rather simple. First, we determine for each BAC b the set of probes that it cannot contain, which we denote by $\overline{\mathbb{O}}_b$. This set can be obtained as follows

$$\overline{\mathbb{O}}_b = \{o \in \mathbb{O} \mid \exists \mathbf{p} \in \mathbb{P} \text{ such that } o \in \mathbf{p} \text{ and } h(b, \mathbf{p}) = 0\}$$

Next, for each pair (b, \mathbf{p}) of positive hybridization, we construct the pair $(b, \mathbb{O}_{b, \mathbf{p}})$ where $\mathbb{O}_{b, \mathbf{p}} = \mathbf{p} - \overline{\mathbb{O}}_b$. The presence of the pair $(b, \mathbb{O}_{b, \mathbf{p}})$ means that BAC b has to contain at least one probe from the set $\mathbb{O}_{b, \mathbf{p}}$. The output from this first step is a list of such pairs, which we denote with symbol Ω . For any pair $(b, \mathbb{O}_{b, \mathbf{p}}) \in \Omega$ such that $|\mathbb{O}_{b, \mathbf{p}}| = 1$, the relationship between BAC b and the only probe $o \in \mathbb{O}_{b, \mathbf{p}}$ is exact, that is b must contain o (or, alternatively, o is assigned to b). We call *exact pairs* those pairs in Ω for which $|\mathbb{O}_{b, \mathbf{p}}| = 1$. By definition, exact pairs can be resolved uniquely.

A straightforward implementation of the above algorithm is not so efficient. This is because $\overline{\mathbb{O}}_b$ is rather large since a BAC typically hybridizes to only a few pools. In our implementation, we construct the compliment of $\overline{\mathbb{O}}_b$ for every BAC b . The details of our implementation is not shown here.

Since the decodability of the pool design is much lower than the number of positives that a BAC may contain, we expect the proportion of exact pairs to be rather small. In practice, the large majority of the pairs in Ω will be non-exact. For

these latter pairs, the relationships between BAC and probes can be solved only by employing additional information. Next, we present an algorithm that resolves the non-exact pairs using a physical map which may contain errors. Then, we present an algorithm for the case where high-quality or near perfect physical map is available.

2.3. Deconvolution Using an Imperfect Physical Map

For the purposes of the deconvolution, the essential information on the physical map is represented by the overlaps between BAC clones. We will show how the knowledge of the overlap between BACs can help resolve ambiguous (non-exact) BAC-probe relationships. The following two observations are the cornerstones of our algorithms.

The first observation is that if BAC b_1 has a large overlap with BAC b_2 , and at the same time probe o belongs to BAC b_2 , then probe o will belong to BAC b_1 with high probability. The second observation is that since probes are carefully designed to be unique for a specific gene (unigene), the probability that they will match/hybridize in some location other than the location of the gene from which they were designed is very low. Therefore, if probe o belongs to BAC b_1 , and BAC b_2 does not overlap BAC b_1 , then o will belong to BAC b_2 with very low probability.

In this section we assume that we are given a physical map that can potentially contains errors. Such a map could have been obtained, for example, by running FPC¹⁰ software on some restriction fingerprinting data. The output of FPC is an assembly of the BAC clones into disjoint sets, called *contigs*. If two BAC clones belong to the same contig, they are very likely to overlap with each other. On the other hand, if two BAC clones are from two different contigs, they are very unlikely to overlap. Formally, each contig is a subset of \mathbb{B} . Let \mathbb{C} denote the collection of contigs. The set \mathbb{C} is a partition of \mathbb{B} . Since probes are designed to hybridize only to one location throughout the genome, we expect that each of them will only belong to one contig.

The list of pairs in Ω can be interpreted as a list of constraints. A pair $(b, \mathbb{O}_{b,p}) \in \Omega$ is satisfied if there exists a probe $o \in \mathbb{O}_{b,p}$ and o is assigned to BAC b . Ideally, we would like to compute a BAC-probe assignment such that each probe is assigned to a set of BACs belonging to the same contig and all the constraints in Ω are satisfied. Due to the fact that the physical map is imperfect and that some of the probes may not match anywhere^f or match multiple locations^g in the genome, we may not be able to satisfy all constraints. Therefore, a reasonable objective is to assign probes to BACs so that the number of satisfied constraints is maximized, subject to the restriction that each probe may be only assigned to one contig.

We have now turned the deconvolution problem into an optimization problem, which is going to be tackled in two phases. In phase I, we will assign probes to

^fThis can happen, for example, if the probe happens to cross a splicing site.

^gIn practice it is impossible to guarantee the uniqueness of probes unless one knows the whole sequence of the genome and the hybridization is modeled perfectly *in silico*.

contigs (not to BACs). The list of BAC constraints Ω is transformed to a new list of contig constraints Ω' of the same size, as follows. For each constraint $(b, \mathbb{O}_{b,\mathbf{p}}) \in \Omega$ we create the contig constraint $(\mathbf{c}, \mathbb{O}_{b,\mathbf{p}})$ where \mathbf{c} is the contig to which b belongs. A contig constraint $(\mathbf{c}, \mathbb{O}_{b,\mathbf{p}})$ is satisfied if probe $o \in \mathbb{O}_{b,\mathbf{p}}$ and o is assigned to contig \mathbf{c} . As said, the goal is to maximize the number of satisfied constraints in Ω' by assigning each probe to at most one contig. If a constraint in Ω' appears in multiple copies, that constraint will contribute its multiplicity to the objective function when satisfied. In phase II, probes will be assigned to BACs. Let o be a probe in \mathbb{O} and assume that o was assigned to contig $\mathbf{c} \in \mathbb{C}$ in phase I. In phase II, o is assigned to the following set of BACs

$$\{b \in \mathbf{c} \mid \exists (b, \mathbb{O}_{b,\mathbf{p}}) \in \Omega \text{ such that } o \in \mathbb{O}_{b,\mathbf{p}}\}. \quad (1)$$

It can be easily verified that if the assignment of probes to contigs in phase I is optimal (i.e., it satisfies the maximum number of constraints from Ω') the final assignment of probes to BACs using (1) is also optimal (i.e., it satisfies the maximum number of constraints from Ω).

Since the final assignment can be easily obtained from an optimal solution to phase I, the rest of this section will be focused on solving the optimization problem in phase I. First we present a formal description of the problem.

MAXIMUM CONSTRAINT SATISFYING PROBE-CONTIG ASSIGNMENT (MCSPCA)

Instance: A set of probes \mathbb{O} , a set of contigs \mathbb{C} and a list of constraints Ω' , where each item of Ω' has the form $(\mathbf{c}, \mathbb{O}_{b,\mathbf{p}})$, $\mathbf{c} \in \mathbb{C}$ and $\mathbb{O}_{b,\mathbf{p}} \subseteq \mathbb{O}$.

Objective: Assign each probe to at most one contig in \mathbb{C} such that the number of satisfied constraints in Ω' is maximized. A constraint $(\mathbf{c}, \mathbb{O}_{b,\mathbf{p}})$ is satisfied if one or more of the probes from $\mathbb{O}_{b,\mathbf{p}}$ is assigned to \mathbf{c} .

Unfortunately but not surprisingly, the MCSPCA optimization problem is NP-complete. This can be proved by a reduction from the 3SAT problem⁸.

Theorem 1. *The MCSPCA problem is NP-hard.*

Proof. Let us define a language problem L which corresponds to a special case of the above optimization problem as follows

$$L = \{(\mathbb{O}, \mathbb{C}, \Omega') \mid \text{there exists a many to one mapping from } \mathbb{O} \text{ to } \mathbb{C} \\ \text{such that all constraints in } \Omega' \text{ are satisfied}\}$$

where \mathbb{O} , \mathbb{C} and Ω' are defined in the original optimization problem above. We will prove that L is NP-complete by a reduction from 3SAT. As a result, the MCSPCA optimization problem is also NP-complete.

Let $\varphi = (V, S)$ be an instance of 3SAT, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of variables and $S = \{s_1, s_2, \dots, s_k\}$ is the set of clauses of φ . We construct an instance $\xi = (\mathbb{O}, \mathbb{C}, \Omega')$ of L , with the property that $\xi \in L$ if and only if φ is

satisfiable. The construction is as follows: $\mathbb{O} = \{v_1^t, v_1^f, v_2^t, v_2^f, \dots, v_n^t, v_n^f\}$, where v_i^t and v_i^f correspond to the “true” and “false” assignments to variable $v_i \in V$ respectively. We set $\mathbb{C} = \{c, v_1, v_2, \dots, v_n\}$, and Ω' consists of two parts, namely Ω'_1 and Ω'_2 . Ω'_1 is used to ensure the satisfiability of all the clauses in S and Ω'_2 is used to ensure that either v_i^t or v_i^f is used to satisfy the constraints, but not both of them. We set

- $\Omega'_1 = \{(c, \{s_1^1, s_1^2, s_1^3\}), (c, \{s_2^1, s_2^2, s_2^3\}), \dots, (c, \{s_k^1, s_k^2, s_k^3\})\}$, where (s_i^1, s_i^2, s_i^3) are the three literals corresponding to clause $s_i \in S$. For example, if $s_1 = (v_1 \cup \bar{v}_2 \cup v_3)$, then the constraint $(c, \{v_1^t, v_2^f, v_3^t\})$ is added to Ω'_1 .
- $\Omega'_2 = \{(v_1, \{v_1^t, v_1^f\}), (v_2, \{v_2^t, v_2^f\}), \dots, (v_n, \{v_n^t, v_n^f\})\}$.

It can be verified that $\xi \in L$ if and only if φ is satisfiable. \square

2.3.1. Solving the MCSPCA problem via integer programming

The MCSPCA optimization problem can be solved with integer linear programming (ILP) as follows. Let $X_{o,c}$ be the variable associated with the possible assignment of probe o to contig \mathbf{c} , which is set to 1 if o is assigned to \mathbf{c} and set to 0 otherwise. Let Y_q be a variable corresponding to the constraint $q \in \Omega'$, which is set to 1 if q is satisfied and set to 0 otherwise. The following integer program encodes the MCSPCA problem.

$$\begin{aligned}
 & \text{Maximize } \sum_{q \in \Omega'} Y_q \\
 & \text{Subject to } \sum_{c \in \mathbb{C}} X_{o,c} \leq 1 & \forall o \in \mathbb{O} \\
 & Y_q \leq 1 & \forall q \in \Omega' \\
 & Y_{q=(c,S)} \leq \sum_{o \in S} X_{o,c} & \forall q \in \Omega' \\
 & X_{o,c} \in \{0, 1\} & \forall o \in \mathbb{O}, \mathbf{c} \in \mathbb{C} \\
 & Y_q \in \{0, 1\} & \forall q \in \Omega'
 \end{aligned} \tag{2}$$

The first constraint ensures that each probe can be assigned to at most one contig, whereas the third constraint makes sure that a constraint $q = (\mathbf{c}, S) \in \Omega'$ is satisfied iff one or more probes from S is assigned to \mathbf{c} .

The integer program above cannot be solved optimally for real world instances because solving integer programs is very time consuming and in worst case takes exponential time. Faster and approximate algorithms must be obtained. Below we present one such fast approximate algorithm based on relaxation and randomized rounding.

2.3.2. Relaxation and randomized rounding

The integer linear program is first relaxed to the corresponding linear program (LP) by turning all $\{0, 1\}$ constraints into $[0, 1]$. Then, the linear program can be

efficiently solved optimally. Let $\{X_{o,\mathbf{c}}^* \mid \forall o \in \mathbb{O}, \mathbf{c} \in \mathbb{C}\}$ and $\{Y_q^* \mid \forall q \in \Omega'\}$ be the optimal solutions to the linear program. In general, $X_{o,\mathbf{c}}^*$ and Y_q^* are fractional values between 0 and 1. $X_{o,\mathbf{c}}^*$ is to be interpreted as the probability of assigning probe o to contig \mathbf{c} . We apply standard randomized rounding technique to convert the fractional solutions into an integral solution as follows. For each probe o , let $C_o = \{\mathbf{c} \mid \mathbf{c} \in \mathbb{C}, X_{o,\mathbf{c}}^* > 0\}$. If $\sum_{\mathbf{c} \in C_o} X_{o,\mathbf{c}}^* = 1$, randomly assign o to one of the contigs in C_o according to the associated probability $X_{o,\mathbf{c}}^*$. If $\sum_{\mathbf{c} \in C_o} X_{o,\mathbf{c}}^* < 1$, randomly assign o to one of the contigs in C_o according to the associated probability $X_{o,\mathbf{c}}^*$, or to none of the contigs with probability $1 - \sum_{\mathbf{c} \in C_o} X_{o,\mathbf{c}}^*$. The output from this rounding step is an assignment of each probe to at most one contig, as required. This rounding procedure can be applied multiple times. Among the assignments produced by each individual rounding step, the one that satisfies the maximum number of constraints will be taken as the final solution. The following theorem shows that the randomized rounding step achieves a constant approximation ration.

Theorem 2. *The randomized MCSPCA algorithm achieves approximation ratio $(1 - e^{-1})$.*

Proof. Let us still use $\{X_{o,\mathbf{c}}^* \mid \forall o \in \mathbb{O}, \mathbf{c} \in \mathbb{C}\}$ and $\{Y_q^* \mid \forall q \in \Omega'\}$ to denote the optimal solutions obtained by solving the linear program. Let OPT_f be the optimal value of the objective function of the linear program, that is, $OPT_f = \sum_{q \in \Omega'} Y_q^*$. Let I_q be an indicator random variable, which is set to 1 if the constraint q is satisfied under our randomized rounding step and to 0 otherwise. Let W denote the total number of satisfied constraints after the rounding step. Clearly, $W = \sum_{q \in \Omega'} I_q$, and $E(W) = \sum_{q \in \Omega'} Prob(I_q = 1)$. Consider the following two cases:

- (1) $Y_q^* = 1$. Let q be of the form (\mathbf{c}, S) where $\mathbf{c} \in \mathbb{C}$ and $S \subseteq \mathbb{O}$. In this case, $\sum_{o \in S} X_{o,\mathbf{c}}^* \geq 1$.

$$\begin{aligned} Prob(I_q = 1) &= 1 - Prob(I_q = 0) \\ &= 1 - \prod_{o \in S} (1 - X_{o,\mathbf{c}}^*) \\ &> 1 - e^{-\sum_{o \in S} X_{o,\mathbf{c}}^*} \\ &\geq 1 - e^{-1} \end{aligned}$$

So, $Prob(I_q = 1)/Y_q^* > 1 - e^{-1}$

- (2) $Y_q^* < 1$. Let q be of the form (\mathbf{c}, S) where $\mathbf{c} \in \mathbb{C}$ and $S \subseteq \mathbb{O}$. In this case, $Y_q^* = \sum_{o \in S} X_{o,\mathbf{c}}^*$.

$$\begin{aligned} Prob(I_q = 1) &= 1 - Prob(I_q = 0) \\ &= 1 - \prod_{o \in S} (1 - X_{o,\mathbf{c}}^*) \\ &> 1 - e^{-\sum_{o \in S} X_{o,\mathbf{c}}^*} \\ &= 1 - e^{-Y_q^*} \end{aligned}$$

Algorithm MCSPCA($\mathbb{O}, \mathbb{C}, \mathbb{B}, \Omega$)

0. Convert Ω to Ω'
1. Generate the integer program in (2) from $(\mathbb{O}, \mathbb{C}, \Omega')$
2. Solve the LP relaxation of the ILP in step 1, and obtain the optimal fractional solution $\{X_{o,c}^*\}$ and $\{Y_q^*\}$
3. Apply K steps of randomized rounding and save the best solution
 - for each** $o \in \mathbb{O}$ **do**
 - $C_o = \{\mathbf{c} \mid \mathbf{c} \in \mathbb{C}, X_{o,c}^* > 0\}$
 - Assign o to $\mathbf{c} \in C_o$ with probability $X_{o,c}^*$ or to none of the contigs with probability $1 - \sum_{\mathbf{c} \in C_o} X_{o,c}^*$
4. Further assign probes to BACs
 - if** o is assigned to \mathbf{c} in step 3 **then** assign o to the set of BACs $\{b \in \mathbb{B} \mid \exists (b, \mathbb{O}_{b,\mathbf{p}}) \in \Omega \text{ s.t. } o \in \mathbb{O}_{b,\mathbf{p}}\}$

Fig. 2. Sketch of the two-phase deconvolution algorithm that exploits an imperfect physical map

$$\text{So, } \text{Prob}(I_q = 1)/Y_q^* > \min_{0 \leq y \leq 1} (1 - e^{-y})/y = 1 - e^{-1}$$

Therefore, to sum up, $E(W) \geq (1 - e^{-1})OPT_f$. \square

The algorithm can be de-randomized via the method of conditional expectation to achieve a deterministic performance guarantee. The de-randomization step follows the procedure in [13]. We observe that the approximation algorithm we propose here is similar to the one for MAX-SAT^{14,5}. To summarize, the sketch of the MCSPCA algorithm is presented in Figure 2.

2.4. Deconvolution Using a Perfect Physical Map

As said, although it is not realistic to assume to have a perfect or near perfect physical map, this variation of the problem allows us to establish the limits of how many assignment we can correctly deconvolute from the hybridization data. This is particularly useful for simulations, to ensure that our algorithm can achieve good results if the input physical map is of good quality.

If we are given a perfect (or near-perfect) physical map, the problem can be tackled from the “opposite” direction. Instead of trying to take advantage of the grouping of BACs into disjoint contigs, we partition each BAC into several pieces. We preprocess the physical maps as follows. We align the BACs along the chromosome according to their relative positions on the physical map. Then, starting from the 5’ end, we cut the chromosome at each location where either a BAC starts or ends. This process breaks the genome into at most $2n$ fragments, where n is the total number of BACs. Each fragment is covered by one or more BACs, and some fragments may be covered by exactly the same set of BACs. In that latter case, only one fragment is kept while the others are removed. At the end of this preprocessing

phase, a set of fragments is produced where each fragment is covered by a distinct set of overlapping BACs. Let us denote the final set of fragments as \mathbb{F} . Given a fragment $f \in \mathbb{F}$ and a BAC $b \in \mathbb{B}$, we use $\mathcal{B}(f)$ to denote the set of BACs that f is covered by, and use $\mathcal{F}(b)$ to denote the set of fragments that b covers.

For the same reasons mentioned above, we expect that each probe will match its intended place in the genome and nowhere else. Our goal is to assign each probe to one fragment while at the same time maximize the number of satisfied constraints in Ω . A constraint $(b, \mathbb{O}_{b,\mathbf{p}}) \in \Omega$ is satisfied if one or more probes from $\mathbb{O}_{b,\mathbf{p}}$ is assigned to any of the fragment in the set $\mathcal{F}(b)$. Given an assignment between probes and fragments, the probe-BAC assignment can be easily obtained. Below is a formal statement of our new optimization problem.

MAXIMUM CONSTRAINT SATISFYING PROBE-FRAGMENT ASSIGNMENT (MCSPFA)

Instance: A set of fragments \mathbb{F} , a set of probes \mathbb{O} , a set of BACs \mathbb{B} and a list of constraints $\Omega = \{(b, \mathbb{O}_{b,\mathbf{p}}) | b \in \mathbb{B}, \mathbb{O}_{b,\mathbf{p}} \subset \mathbb{O}\}$.

Objective: Assign each probe in \mathbb{O} to at most one fragment in \mathbb{F} , such that the number of satisfied constraints in Ω is maximized.

The MCSPFA problem is also NP-hard, since it is a special case of MCSPCA when all BACs in \mathbb{B} are disjoint.

2.4.1. Solving the MCSPFA problem via integer programming

A variant of the integer program that we presented for MCSPCA can also solve this problem optimally. Let $X_{o,f}$ be a variable associated with the possible assignment of probe o to fragment f , which is set to 1 if o is assigned to f , 0 otherwise. Let Y_q be defined in the same way as the previous integer program. The integer linear program for MCSPFA follows.

$$\begin{aligned}
 & \text{Maximize } \sum_{q \in \mathcal{C}} Y_q \\
 & \text{Subject to } \sum_{f \in \mathbb{F}} X_{o,f} \leq 1 && \forall o \in \mathbb{O} \\
 & Y_q \leq 1 && \forall q \in \Omega \\
 & Y_{q=(b,S)} \leq \sum_{o \in S} \sum_{f \in \mathcal{F}(b)} X_{o,f} && \forall q \in \Omega \\
 & X_{o,f} \in \{0, 1\} && \forall o \in \mathbb{O}, f \in \mathbb{F} \\
 & Y_q \in \{0, 1\} && \forall q \in \Omega
 \end{aligned} \tag{3}$$

The major difference between the ILP (3) and the ILP (2) is in the third constraint. The third constraint in the ILP above translates the fact that a constraint $(b, S) \in \Omega$ is satisfied if any probe in S is assigned to any fragment in $\mathcal{F}(b)$.

2.4.2. Relaxation, rounding and analysis

Following the same strategy used in the MCSPCA problem, the ILP is relaxed to its corresponding LP, and then the LP can be solved optimally. Let $\{X_{o,f}^* | \forall o \in$

$\mathbb{O}, f \in \mathbb{F}$ and $\{Y_q^* \mid \forall q \in \Omega'\}$ denote the optimal solutions to the LP. The fractional solution will be rounded to an integer solution by interpreting $X_{o,f}^*$ as the probability of assigning probe o to fragment f .

Let OPT_f be the optimal value of the objective function in the LP, that is $OPT_f = \sum_{q \in \Omega} Y_q^*$. Let I_q be the indicator random variable, which is 1 if the constraint q is satisfied under the above randomized rounding step, 0 otherwise. Let W denote the total number of satisfied constraints after the rounding step. Clearly, $W = \sum_{q \in \Omega} I_q$, and $E(W) = \sum_{q \in \Omega} Prob(I_q = 1)$. In the following, we prove that $E(W) \geq (1 - e^{-1})OPT_f$. This is formally stated as the following theorem:

Theorem 3. *The randomized MCSPFA algorithm achieves an approximation ratio $(1 - e^{-1})$.*

Proof. As we did in the proof for theorem 2, in order to show $E(W) \geq (1 - e^{-1})OPT_f$, it is sufficient to show that $Prob(I_q = 1)/Y_q^* > 1 - e^{-1} \quad \forall q \in \Omega$. We consider the following two cases:

- (1) $Y_q^* = 1$. Let q be of the form (b, S) where $b \in \mathbb{B}$ and $S \subseteq \mathbb{O}$. In this case, $\sum_{o \in S} \sum_{f \in \mathcal{F}(b)} X_{o,f}^* \geq 1$.

$$\begin{aligned} Prob(I_q = 1) &= 1 - Prob(I_q = 0) \\ &= 1 - \prod_{o \in S} \left(1 - \sum_{f \in \mathcal{F}(b)} X_{o,f}^*\right) \\ &> 1 - e^{-\sum_{o \in S} \sum_{f \in \mathcal{F}(b)} X_{o,f}^*} \\ &\geq 1 - e^{-1} \end{aligned}$$

So, $Prob(I_q = 1)/Y_q^* > 1 - e^{-1}$

- (2) $Y_q^* < 1$. Let q be of the form (b, S) where $b \in \mathbb{B}$ and $S \subseteq \mathbb{O}$. In this case, $Y_q^* = \sum_{o \in S} \sum_{f \in \mathcal{F}(b)} X_{o,f}^*$.

$$\begin{aligned} Prob(I_q = 1) &= 1 - Prob(I_q = 0) \\ &= 1 - \prod_{o \in S} \left(1 - \sum_{f \in \mathcal{F}(b)} X_{o,f}^*\right) \\ &> 1 - e^{-\sum_{o \in S} \sum_{f \in \mathcal{F}(b)} X_{o,f}^*} \\ &= 1 - e^{-Y_q^*} \end{aligned}$$

So, $Prob(I_q = 1)/Y_q^* > \min_{0 \leq y \leq 1} (1 - e^{-y})/y = 1 - e^{-1}$

Therefore, it follows that $E(W) \geq (1 - e^{-1})OPT_f$. \square

The MCSPFA algorithm can also be de-randomized via the method of conditional expectation to achieve a deterministic performance guarantee. The pseudo-code for our algorithm is presented in Figure 3.

Algorithm MCSPFA($\mathbb{O}, \mathbb{F}, \mathbb{B}, \Omega$)

1. Generate the integer program (3) from $(\mathbb{O}, \mathbb{F}, \Omega)$
2. Solve the LP relaxation of the ILP in step 1, and obtain the optimal fractional solution $\{X_{o,f}^*\}$ and $\{Y_q^*\}$
3. Apply K steps of randomized rounding and save the best solution
 - for each** $o \in \mathbb{O}$ **do**
 - $F_o = \{f \mid f \in \mathbb{F}, X_{o,f}^* > 0\}$
 - Assign o to $f \in F_o$ with probability $X_{o,f}^*$ or to none of the fragments with probability $1 - \sum_{f \in F_o} X_{o,f}^*$
4. Further assign probes to BACs.
 - if** o is assigned to f in step 3 **then** assign o to all the BACs in $\mathcal{F}(b)$

Fig. 3. Sketch of the two-phase deconvolution algorithm that exploits a perfect physical map

3. EXPERIMENTAL RESULTS

In order to evaluate the performance of our algorithms, we applied them to two datasets. The first one is partially simulated data on the rice genome while the second is real-world data from hybridizations carried out in Prof. Close lab at UC Riverside on the barley genome. Before delving in the experimental setup, we give a short description of the pooling design which is relevant to the discussion.

3.1. Pooling design

Pooling design (or group testing) is a well-studied problem in the scientific literature (see [2] and references therein). Traditionally, biologists use the rather rudimentary 2D or 3D grid design. There are however more sophisticated pooling strategies (see, e.g., [7, 3, 12]). To the best of our knowledge, the shifted transversal design (STD)¹² is among the best choices due to its capability to handle multiple positives, its flexibility and efficiency. STD pools are constructed in layers, where each layer consists of P pools, where P is a prime number. Each layer constitutes a partition of the probes, and the larger is the number of layers, the higher is the decodability of the pooling. More specifically, let Γ be the smallest integer such that $P^{\Gamma+1}$ is greater than or equal to the number of probes to be pooled, and let L be the number of layers. Then, the decodability of the pool set is equal to $\lfloor (L-1)/\Gamma \rfloor$. In order to increase the decodability by one, an additional $P\Gamma$ pools are needed. By a simple calculation, one can realize that the number of pools required to provide sufficient information for deconvolution (e.g., to be at least 10-decodable) for a real-world problem assuming 50,000 BACs and 50,000 unigenes is prohibitively high.

3.2. *Experimental Results On the Rice Genome*

The rice “virtual” BAC library used here is a subset of the Nipponbare BAC library, whose BAC end sequences are hosted at Arizona Genomic Institute (AGI). The fingerprinting data for this library was also obtained from AGI. Our rice virtual library contains the subset of BACs in the original library whose location on the rice genome (*Oryza sativa*) can be uniquely identified and for which restriction fingerprinting data was available. The location of the BACs was determined by BLASTing the BAC end sequences against the rice genome.

Since our rice virtual BAC library is based on real fingerprinting data (agarose gel), we expect the overlapping structure of the rice BACs in the physical map to be an accurate representation of what would happen in other organisms. Also, since we know the actual coordinates of the BACs on the rice genome, we can also produce a perfect physical map and test the maximum amount of the deconvolution that can be extracted from the hybridization data.

For the purposes of this simulation, we restricted our attention to chromosome 1 of rice, which includes 2,629 BACs. This set of BACs provides a 8.59x coverage of chromosome 1. We created a physical map of that chromosome by running FPC¹⁰ on the fingerprinting data with cutoff parameter set to $1e-15$ (all other parameters are left to default). FPC assembled the BACs into 347 contigs and 416 singletons. Not including the singletons, each contig contains on average about 6.4 BACs. Given that the fingerprinting data is noisy, the physical map assembled by FPC cannot be expected to be perfect. It is also well known that the order of the BACs within a contig is generally not reliable⁴.

We then obtained rice unigenes from NCBI (build #65) with the objective of designing the probes. First, we had to establish the subset of these unigenes that belong to chromosome 1. We BLASTed the unigenes against the rice genome, and we selected a subset of 2,301 unigenes for which we had high confidence to be located on chromosome 1. Then, we computed unique probes using OLIGOSPAWN^{16,17}. This tool produces 36 nucleotides long probes, each of which matches exactly the unigene it represents and at the same time it does not match (even approximately) to any other unigenes in the dataset.

OLIGOSPAWN successfully produced unique probes for 2,002 unigenes (out of 2,301). The remaining unigenes were not represented by a probes because no unique 36-mer could be found. This set of 2,002 probes is named probe set 1. Some of the probes in probe set 1, however, did not match anywhere in the genome. This will happen if the probe chosen happens to cross a splicing cite or when the unigene from which it was selected was misassembled. In probe set 1, 330 probes did not match anywhere on rice chromosome 1. The remaining 1,672 probes matched exactly once in rice chromosome 1. This latter set constitutes our probe set 2, which is a “cleaner” version of probe set 1. We observe that in order to clean the probe set one has to have access to the whole genome, which is somewhat unrealistic in practice. Each BAC contains on average 5.8 probes and at most 20 probes.

The probes were hybridized *in silico* to the BACs using the following criteria. A 36 nucleotides probe hybridizes a BAC if they share a common (exact) substring of 30 nucleotides or more. The criteria was debated at length with the biologists in Prof. Close lab and among all suggestions, this one was chosen for its simplicity. Observe that the hybridization table is the only synthetic data in this dataset.

The next step was to pool the probes for group testing. We followed the shifted transversal design pooling strategy¹² and designed four sets of pools, for different choices of the pooling parameters P and L . Recall that in STD the number of pools is $P * L$. Pool set 1 is 1-decodable, obtained by choosing $P = 13$ and $L = 3$. Pool set 2 uses two extra layers ($P = 13, L = 5$), which increased the decodability by 1. For pool set 3, we chose $P = 47$ and $L = 2$. Pool set 3 is also 1-decodable, but since each pool contains a smaller number of probes, it will deconvolute the BAC-probe relationships better than pool set 1. Pool set 4 is constructed from pool set 3 by adding an additional layer ($P = 47, L = 3$). As a consequence, pool set 4 is 2-decodable. The four pooling designs were applied to probe set 1 and probe set 2. In total, we constructed 8 sets of pools.

The hybridization tables h between pools and BACs were formed for the 8 sets of pools. Then, the basic deconvolution step (described in Section 2.2) was carried out. This step produced a list of constraints, some of which were exact pairs and could deconvolute immediately.

The set of BACs, the set of probes, the list of constraints from the previous step, and the physical map produced by FPC, were then fed into MCSPCA. Our ILP-based algorithm produced a set of BAC-probe assignments, which was then merged with the exact pairs obtained by the basic deconvolution to produce the final assignment.

Similarly, the set of BACs, the set of probes, the list of constraints from the basic deconvolution and the perfect physical map were fed into MCSPFA. The assignment obtained by MCSPFA was also merged with the exact pairs to produce the final assignment.

For both algorithms we used the GNU Linear Programming Kit (GLPK) to solve the linear programs. The size of the linear programs are quite large. The number of variables ranged from 47,820 to 165,972 and the number of constraints ranged from 29,412 to 60,475.

To evaluate the accuracy of our algorithms, we employ two performance metrics, namely recall and precision. Recall is defined as the number of correct assignments made by our algorithm divided by the total number of true assignments. Precision is defined as the number of correct assignments made by our algorithm divided by the total number of assignments our algorithm made. Tables 1 and 2 summarize the assignment accuracy of our algorithms. “Basic recall” is the recall of the basic deconvolution step (precision is not reported because it is always 100%).

A few observations are in order. First, note that 2-decodable pooling designs achieve a much better performance than 1-decodable pooling. Second, probe set 2 provides better quality data and as a consequence it improves the deconvolution.

Table 1. Assignment accuracy of MCSPCA (with imperfect physical map) and MCSPFA (with perfect physical map) on probe set 1

pooling	#pools	# true assigns	basic recall	MCSPCA			MCSPFA		
				recall	precision	time*	recall	precision	time*
$\{ \begin{smallmatrix} P = 13 \\ L = 3 \end{smallmatrix} \}$	39	14742	0.0103	0.199	0.2647	56	0.4857	0.4227	29
$\{ \begin{smallmatrix} P = 13 \\ L = 5 \end{smallmatrix} \}$	65	14742	0.2726	0.618	0.7668	41	0.9708	0.8511	409
$\{ \begin{smallmatrix} P = 47 \\ L = 2 \end{smallmatrix} \}$	94	14742	0.0173	0.4005	0.5236	9	0.8856	0.7626	83
$\{ \begin{smallmatrix} P = 47 \\ L = 3 \end{smallmatrix} \}$	141	14742	0.763	0.9069	0.9446	2	0.9991	0.9798	<1

Note: *The time reported is the running time in minutes for GLPK to solve the linear program on a 3GHz Pentium machine with 3GB memory. The running time for the randomized rounding step is negligible.

However, if we stick with the more realistic probe set 1 and noisy physical map, our algorithm still achieves 91% recall and 94% precision for the pooling $P = 47, L = 3$. Even more impressive is the amount of additional deconvolution achieved for the other 2-decodable pooling when compared to the basic deconvolution. For example, for $P = 13, L = 5$ the pooling is composed by only 65 pools, and thereby the basic deconvolution achieves just 27% recall. Our algorithm however achieves 62% recall with 77% precision. The results for the perfect map shows that our algorithm could potentially deconvolute all BAC-probe pairs with almost 100% precision (if the pooling is “powerful” enough).

Finally, in order to show the effectiveness of our randomized rounding scheme, Tables 3 and 4 report the total number of constraints, the optimal value OPT_f of the LP, and the number W of satisfied constraints. Note that the rounding scheme does not significantly affect the value of the objective function (see ratio OPT_f/W).

3.3. Experimental Results On the Barley Genome

The second dataset is related to the barley (*Hordeum vulgare*) project currently in progress at UC Riverside. The Barley BAC library used is a Morex library covering 6.3 genome equivalents¹⁵. Selected BACs from the BAC library were fingerprinted using a techniques that employs four different restriction enzymes, called high information content fingerprinting⁶. The physical map was constructed using FPC. The total number of BACs that were successfully fingerprinted and that were present in the physical map is 43,094. Among the set of BACs present in the physical map, about 20 have been fully sequenced. They will be used for validation of our algorithm. The Barley unigenes were obtained by assembling the ESTs downloaded from the NCBI EST database. The unigene assembly contains in total 26,743 contigs and 27,121 singletons.

Table 2. Assignment accuracy of MCSPCA (with imperfect physical map) and MCSPFA (with perfect physical map) on probe set 2

pooling	#pools	# true assigns	basic recall	MCSPCA			MCSPFA		
				recall	precision	time*	recall	precision	time*
$\{ \begin{smallmatrix} P = 13 \\ L = 3 \end{smallmatrix} \}$	39	14742	0.0121	0.2163	0.3182	26	0.625	0.6214	28
$\{ \begin{smallmatrix} P = 13 \\ L = 5 \end{smallmatrix} \}$	65	14742	0.3111	0.6488	0.8314	26	0.9984	0.9964	149
$\{ \begin{smallmatrix} P = 47 \\ L = 2 \end{smallmatrix} \}$	94	14742	0.0298	0.4348	0.6009	5	0.9971	0.9962	24
$\{ \begin{smallmatrix} P = 47 \\ L = 3 \end{smallmatrix} \}$	141	14742	0.8182	0.9285	0.9767	<1	0.9995	0.9997	<1

Note: *The time reported is the running time in minutes for GLPK to solve the linear program on a 3GHz Pentium machine with 3GB memory. The running time for the randomized rounding step is negligible.

Table 3. Performance of the randomized rounding scheme on probe set 1

pooling	# constraints	MCSPCA			MCSPFA		
		OPT_f	W	OPT_f/W	OPT_f	W	OPT_f/W
$P = 13, L = 3$	35071	28683	22615	0.7884	35033	30562	0.8724
$P = 13, L = 5$	58472	45220	41462	0.9169	58425	58277	0.9975
$P = 47, L = 2$	27591	21472	18633	0.8678	27567	26524	0.9622
$P = 47, L = 3$	41509	29458	29378	0.9973	41467	41467	1

Table 4. Performance of the randomized rounding scheme on probe set 2

pooling	# constraints	MCSPCA			MCSPFA		
		OPT_f	W	OPT_f/W	OPT_f	W	OPT_f/W
$P = 13, L = 3$	35102	27161	21567	0.7940	35089	31183	0.8887
$P = 13, L = 5$	58210	42795	39934	0.9331	58179	58179	1
$P = 47, L = 2$	27739	20127	17990	0.8938	27711	27711	1
$P = 47, L = 3$	41378	28567	28532	0.9988	41378	41338	0.9990

About a dozen research groups around the world contributed hybridization data. Each group designed probes for certain genes of interest and performed the hybridization experiments. Since those efforts were not centrally coordinated, the probe design and the pool design were completely *ad hoc*. The length of probe ranges from 36 nucleotides to a few hundreds bases. The number of unigenes that each pool represents also ranges from one to a few hundreds. We collected the data, and we transformed it into a list of constraints that we processed first with the basic deconvolution.

Recall that if we obtain an exact pair, the assignment is immediate. But if a constraint is non-exact, we cannot conclude much even if the size of $\mathbb{O}_{b,p}$ is very

small. However, intuitively those constraints for which $|\mathbb{O}_{b,p}|$ is small are the most informative.

In an attempt to filter out the noise and isolate the informative constraints we selected only those for which $|\mathbb{O}_{b,p}| \leq 50$. In total 14,796 constraints were chosen. Then, we focused only on the 5,327 BACs and the 2,263 unigenes that were involved in this selected set of constraints. We then used our MCSPCA method on this reduced set (along with the barley physical map produced by FPC) and we obtained 9,587 assignments. We cross-referenced these assignments to the small set of sequenced BACs and we determined that six of them were in common to the 5,327 BACs we selected. Our algorithm assigned eight unigenes to those 6 BACs, and six of them turned out to be correct by matching them to the sequences of 20 known BACs.

4. CONCLUSIONS

In this paper, we proposed a new method to solve the BAC-gene deconvolution problem. Our method compensates for a weaker pooling design by exploiting a physical map. The deconvolution problem is formulated as pair of combinatorial optimization problems, both of which are proved to be NP-complete. The combinatorial optimization problems are solved approximately via Integer Programming followed by Linear Programming relaxation and then randomized rounding. Our experimental results on both real and simulated data show that our method is very accurate in determining the correct mapping between unigenes and BACs. The right combination of combinatorial pooling and our method not only can dramatically reduce the number of pools required, but also can deconvolute the BAC-gene relationships almost perfectly.

5. FUTURE WORK

The time complexity for both our algorithms MCSPCA and MCSPFA are dominated by the step of solving the LPs. Among the data sets that we have tried, the largest LP takes several hours for GLPK to solve optimally. One way to speed up our algorithm is to rely on the Lagrangian relaxation technique to solve the LP only approximately. This will inevitably sacrifice the accuracy of the algorithms, but will dramatically increase the efficiency.

ACKNOWLEDGMENTS

This project was supported in part by NSF CAREER IIS-0447773, NIH LM008991-01, and NSF DBI-0321756. The authors thank Serdar Bozdag for providing the data for the simulation on the rice genome.

References

1. Barbazuk WB, Bedell JA, Rabinowicz PD. Reduced representation sequencing: a success in maize and a promise for other plant genomes. *Bioessays* 2005; **27**: 839–848

2. Du DZ, Hwang FK. Combinatorial Group Testing and its applications 2nd edition. *World Scientific* 2000
3. Dyachkov A, Hwang F, Macula A, *et al.* A Construction of Pooling Designs with Some Happy Surprises. *Journal of Computational Biology* 2005; **12**: 1129–1136
4. Flibotte S, Chiu R, Fjell C, *et al.* Automated ordering of fingerprinted clones. *Bioinformatics*, **20**: 1264–1271
5. Goemans MX, Williamson DP. New $\frac{3}{4}$ -Approximation Algorithms for the Maximum Satisfiability Problem. *SIAM Journal on Discrete Mathematics* 1994; **7**: 656–666
6. Luo MC, Thomasa C, Youa FM *et al.* High-throughput fingerprinting of bacterial artificial chromosomes using the snapshot labeling kit and sizing of restriction fragments by capillary electrophoresis. *Genomics* 2003; **82**: 378–389
7. Macula AJ. A simple construction of d -disjunct matrices with certain constant weights. *Discrete Mathematics* 1996; **162**: 311–312
8. Michael S. Introduction to the Theory of Computation. *International Thomson Publishing* 1996
9. Sandhu D, Gill KS. Gene-Containing Regions of Wheat and the Other Grass Genomes *Plant Physiology* 2002; **128**: 803-811
10. Soderlund C, Longden I, Mott R. FPC: a system for building contigs from restriction fingerprinted clones. *Computer Applications in the Biosciences* 1997; **135**, 523–535,
11. Sumner AT, de la Torre J, Stuppia L. The distribution of genes on chromosomes: a cytological approach *Journal of molecular evolution* 1993; **37**: 117–122
12. Thierry-Mieg N. A new pooling strategy for high-throughput screening: the Shifted Transversal Design. *BMC Bioinformatics* 2006; **7**:28 – 37
13. Vazirani VV. Approximation Algorithms *Springer* 2001
14. Yannakakis M. On the approximation of maximum satisfiability. *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms* 1992; 1–9
15. Yu Y, Tomkins JP, Waugh R, *et al.* A bacterial artificial chromosome library for barley (*Hordeum vulgare L.*) and the identification of clones containing putative resistance genes. *Theoretical and Applied Genetics* 2000; **101**: 1093–1099.
16. Zheng J, Close TJ, Jiang T, Lonardi S. Efficient Selection of Unique and Popular Oligos for Large EST Databases. *Bioinformatics* 2004; **20** : 2101–2112
17. Zheng J, Svensson JT, Madishetty K *et al.* Oligospawn: a web-based tool for the design of overgo probes from large unigene databases *BMC Bioinformatics* 2006; **7**: 7–15