A POSITIVE DETECTING ALGORITHM FOR DNA LIBRARY SCREENING BASED ON CCCP

Hiroaki Uehara* and Masakazu Jimbo**

We describe an algorithm for extracting as much information as possible from pooling experiments for library screening based on the concave-convex procedure (CCCP). Called the CCCP pool result decoder (CCPD), it is a positive clone detecting algorithm. Its performance is compared, by simulation, with the Bayesian network pool result decoder (BNPD) proposed by Uehara and Jimbo and the Markov chain pool result decoder (MCPD) proposed by Knill et al. in 1996.

Key words and phrases: BP, CCCP, design of experiment, DNA library screening, group testing, packing design, pooling experiment, two-stage test.

1. Introduction

To find a few positives among a large number of items, one can use group testing. In group testing, multiple items are assayed in groups. If a group has a negative outcome, all items contained in it are negative. This can reduce the total number of tests. On the other hand, if a group is positive, we know that the group contains at least one positive item. By designing many kinds of groups and by testing each of them, we obtain the results for all groups. After knowing the group results, we may be able to estimate which items are likely to be positive. For each of such items, we apply individual tests to determine whether it is positive or negative. Group testing has been used in medical, chemical, and electrical testing; drug screening; pollution control; multiaccess channel communication; and recently in gene assays like clone library screening, protein-protein interaction tests, and other subjects. See for example Du and Hwang (1999), Schliep and Rahmann (2006), Thierry-Mieg (2006), Klau et al. (2007), Thierry-Mieg and Bailly (2008).

In this paper we restrict ourselves to group testing for DNA library screening to give a concrete image of testing and to consider a specialized problem in clone library screening. However, the algorithm given in this paper can be applied to any other fields of group testing. In DNA library screening experiments there are a large number of clones, which are short strings of nucleotides A, T, G and C. Through the use of high-quality gene libraries, the study of gene functions has been developed into a very important research field. The gene libraries are obtained from extensive testing and screening of DNA clones. For each clone,
it is desirable to determine whether the clone contains a specific sequence of nucleotides. A clone is positive if it contains the specific sequence. A goal of DNA library screening is to identify all positive clones. For this purpose, group testing is often applied. Each group is called a pool, and the screening method is called a pooling experiment. A pooling experiment is advantageous when the proportion of positive clones is fairly small. The efficiency of pooling experiments has been studied by Barillot et al. (1991), Bruno et al. (1995), Du and Hwant (1999), Berger et al. (2000), Sham et al. (2002). Usually, if many kinds of clones are mixed in a pool, the possibility of a false negative may increase more than in the case of single clone assays. However, when the PCR (polymerase chain reaction) method is applied for the testing procedure, copies of target clones are repeatedly and selectively made. Thus it was claimed in Knill et al. (1996) that the possibility of false positive/negative is almost the same with the case of single clone if the number of distinct kinds of clones in a pool is at most several hundreds.

Each well of a microtiter plate is a pool of hundreds of kinds of clones. Complicated pooling experiments are performed robotically to avoid human errors. Therefore, it is necessary to develop the experimental software and hardware which manage sophisticated pooling experiments.

There are two types of group testing: adaptive or non-adaptive. In adaptive testing, we can make the next group after examining results of the previous experiment. But the analysis and experiments are often performed by different people or different organizations. In this case, nonadaptive group testing is required. That is, we must determine the design of all groups before the experiments. Here, we consider nonadaptive group testing.

From the viewpoint of positive detecting procedures based on the outcome of experiments, there are two kinds of group testing algorithms: combinatorial and probabilistic. In combinatorial group testing, it is often assumed that the number of positives among $n$ clones is either fixed or bounded by some fixed positive integer $d$, and a deterministic model is used. The main problem in combinatorial group testing is to construct a good pooling design consisting of various sets of clones so that positive clones can be distinguished from negative ones under some assumptions about the maximum number of positives. However, in a real experiment, there may be more positives than expected. Moreover, the outcome may have errors such as false positives or false negatives. In probabilistic group testing, we assume that a positive clone appears probabilistically according to a prior probability. The probabilistic algorithm detects positive clones in accordance with a probabilistic model. This approach can detect positives even if their number exceeds the maximum number of positives set by the pooling design under the existence of the observation errors of false positive/negative.

The aim of a pooling experiment is to reduce the number of tests required to find positive clones among a large number of clones under the condition that experimental or observation errors cannot be disregarded. That is, we should take into account the possibility of false positives and false negatives. There-
fore, pooling experiments require that positive clones be detected with a high probability even when errors exist. To detect positive clones correctly in pooling experiments, we focus on efficient positive clone detecting algorithms. Knill et al. (1996) found an algorithm based on the Markov Chain Monte Carlo (MCMC) method. They called it the Markov Chain pool result decoder (MCPD). Using another method, Uehara and Jimbo (2007) developed an efficient algorithm based on belief propagation (BP) in a Bayesian network when there are no short cycles in the Tanner graph of a pooling design. This algorithm was called the Bayesian network pool result decoder (BNPD).

In this paper we first introduce an algorithm based on the concave-convex procedure (CCCP) and explain the algorithm in terms of the minimization of the Bethe free energy. Our new algorithm is called the concave-convex procedure pool result decoder (CCPD). Secondly, by pursuing several kinds of simulations, we examine the performance of CCPD, BNPD and MCPD together with the combinatorial property of the pooling design. The performance of the algorithms are tested in the sense of the detection ability of all positive clones and the execution speed. Especially, it is shown that BNPD can not be used when there are many cycles of length four in the Tanner graph, and CCPD is a good alternative even when there are short cycles in the Tanner graph. Furthermore, by comparing CCPD with MCPD, it is shown that the execution time of CCPD is much faster than MCPD, and the detection ability is in the same level if the iteration number of MCPD is moderate. Finally, when there are many cycles of length four in the Tanner graph of a pooling design, we propose the usage of a hybrid algorithm of CCPD and MCPD to get more precise results within a short computation time.

2. Pooling experiments and their stochastic model

A pooling design is a method for planning a group test. In pooling experiments, each pool consists of many different clones. A collection of pools that consist of various combinations of clones is called a pooling design. When there are only a few positive clones, it may be possible to reduce the number of tests by designing an efficient pooling experiment.

We denote clones by \( c \) and pools by \( G \). Let \( C \) be the set of \( n \) clones and let \( \mathcal{G} \) be a collection of \( m \) pools. Each pool \( G \) in \( \mathcal{G} \) is a subset of \( C \) corresponding to clones in the pool. The incidence relation of \( C \) and \( \mathcal{G} \) is written as an \( m \times n \) \{0, 1\}-matrix or a bipartite graph. A pooling design is represented by such a bipartite graph. Let \( E = \{(c, G) \mid c \in C, G \in \mathcal{G} \text{ and } c \in G\} \), then a pair in \( E \) is regarded as an edge of a bipartite graph \((C, \mathcal{G}; E)\), which is called a Tanner graph. An example is shown in Fig. 1. The name “Tanner graph” comes from the field of coding theory. Here, we use the same terminology to make our algorithm correspond to that of “low density parity check” (LDPC) codes. The reader can find the corresponding relationship between group testing for DNA library screening and LDPC codes in Uehara and Jimbo (2007).
2.1. Stochastic model of pooling experiments

Let $X_c$ be a random variable such that $X_c = 1$ or 0 according to whether a clone $c$ is positive/negative. Usually, the prior probability $P(X_c = 1)$ is small, for example, $P(X_c = 1) = 0.0001$ or 0.001. Furthermore, let $Z_G$ be a random variable defined by $Z_G = \bigvee_{c \in G} X_c$, where $\bigvee_{c \in G} X_c$ implies the OR-sum of $X_c$'s in $G$. If $G$ includes only negative clones, $Z_G = 0$. On the other hand, if $G$ includes at least one positive clone, $Z_G = 1$. Let $S_G$ be a random variable representing the experimental observation for $G$. As in Knill et al. (1996), an observation $S_G$ of the response to a screening test is often represented by using the four level values 0 (negative), 1 (weak positive), 2 (medium positive) and 3 (strong positive). This is because the response of the experiment is automatically measured from the fluorescence strength. Therefore, we should take into account the error probability $P(S_G = s \mid Z_G = z)$; i.e., the probability of a false positive $P(S_G = 1, 2, 3 \mid Z_G = 0)$, and that of a false negative $P(S_G = 0 \mid Z_G = 1)$. Here, we assume that the $X_c$'s are independent, each observation $S_G$ depends only on $Z_G$, and the $S_G$'s are independent under the condition that all $Z_G$'s are known.

Let $X = (X_{c_1}, \ldots, X_{c_n})$ and $S = (S_{G_1}, \ldots, S_{G_m})$ be random vectors. In an experiment, when $S$ is measured, the marginal posterior probability for $X_c$ is

$$P(X_c = x \mid S = s) = \sum_{x \in \{0, 1\}^n \text{ s.t. } x_c = x} P(X = x \mid S = s)$$

$$= K \sum_{x \in \{0, 1\}^n \text{ s.t. } x_c = x} P(X = x) P(S = s \mid X = x)$$

$$= K \sum_{x \in \{0, 1\}^n \text{ s.t. } x_c = x} \prod_{x' \in \{0, 1\}^n \text{ s.t. } x_c = x} P(X_{c'} = x_{c'}) \prod_{G \in G} P(S_G = s_G \mid Z_G = z_G),$$

where $K = P(S = s)^{-1}$ is a constant, $z_G = \bigvee_{c \in G} x_c$ and $\sum_{x \in \{0, 1\}^n \text{ s.t. } x_c = x}$ means the sum of all $x = (x_{c_1}, \ldots, x_{c_n}) \in \{0, 1\}^n$ such that $x_c = x$. Moreover, it is assumed that the values $P(S_G = s_G \mid Z_G = z_G)$ are empirically known from the previous experiments.
Note that we need $2^{n-1}$ additions to calculate the probability $P(X_c = x \mid S = s)$ given by (2.1). Hence, as the number of clones $n$ becomes large, the number of calculations increases exponentially. Therefore, the algorithm used to calculate this value must be efficient. As an efficient algorithm, Knill et al. (1996) proposed MCPD. Recently, Uehara and Jimbo (2007) proposed BNPD and showed by simulations that it is efficient when there are no cycles of length four. For such a case, we propose another algorithm that is efficient in the calculation of the probability (2.1) even when there are cycles of length four.

3. Bethe free energy

Our new algorithm CCPD is based on the concave-convex procedure (CCCP) in Yuille (2002), and Yuille and Rangarajan (2003), which is a procedure for minimizing the Bethe free energy. Yedidia et al. (2001) clarified the relation between the Bethe free energy and the BP algorithm. They proved that the fixed points of the BP algorithm are equivalent to those of the zero-gradient constraints.

3.1. Definition of Bethe free energy

First, we give a general definition of the Bethe free energy. Let $I$ be an index set of nodes. For every $i \in I$, let $W_i$ be a finite set and consider a random variable $W_i$ on $W_i$. We also consider a random vector $W := (W_i)_{i \in I}$ on $W := \prod_{i \in I} W_i$ and an observation random vector $Y$.

Moreover, we denote the value of $W_i$ by $w_i$ and an observation vector by $y$. We assume that the posterior probability of $W$ is given as follows:

\begin{align}
P(W = w \mid Y = y) = K \prod_{(i,j) \in T} \psi_{ij}(w_i, w_j) \prod_{i \in I} \psi_i(w_i),
\end{align}

where $w := (w_i)_{i \in I}$. $\psi_i(w_i)$ and $\psi_{ij}(w_i, w_j)$ are potentials and $K$ is a normalization constant. Note that some $\psi_i(w_i)$ are functions of $w_i$ and $y_i$. Since $y$ is fixed, we explicitly specify only $w_i$ and omit $y_i$ from those functions to emphasize that $\psi_i(w_i)$ is a function of one value. $T$ is a subset of $I \times I$. Moreover, let $q := ((q_i(w_i))_{i \in I}, w_i \in W_i, (q_{ij}(w_i, w_j))_{(i,j) \in T}, (w_i, w_j) \in W_i \times W_j)$. Then the Bethe free energy of $P(W = w \mid Y = y)$ is defined by

\begin{align}
F_B(q) := \sum_{(i,j) \in T} \sum_{(w_i, w_j) \in W_i \times W_j} q_{ij}(w_i, w_j) \log \frac{q_{ij}(w_i, w_j)}{\phi_{ij}(w_i, w_j)} \\
- \sum_{i \in I} (|N(i)| - 1) \sum_{w_i \in W_i} q_i(w_i) \log \frac{q_i(w_i)}{\psi_i(w_i)},
\end{align}
where \( \phi_{ij}(w_i, w_j) := \psi_{ij}(w_i, w_j)\psi_i(w_i)\psi_j(w_j) \) and \( N(i) \) is an index set of nodes neighboring the \( i \)-th node \( W_i \) in a Tanner graph \((I, T)\). Assume that \( q_{ij}(w_i, w_j) = 0 \) if \( \psi(w_i, w_j) = 0 \) and that \( q_i(w_i) = 0 \) if \( \psi_i(w_i) = 0 \).

Our problem is to compute the conditional marginal probabilities \( P(W_i = w_i \mid Y = y) \) for all \( i \). But when the Tanner graph \((I, T)\) has short cycles, we cannot apply the belief propagation (BP) algorithm. Instead, we can utilize the “ CCCP algorithm,” which is a procedure for minimizing the Bethe free energy.

Thus, when the Tanner graph has no cycles, the beliefs minimizing the Bethe free energy correspond to the exact marginal probabilities. When it does have cycles, we cannot say that for certain. However, \( P(W_i = w_i \mid Y = y) \) and \( P(W_i = w_i, W_j = w_j \mid Y = y) \) can be estimated by finding \( q \) such that \( \nabla F_B(q) = 0 \) and \( F_B(q) \) becomes as small as possible.

3.2. Bethe free energy of a pooling experiment

For any \((c, G) \in E\), let \( U_c^G \) be a random variable such that \( U_c^G = 1 \) or 0 depending on \( x_c = 1 \) or 0. Further let \( U_G := (U_c^G)_{c \in G} \) and \( U := (U_G)_{G \in G} \). Then we have \( P(X = x, U = u \mid S = s) = P(X = x \mid S = s) \) if \( u_G = x_G \) for \( G \in \mathcal{G} \), otherwise 0, where \( x_G := (x_c)_{c \in G} \in \{0, 1\}^{|G|} \) is a subvector of \( x \) having elements \( x_c \) corresponding to \( c \in G \), \( u_G := (u_c^G)_{c \in G} \in \{0, 1\}^{|G|} \), and \( u := (u_G)_{G \in E} \).

By utilizing \( U \), we can represent the conditional joint probability \( P(X = x \mid S = s) \) by the form of (3.1):

\[
P(X = x, U = u \mid S = s) = K \prod_{G \in \mathcal{G}} \left( P(S_G = s_G \mid Z_G = \vee u_c^G) \prod_{c \in G} \delta(x_c, u_c^G) \right) \times \prod_{c \in C} P(X_c = x_c),
\]

(3.2)

where \( \delta(x, y) = 1 \) if \( x = y \), otherwise 0.

Now we define

\[
\psi_c(x_c) := P(X_c = x_c) \quad \text{for} \quad x_c \in \{0, 1\} \quad (c \in C),
\]

\[
\psi_G(u_G) := P(S_G = s_G \mid Z_G = \vee u_c^G) \quad \text{for} \quad u_G \in \{0, 1\}^{|G|} \quad (G \in \mathcal{G}),
\]

\[
\psi_{cG}(x_c, u_G) := \delta(x_c, u_c^G) \quad \text{for} \quad (x_c, u_G) \in \{0, 1\}^{1+|G|} \quad ((c, G) \in E).
\]

Then (3.2) can be written as

\[
P(X = x, U = u \mid S = s) = K \prod_{(c,G) \in E} \psi_{cG}(x_c, u_G) \prod_{c \in C} \psi_c(x_c) \prod_{G \in E} \psi_G(u_G).
\]
Now, let \( q_c(x_c) \), \( q_G(u_G) \) and \( q_{cG}(x_c, u_G) \) be probabilities that satisfy the following:

\[
\sum_{x_c \in \{0,1\}} q_c(x_c) = 1 \quad (c \in C), \quad \sum_{u_G \in \{0,1\}^{|G|}} q_G(u_G) = 1 \quad (G \in G),
\]

(3.4)

\[
\sum_{x_c \in \{0,1\}} q_{cG}(x_c, u_G) = q_G(u_G) \quad \text{for} \quad u_G \in \{0,1\}^{|G|} \quad ((c, G) \in E),
\]

\[
\sum_{u_G \in \{0,1\}^{|G|}} q_{cG}(x_c, u_G) = q_c(x_c) \quad \text{for} \quad x_c \in \{0,1\} \quad ((c, G) \in E).
\]

Moreover, let \((c, G)\) denote the set of pools that includes a clone \(c\), and let

\[
\phi_{cG}(x_c, u_G) := \psi_c(x_c) \psi_G(u_G) \psi_{cG}(x_c, u_G).
\]

Then, the Bethe free energy for \(P(X = x, U = u \mid S = s)\) is

\[
F_B(q) = \sum_{(c, G) \in E} \sum_{(x_c, u_G) \in \{0,1\}^{1+|G|}} q_{cG}(x_c, u_G) \log \frac{q_{cG}(x_c, u_G)}{\phi_{cG}(x_c, u_G)}
\]

(3.6)

\[
- \sum_{c \in C} (|\{(c)\} - 1) \sum_{x_c \in \{0,1\}} q_c(x_c) \log \frac{q_c(x_c)}{\psi_c(x_c)}
\]

\[
- \sum_{G \in G} (|G| - 1) \sum_{u_G \in \{0,1\}^{|G|}} q_G(u_G) \log \frac{q_G(u_G)}{\psi_G(u_G)},
\]

where \( q = ((q_c(x_c))_{c \in C}, x_c \in \{0,1\}, (q_G(u_G))_{G \in G}, u_G \in \{0,1\}^{|G|} \).

\( q_{cG}(x_c, u_G) \) whenever \( \psi_c(x_c), \psi_G(u_G) \) and \( \phi_{cG}(x_c, u_G) \) are zero, respectively. Then

\[
q_{cG}(x_c, u_G) = \delta(x_c, u_G^C) q_G(u_G)
\]

(3.7)

holds from (3.3) and (3.4).

In the rest of this paper, we assume that \( \psi_c(x_c) = P(X_c = x_c) \) is always non-zero. By noting (3.5) and (3.7) the Bethe free energy (3.6) is simplified to

\[
F_B(q) = \sum_{c \in C} \sum_{x_c \in \{0,1\}} q_c(x_c) \log \frac{q_c(x_c)}{\psi_c(x_c)}
\]

(3.8)

\[
+ \sum_{G \in G} \sum_{u_G \in \{0,1\}^{|G|}} q_G(u_G) \log \frac{q_G(u_G)}{\psi_G(u_G)}
\]

\[
- \sum_{c \in C} (|\{(c)\} - 1) \sum_{x_c \in \{0,1\}} q_c(x_c) \log q_c(x_c)
\]

and the conditions (3.4) are reduced to

\[
\sum_{x_c \in \{0,1\}} q_c(x_c) = 1 \quad (c \in C),
\]

(3.9)

\[
\sum_{u_G \in \{0,1\}^{|G|}} q_G(u_G) = q_c(x_c) \quad \text{for} \quad x_c \in \{0,1\} \quad ((c, G) \in E).
\]
Note that, in this case, \( q = ((q_c(x_c))_{c \in C, x_c \in \{0,1\}}, (q_G(u_G))_{G \in \mathcal{G}, u_G \in \{0,1\}^{|G|}} \).

4. Positive clone detecting algorithm

This section outlines the CCCP algorithm, which can be applied to any Bayesian network as in Yuille (2002). For example, CCCP is utilized for the decoding of LDPC codes in Shibuya et al. (2005).

4.1. Minimizing the Bethe free energy

CCCP is a procedure for finding \( q \) that minimizes the Bethe free energy (3.8) subject to the linear constraints (3.9) by utilizing the Lagrange multiplier method.

The following proposition was shown in Yuille (2002).

**Proposition 1.** Let \( q = (q_1, \ldots, q_N) \) and \( F(q) \) be an energy function (bounded below) of form \( F(q) = F_{\text{vex}}(q) + F_{\text{cave}}(q) \), where \( F_{\text{vex}}(q) = \sum_{i=1}^{N} q_i \log(q_i/\psi_i) \) and \( F_{\text{cave}}(q) \) are the convex and concave functions of \( q \), respectively. Consider \( F(q) \) subject to \( L \) linear constraints \( a_\ell \cdot q = b_\ell \) (\( \ell = 1, \ldots, L \)), where \( a_\ell := (a_{\ell 1}, \ldots, a_{\ell N}) \) and \( b_\ell \) is a constant. Then the algorithm \( q^{(t-1)} \mapsto q^{(t)} \) given as

\[
\nabla F_{\text{vex}}(q^{(t)}) = -\nabla F_{\text{cave}}(q^{(t-1)}) - \sum_{\ell=1}^{L} \alpha_\ell a_\ell,
\]

is guaranteed to monotonically decrease \( F(q^{(t)}) \) and hence to converge to an extremum of \( F(q) \). The \( \alpha_\ell \) (\( \ell = 1, \ldots, L \)) is a Lagrange multiplier for the \( \ell \)-th linear constraint and it is chosen to ensure that \( a_\ell \cdot q^{(t)} = b_\ell \). The update equation can be expressed as minimizing the convex energy function

\[
F^{(t)}(q^{(t)}) := q^{(t)} \cdot h + \sum_{i=1}^{N} q_i^{(t)} \log \frac{q_i^{(t)}}{\psi_i} + \sum_{\ell=1}^{L} \alpha_\ell (a_\ell \cdot q^{(t)} - b_\ell),
\]

where \( h := \nabla F_{\text{cave}}(q^{(t-1)}) \). The solution \( q^{(t)}(\alpha) := (q_1^{(t)}(\alpha), \ldots, q_N^{(t)}(\alpha)) \) with \( \alpha := (\alpha_1, \ldots, \alpha_L) \) is

\[
q_i^{(t)}(\alpha) = \psi_i e^{-(1+h_i)} \prod_{\ell=1}^{L} e^{-\alpha_\ell a_{\ell i}},
\]

where the Lagrange multipliers \( \{\alpha_\ell\}_{\ell=1}^{L} \) are constrained to maximize the concave dual energy

\[
\hat{F}^{(t)}(\alpha) := -\sum_{i=1}^{N} q_i^{(t)}(\alpha) - \sum_{\ell=1}^{L} \alpha_\ell b_\ell.
\]

Moreover, maximizing \( \hat{F}^{(t)}(\alpha) \) with respect to a specific \( \alpha_\ell \) enables us to exactly satisfy the corresponding constraint.
As seen in Proposition 1, the CCCP algorithm can be implemented as a double loop algorithm, i.e., the outer loop calculates (4.1) and the inner loop determines the value of $\alpha$ at which $\hat{F}^{(t)}(\alpha)$ of (4.2) is maximum.

### 4.2. Concave-convex positive detector

Now we apply the CCCP algorithm to our group testing problem.

We split $F_B(q)$ of (3.8) into the convex part $F_{\text{cave}}(q)$ and the concave part $F_{\text{cave}}(q)$ such that $F_B(q) = F_{\text{cave}}(q) + F_{\text{cave}}(q)$, which are defined as

$F_{\text{cave}}(q) := \sum_{c \in C} \sum_{x_c \in \{0,1\}} q_c(x_c) \log \frac{q_c(x_c)}{\psi_c(x_c)} + \sum_{G \in \mathcal{G}} \sum_{u_G \in \{0,1\}^{|G|}} q_G(u_G) \log \frac{q_G(u_G)}{\psi_G(u_G)},$

$F_{\text{cave}}(q) := -\sum_{c \in C} \left| \langle c \rangle \right| \sum_{x_c \in \{0,1\}} q_c(x_c) \log q_c(x_c).$

From the definition of $F_{\text{cave}}(q)$, we have

$h_c(x_c) := \frac{\partial F_{\text{cave}}(q^{(t-1)})}{\partial q_c^{(t-1)}(x_c)} = -\left| \langle c \rangle \right| \log q_c^{(t-1)}(x_c) + 1$

for $x_c \in \{0,1\}$ $(c \in C),$

$h_G(u_G) := \frac{\partial F_{\text{cave}}(q^{(t-1)})}{\partial q_G^{(t-1)}(u_G)} = 0$

for $u_G \in \{0,1\}^{|G|}$ $(G \in \mathcal{G}).$

Hence, for the Bethe free energy of (3.8) and the constraints of (3.9), the update equations of the outer loop of (4.1) are

$$q_c^{(t)}(x_c) = e^{-(1+\gamma_c)} \frac{\psi_c(x_c)q_c^{(t-1)}(x_c)^{\langle c \rangle}}{\prod_{G \in \langle c \rangle} e^{\lambda_G(x_c)}}$$

for $x_c \in \{0,1\}$ $(c \in C),$

$$q_G^{(t)}(u_G) = e^{-(1+\gamma_G)} \psi_G(u_G) \prod_{c \in G} e^{\lambda_c(x_c)^{G}}$$

for $u_G \in \{0,1\}^{|G|}$ $(G \in \mathcal{G}),$

where $\gamma_c,\gamma_G$ and $\lambda_c(x_c)$ are the Lagrange multipliers.

From (4.3) and (4.2), we have

$$\hat{F}^{(t)}(\alpha) = -\sum_{c \in C} \sum_{x_c \in \{0,1\}} e^{-(1+\gamma_c)} \frac{\psi_c(x_c)q_c^{(t-1)}(x_c)^{\langle c \rangle}}{\prod_{G \in \langle c \rangle} e^{\lambda_G(x_c)}}$$

$$-\sum_{G \in \mathcal{G}} \sum_{u_G \in \{0,1\}^{|G|}} e^{-(1+\gamma_G)} \psi_G(u_G) \prod_{c \in G} e^{\lambda_c(x_c)^{G}} - \sum_{c \in C} \gamma_c - \sum_{G \in \mathcal{G}} \gamma_G,$$

where $\alpha := ((\gamma_c)_{c \in C},(\gamma_G)_{G \in \mathcal{G}},(\lambda_c(x_c))_{x_c \in \{0,1\},(c,G) \in E}).$

Next, we solve

$$\frac{\partial \hat{F}^{(t)}(\alpha)}{\partial \alpha^\ell} = 0.$$
Similarly, let \( q_c \) denote the equation of \( x_c \), and \( \Lambda_{\gamma}(x_c) := e^{\Lambda_{\gamma}(x_c)} \). Then, for the Bethe free energy of (3.8) and the constraints of (3.9), the update equations of the inner loop are

\[
\begin{align*}
\Gamma_c &= e^{-1} \sum_{x_c \in \{0, 1\}} \frac{e^{\psi_c(x_c)} q_c^{(t-1)}(x_c) |(c)|}}{\prod_{G \in (c)} \Lambda_{\gamma}(x_c)} \quad (c \in C), \\
\Gamma_G &= e^{-1} \sum_{u_G \in \{0, 1\}^{[G]}} \frac{\psi_G(u_G) \prod_{c \in G} \Lambda_{\gamma}(u_G^c)}{(G \in G)}, \\
\Lambda_{\gamma}(x_c) &= \frac{\Gamma_c e^{\psi_c(x_c)} q_c^{(t-1)}(x_c) |(c)|}}{\prod_{G' \in (c) \setminus \{G\}} \Lambda_{\gamma}(u_{G'}^c)} \times \frac{1}{\sum_{u_G \in \{0, 1\}^{[G]}} \psi_G(u_G) \prod_{c' \in G \setminus \{c\}} \Lambda_{\gamma}(u_{c'}^G)} \quad (x_c \in \{0, 1\}, (c, G) \in E). 
\end{align*}
\]

Equations of (3.3) and (4.4) indicate that \( q_c^{(t-1)}(u_G) \) appears neither in the update equation of \( q_c^{(t)}(x_c) \) nor in those of \( \Gamma_c, \Gamma_G \) and \( \Lambda_{\gamma}(x_c) \). Therefore, we only have to update \( \{q_c^{(t)}(0), q_c^{(t)}(1)\} \in C \) in the outer loop.

To simplify the notation, we use \( p(s_G \mid z_G) := P(S_G = s_G \mid Z_G = z_G) \). Since \( \bigvee_{c \in G} x_c = 0 \) holds only when all \( x_c \)'s are 0, \( \Gamma_G \) in (4.4) is rewritten as follows:

\[
\begin{align*}
\Gamma_G &= e^{-1} \sum_{u_G \in \{0, 1\}^{[G]}} p(s_G \mid \bigvee_{c \in G} u_c^G) \prod_{c \in G} \Lambda_{\gamma}(u_c^G) \\
&= e^{-1} \left( p(s_G \mid 1) \sum_{u_G \in \{0, 1\}^{[G]}} \prod_{c \in G} \Lambda_{\gamma}(u_c^G) \\
&\quad + (p(s_G \mid 0) - p(s_G \mid 1)) \prod_{c \in G} \Lambda_{\gamma}(0) \right) \\
&= e^{-1} \left( p(s_G \mid 1) \prod_{c \in G} \sum_{u_c^G \in \{0, 1\}} \Lambda_{\gamma}(u_c^G) + (p(s_G \mid 0) - p(s_G \mid 1)) \prod_{c \in G} \Lambda_{\gamma}(0) \right).
\end{align*}
\]

Similarly, let

\[
R_{\gamma}(x) := \sum_{u_G \in \{0, 1\}^{[G]}} \psi_G(u_G) \prod_{c' \in G \setminus \{c\}} \Lambda_{\gamma}(u_{c'}^G).
\]

Denote that \( R_{\gamma}(x) \) is the last term of the denominator of \( \Lambda_{\gamma}(x_c)^2 \) in (4.4). Then

\[
R_{\gamma}(1) = \sum_{u_G \in \{0, 1\}^{[G]}} p(s_G \mid \bigvee_{c' \in G} u_{c'}^G) \prod_{c' \in G \setminus \{c\}} \Lambda_{\gamma}(u_{c'}^G).
\]
\[ p(s_G \mid 1) \sum_{\mathcal{G} \setminus \{c\} \in \{0,1\}^{G-1}} \prod_{c' \in \mathcal{G} \setminus \{c\}} \Lambda_{c'}(u_{c'}^G) \]

\[ = p(s_G \mid 1) \prod_{c' \in \mathcal{G} \setminus \{c\}} \sum_{u_{c'}^G \in \{0,1\}} \Lambda_{c'}(u_{c'}^G) \]

and

\[ R_{cG}(0) = \sum_{u_G \in \{0,1\}^{|G|}} p(s_G \mid \bigvee_{c' \in G} u_{c'}^G) \prod_{c' \in \mathcal{G} \setminus \{c\}} \Lambda_{c'}(u_{c'}^G) \]

\[ = p(s_G \mid 1) \sum_{u_G \in \{0,1\}^{|G|} \text{ s.t. } u_c^G = 0} \prod_{c' \in \mathcal{G} \setminus \{c\}} \Lambda_{c'}(u_{c'}^G) \]

\[ + (p(s_G \mid 0) - p(s_G \mid 1)) \prod_{c' \in \mathcal{G} \setminus \{c\}} \Lambda_{c'}(0) \]

\[ = R_{cG}(1) + (p(s_G \mid 0) - p(s_G \mid 1)) \prod_{c' \in \mathcal{G} \setminus \{c\}} \Lambda_{c'}(0) \]

are satisfied.

### 4.3. Algorithm

Our CCPD algorithm is shown in Fig. 2. Q, \( \Gamma \) and \( \Lambda \) correspond to \( q, e^\gamma \) and \( e^{\lambda} \) in (4.3) and (4.4), respectively. Now, for each clone \( c \in C \), choose \( P(X_c = x) \). For example, \( P(X_c = 1) = 0.001, 0.002 \) and \( P(X_c = 0) = 0.999, 0.998 \).

The CCPD algorithm consists of two nested loops: the inner loop and the outer loop. \( \Gamma_c, \Gamma_G \) and \( \Lambda_{cG}(x_c) \) are updated in the inner loop indexed by \( \tau \) from Step 2.1 to Step 2.4 in Fig. 2. \( Q_c(x) \) is updated in the outer loop indexed by \( t \) from Step 1 to Step 4. If a Tanner graph has no cycles, it is known in general that each estimate of the CCPD and BNPD algorithms converges to the correct marginal probability after sufficient iteration. If it has cycles, we cannot give any theoretical result about convergence or the convergence point. However, the longer cycle is desirable from the empirical point of view even if there are cycles in the Tanner graph. At the very least, the Tanner graph of a pooling design must not have short cycles of length four. To avoid cycles of length four, a pooling design should satisfy the property that any two clones are included together in at most one pool.

### 5. Simulation results

We compared the performance of CCPD with BNPD and MCPD. The programs can be downloaded from [1], [2] and [3], respectively.

---

[1] http://jim.math.cm.is.nagoya-u.ac.jp/~uehara/ccpd/
5.1. Simulation method

The simulations used the following procedure.

(i) Fix the number of clones \( n \), the number of pools \( m \), and the number of pools \( k \) containing each clone. In this paper we fix \( k = 4 \) because it was pointed out in Uehara and Jimbo (2007) that \( k \geq 4 \) is necessary for positive detection.

(ii) Set the prior probability \( P(X_c = 1) \) that each clone is positive.

(iii) Fix the number \( d \) of positive clones that can likely exist under the prior probability. For example, if \( n = 1000 \) and \( p = 0.002 \), then the expected

---

**Step 1** (Initialization of \( Q \)): Let \( Q^{(0)}_c(x) := P(X_c = x) \) for each \( c \in C \) and \( x \in \{0, 1\} \).

Let \( t := 1 \) and \( \epsilon > 0 \) be a small real number.

**Step 2.1** (Initialization of \( \Gamma \) and \( \Lambda \)): Let \( \Gamma^{(0)}_c := 1 \) and \( \Lambda^{(0)}_c(x) := 1 \) for each \( c \in C \) and \( G \in \mathcal{G} \).

Let \( \Lambda^{(0)}_c(x) := 1 \) for each \( (c, G) \in E \) and \( x \in \{0, 1\} \). Let \( \tau := 1 \).

**Step 2.2** (Update of \( \Lambda \)): For each edge \( (c, G) \in E \), update \( \Lambda^{(\tau)}_c(x) \) by

\[
\Lambda^{(\tau)}_c(x) := \frac{\Gamma^{(\tau-1)}_c e^{(c)} | P(X_c = x) Q^{(\tau-1)}_c(x) |}{\prod_{G' \in (c) \setminus \{G\}} \Lambda^{(\tau-1)}_{c'}(x)} \frac{1}{R^{(\tau-1)}_c(x)}, \quad x \in \{0, 1\},
\]

where \( \Lambda^{(\tau)}_c(x) := \Lambda^{(\tau)}_{cG}(x) \) or \( \Lambda^{(\tau)}_{cG}(x) \) depending on whether the computation of \( \Lambda^{(\tau)}_{cG}(x) \)

is finished or not.

**Step 2.3** (Update of \( \Gamma \)): For each \( c \in C \), update \( \Gamma^{(\tau)}_c(x) \) by

\[
\Gamma^{(\tau)}_c(x) := \frac{1}{\epsilon} \sum_{x \in \{0, 1\}} e^{(c)} | P(X_c = x) Q^{(\tau-1)}_c(x) | \prod_{G \in (c)} \Lambda^{(\tau)}_{cG}(x).
\]

For each \( G \in \mathcal{G} \), update \( \Gamma^{(\tau)}_G(x) \) by

\[
\Gamma^{(\tau)}_G(x) := \frac{1}{\epsilon} \left( (p(s_G \mid 0) - p(s_G \mid 1)) \prod_{c \in G} \Lambda^{(\tau)}_{cG}(0) + p(s_G \mid 1) \prod_{c \in G} x \sum \Lambda^{(\tau)}_{cG}(x) \right).
\]

Set \( \tau := \tau + 1 \).

**Step 2.4** (Iteration of inner loop): Iterate Step 2.2 and Step 2.3 several times.

**Step 3** (Update of \( Q \)): For each \( c \in C \) and \( x \in \{0, 1\} \),

\[
Q^{(t)}_c(x) := \frac{1}{\epsilon} \frac{1}{\Gamma^{(\tau-1)}_c} e^{(c)} | P(X_c = x) Q^{(\tau-1)}_c(x) | \prod_{G \in (c)} \Lambda^{(\tau-1)}_{cG}(x).
\]

**Step 4** (Iteration of outer loop): If \( |Q^{(t-1)}_c(x) - Q^{(t)}_c(x)| \leq \epsilon \) for each \( c \in C \) and \( x \in \{0, 1\} \),

go to Step 5; otherwise, set \( t := t + 1 \) and go to Step 2.

**Step 5** (Output of marginal probability): Output \( Q^{(\tau)}_c(x) \) as the result.

---

Figure 2. CCPD algorithm.
number of positive clones is 2, so \( d = 1, 2, 3, \) or 4 may often occur. Here, we fix \( d = 1, 2, 3, 4. \)

(iv) Choose \( d \) positive clones randomly.

(v) The experimental results \( S_G \) for pool \( G \) are determined randomly according to the true values \( Z_G \)'s and the conditional probability \( P(S_G = s_G \mid Z_G = z_G) \). These conditional probabilities are given beforehand.

(vi) The experimental results obtained in (v) are passed to each algorithm BNPD, MCPD and CCPD, and the probability of positiveness \( P(X_c = 1 \mid S = s) \) is computed for each clone. The clones are then sorted according as the probability.

(vii) Repeat (iii)–(vi), 10000 times for each \( d \).

5.2. Probability of false positive and false negative

In our simulation, the prior probability for each clone being positive was set to \( P(X_c = 1) = 0.002 \). Though the probability \( P(X_c = 1) \) is small, it is often that the number of positive clones \( d \) is more than one because the clones are copied from multiple DNA sequences. In the real experiment of Knill et al. (1996), they estimated the number of positive clones are about 2.6 in mean among 1298 kinds of clones. We considered four cases when the actual numbers of positive clones were \( d = 1, 2, 3 \) and 4. For a given number \( d \) of true positive clones, we chose true positive clones randomly from 1298 clones. The conditional probability of a false positive or false negative for the experiment was obtained from the results of an actual DNA library screening performed by Knill et al. (1996). They fixed the conditional probabilities as follows:

\[
\begin{align*}
\text{(P1)} & \quad p(0 \mid 0) = 0.871, \quad p(1 \mid 0) = 0.016, \quad p(2 \mid 0) = 0.035, \quad p(3 \mid 0) = 0.078, \\
p(0 \mid 1) = 0.05, & \quad p(1 \mid 1) = 0.11, \quad p(2 \mid 1) = 0.27, \quad p(3 \mid 1) = 0.57.
\end{align*}
\]

The reader finds that the probabilities of false positive/negative in (P1) are considerably high. Especially, the probability of false negative is higher than that of false positive. The probability highly depends on the circumstance of the experiment and the error probability (P1) may be improved by the development of the experimental technique, but the error probability of false negative is usually higher than that of false positive in the case of DNA library screening. Since this probabilities (P1) were estimated from a real experiment, there must be some reason why the error pattern of (P1) occurred. The reader, however, may consider that the error probabilities in (P1) do not seem to be natural because the monotoneity

\[
p(0 \mid 0) > p(1 \mid 0) > p(2 \mid 0) > p(3 \mid 0)
\]

is not satisfied. Thus, we also treated the following artificial errors besides (P1):

\[
\begin{align*}
\text{(P2)} & \quad p(0 \mid 0) = 0.856, \quad p(1 \mid 0) = 0.126, \quad p(2 \mid 0) = 0.016, \quad p(3 \mid 0) = 0.002, \\
p(0 \mid 1) = 0.021, & \quad p(1 \mid 1) = 0.154, \quad p(2 \mid 1) = 0.288, \quad p(3 \mid 1) = 0.537.
\end{align*}
\]
The probabilities (P2) are given by

\[ p(0 \mid 0) = 1 - q - q^2 - q^3, \quad p(1 \mid 0) = q, \]
\[ p(2 \mid 0) = q^2, \quad p(3 \mid 0) = q^3, \]
\[ p(0 \mid 1) = 1 - q' - q'^2 - q'^3, \quad p(1 \mid 1) = q'^3, \]
\[ p(2 \mid 1) = q'^2, \quad p(3 \mid 1) = q', \]

(5.1)

where \( q = 0.126 \) and \( q' = 0.537 \). The values \( q \) and \( q' \) were set so that the probabilities were similar to \( p(0 \mid 0), p(1 \mid 1), p(2 \mid 1) \) and \( p(3 \mid 1) \) of (P1).

Moreover, we also treated the following artificially chosen values of errors:

\[ p(0 \mid 0) = 0.825, \quad p(1 \mid 0) = 0.100, \quad p(2 \mid 0) = 0.050, \quad p(3 \mid 0) = 0.025, \]
\[ p(0 \mid 1) = 0.055, \quad p(1 \mid 1) = 0.135, \quad p(2 \mid 1) = 0.270, \quad p(3 \mid 1) = 0.540. \]

The probabilities (P3) are given by

\[ p(0 \mid 0) = 1 - 7r, \quad p(1 \mid 0) = 4r, \quad p(2 \mid 0) = 2r, \quad p(3 \mid 0) = r, \]
\[ p(0 \mid 1) = 1 - 7r', \quad p(1 \mid 1) = r', \quad p(2 \mid 1) = 2r', \quad p(3 \mid 1) = 4r', \]

where \( r = 0.025 \) and \( r' = 0.135 \). The values \( r \) and \( r' \) were set so that the probabilities were similar to \( p(0 \mid 0), p(1 \mid 1), p(2 \mid 1) \) and \( p(3 \mid 1) \) of (P1).

5.3. Simulation 1: Comparison of detection performance of CCPD, BNPD and MCPD

First, we compared the abilities of the algorithms to detect true positive clones for \( n = 1298 \). The number \( n = 1298 \) was chosen to compare the simulation with that of Knill et al. (1996). Note that the Tanner graph corresponding to their pooling design had cycles of length four. To utilize a pooling design satisfying the condition that a Tanner graph has no cycles of length four, we introduce a combinatorial notion of “packing design.” An \((m, k)\)-packing is a pair \((V, B)\), where \( V \) is a set of \( m \) elements (called points, which correspond to pools) and \( B \) is a collection of \( k \)-element subsets of \( V \) (called blocks, corresponding to clones), such that every pair of \( V \) occurs in at most one block in \( B \). When a packing design is utilized as a pooling design, the Tanner graph has no cycles of length four by its definition; i.e., any two clones are included together in at most one pool. In our simulation a clone corresponded to a block, and a pool corresponded to a point. We utilized a packing design so that the BNPD algorithm was expected to converge. The case when the Tanner graph has cycles of length four is treated in Subsection 5.5.

In order that the Tanner graph of a pooling design has no cycles of length four, \( m(m - 1) \geq nk(k - 1) \) must be satisfied. Thus, at least 125 pools are necessary. If \( m \) is a prime number, we can adopt an easy method to generate an \((m, n)\)-packing by utilizing the finite field \( GF(m) \). Let \( m = 131 \), which is a prime. We utilized a \((131, 4)\)-packing, which can be generated by the finite field \( GF(131) \). The base blocks of the packing are

\[ \{0, 1, 3, 7\}, \{0, 70, 79, 97\}, \{0, 28, 53, 109\}, \{0, 32, 42, 126\}, \]
{0, 13, 43, 58}, {0, 8, 19, 48}, {0, 20, 36, 85}, {0, 31, 55, 90},
{0, 12, 51, 74}, {0, 33, 54, 71} (mod 131).

All blocks can be generated by adding 1 modulo 131. Then, in total, there are 131 × 10 blocks. From these 1310 blocks we delete 12 blocks and get 1298 blocks. Hence, this pooling design has \( n = 1298 \) clones and \( m = 131 \) pools.

First, to see the convergence, let \( d = 1 \). By using error probability (P1), we executed a single simulation. The convergences of BNPD and MCPD are shown in Fig. 3. In each figure, the convergences of conditional probabilities are given for two clones: one clone has the maximum probability and the other is randomly chosen. Figure 3(1) indicates that CCPD converged within about 30 iterations. Figure 3(2) indicates that BNPD converged within a few iterations. Figure 3(3) indicates that MCPD was stable after around 2000 iterations, but it did not completely converge even after 10000 iterations. Actually, in the MCPD program of Knill et al. (1996) the default setting of the iteration number is 10000. We denote the MCPD procedure with \( t \) times iteration by MCPD\( t \). When \( t = 10000 \), we simply write MCPD instead of MCPD10000. The convergences are also examined in the next subsection for a larger number of clones.

Second, we repeated the simulations 10000 times for each case of positive clones \( d = 1, 2, 3, 4 \) given the error probability (P1). The same was also done for each of the error probabilities (P2) and (P3).

In our simulation for the number of positive clones \( d \), let \( n_i(d, A) \) be the ranking where the \( d \)-th positive clone appears in the \( i \)-th simulation when clones are ordered according to the value of \( P(X_c = 1 \mid S = s) \) if algorithm \( A \) is utilized. Among the 10000 simulations we sorted \( \{n_i(d, A)\}_{i=1}^{10000} \) in ascending order. Now let \( N_d(\rho, A) \) be the number of clones to be tested in the second individual tests so that all of the \( d \) positive clones can be detected with probability \( \rho \). \( N_d(\rho, A) \) can be estimated as the 10000\( \rho \)-th value in the ordered sequence \( \{n_i(d, A)\} \).

In a real pooling experiment, after getting the result of each pool, we proceed to the second stage; that is, clones are tested from the highest rank to the lower rank. Thus, to reduce the number of individual experiments in the second stage, it is desirable to obtain a good \( N_d(\rho, A) \). That is, \( N_d(\rho, A) \) may differ not only according to the pooling experiment but also according to the detecting algorithm \( A \). If \( N_d(\rho, A_1) \leq N_d(\rho, A_2) \) holds for any \( \rho \), then algorithm \( A_1 \) is
considered to be better than $A_2$. We compare $N_d(\rho, \text{CCPD})$ with $N_d(\rho, \text{BNPD})$ and $N_d(\rho, \text{MCPD})$.

Graphs of $N_d(\rho, \text{CCPD})$, $N_d(\rho, \text{BNPD})$ and $N_d(\rho, \text{MCPD}_t)$ for (P1), (P2) and (P3) are shown in Fig. 4. This figure indicates that the numbers of clones to be tested in the second stage are almost the same between CCPD and BNPD. On the other hand, if the iteration number $t$ of MCPD$_t$ is within 20000, the value $N_d$ of MCPD$_t$ is larger than those of CCPD or BNPD when $\rho$ is large (about 98–99%) and $d = 3, 4$, except for the case of (P2). In (P2), the error decreases exponentially as is in (5.1).

Values of $N_d(\rho, \text{CCPD})$, $N_d(\rho, \text{BNPD})$ and $N_d(\rho, \text{MCPD}_t)$ for (P1), (P2) and (P3) are shown in Table 1. Table 1 indicates that the detection performances of MCPD$_t$ depends on the iteration number $t$. In order to assure the same accuracy with CCPD/BNPD, MCPD needed higher numbers of iterations when $d$ and $\rho$ were larger. If the iteration number $t$ of MCPD was fixed, the detection performances of CCPD and BNPD for positive clones were better than that of MCPD as $d$ and $\rho$ were larger.

Next, we consider the case when the number of clones is about 10000. We utilized a (349, 4)-packing design whose number of clones is 10121. Hence, we consider the case of $(n, m) = (10121, 349)$.

In the case of $n = 10121$, the convergences for BNPD and MCPD are shown in Fig. 5. Similarly to the case of $n = 1298$, CCPD converged within about 40 iterations, BNPD converged within a few, and MCPD took at least 3000 to
Table 1. Detectability of positive clones ($n = 1298$, $m = 131$).

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>Error probability (P1)</th>
<th>Error probability (P2)</th>
<th>Error probability (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>90% 95% 98% 99%</td>
<td>90% 95% 98% 99%</td>
<td>90% 95% 98% 99%</td>
</tr>
<tr>
<td>$N_1(\rho, \text{CCPD})$</td>
<td>6 12 22 41</td>
<td>1 1 2 3</td>
<td>4 9 21 41</td>
</tr>
<tr>
<td>$N_1(\rho, \text{BNPD})$</td>
<td>6 13 22 42</td>
<td>1 1 2 3</td>
<td>4 8 21 41</td>
</tr>
<tr>
<td>$N_1(\rho, \text{MCPD})$</td>
<td>6 12 23 44</td>
<td>1 1 2 3</td>
<td>4 9 21 43</td>
</tr>
<tr>
<td>$N_1(\rho, \text{MCPD20000})$</td>
<td>7 13 22 42</td>
<td>1 1 2 3</td>
<td>4 9 20 41</td>
</tr>
<tr>
<td>$N_1(\rho, \text{MCPD40000})$</td>
<td>6 12 22 41</td>
<td>1 1 2 3</td>
<td>4 8 21 42</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>Error probability (P1)</th>
<th>Error probability (P2)</th>
<th>Error probability (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>90% 95% 98% 99%</td>
<td>90% 95% 98% 99%</td>
<td>90% 95% 98% 99%</td>
</tr>
<tr>
<td>$N_2(\rho, \text{CCPD})$</td>
<td>15 24 46 90</td>
<td>2 3 5 8</td>
<td>12 23 53 89</td>
</tr>
<tr>
<td>$N_2(\rho, \text{BNPD})$</td>
<td>15 24 47 85</td>
<td>2 3 5 8</td>
<td>12 23 53 89</td>
</tr>
<tr>
<td>$N_2(\rho, \text{MCPD})$</td>
<td>15 24 50 107</td>
<td>2 3 5 8</td>
<td>12 24 56 105</td>
</tr>
<tr>
<td>$N_2(\rho, \text{MCPD20000})$</td>
<td>15 24 49 91</td>
<td>2 3 5 8</td>
<td>12 24 54 93</td>
</tr>
<tr>
<td>$N_2(\rho, \text{MCPD40000})$</td>
<td>15 24 47 91</td>
<td>2 3 5 8</td>
<td>12 24 53 91</td>
</tr>
<tr>
<td>$N_3(\rho, \text{CCPD})$</td>
<td>25 39 82 128</td>
<td>4 5 9 14</td>
<td>24 43 88 126</td>
</tr>
<tr>
<td>$N_3(\rho, \text{BNPD})$</td>
<td>25 39 82 129</td>
<td>4 5 9 14</td>
<td>24 44 87 127</td>
</tr>
<tr>
<td>$N_3(\rho, \text{MCPD})$</td>
<td>25 41 99 164</td>
<td>4 5 9 15</td>
<td>24 45 95 169</td>
</tr>
<tr>
<td>$N_3(\rho, \text{MCPD20000})$</td>
<td>25 40 87 143</td>
<td>4 5 9 14</td>
<td>24 44 90 142</td>
</tr>
<tr>
<td>$N_3(\rho, \text{MCPD40000})$</td>
<td>25 40 87 131</td>
<td>4 5 9 15</td>
<td>24 44 88 134</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>Error probability (P1)</th>
<th>Error probability (P2)</th>
<th>Error probability (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>90% 95% 98% 99%</td>
<td>90% 95% 98% 99%</td>
<td>90% 95% 98% 99%</td>
</tr>
<tr>
<td>$N_4(\rho, \text{CCPD})$</td>
<td>37 56 115 179</td>
<td>6 9 17 26</td>
<td>40 68 124 183</td>
</tr>
<tr>
<td>$N_4(\rho, \text{BNPD})$</td>
<td>37 56 116 179</td>
<td>6 9 17 25</td>
<td>40 68 123 183</td>
</tr>
<tr>
<td>$N_4(\rho, \text{MCPD})$</td>
<td>37 59 140 228</td>
<td>6 9 17 29</td>
<td>40 69 145 259</td>
</tr>
<tr>
<td>$N_4(\rho, \text{MCPD20000})$</td>
<td>37 58 126 185</td>
<td>6 9 16 27</td>
<td>40 68 135 206</td>
</tr>
<tr>
<td>$N_4(\rho, \text{MCPD40000})$</td>
<td>37 59 119 179</td>
<td>6 9 16 26</td>
<td>39 68 129 198</td>
</tr>
</tbody>
</table>

10000 iterations. Thus, the number of iterations of each of the three algorithms did not depend much on the number of clones.

The simulation results are shown Table 2 for error probability (P1) ($n = 10121$, $m = 349$).

<table>
<thead>
<tr>
<th>$\rho$</th>
<th>Error probability (P1)</th>
<th>Error probability (P2)</th>
<th>Error probability (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>90% 95% 98% 99%</td>
<td>90% 95% 98% 99%</td>
<td>90% 95% 98% 99%</td>
</tr>
<tr>
<td>$N_1(\rho, \text{CCPD})$</td>
<td>43 73 115 258</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_1(\rho, \text{BNPD})$</td>
<td>43 73 115 257</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_1(\rho, \text{MCPD})$</td>
<td>42 73 114 272</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_2(\rho, \text{CCPD})$</td>
<td>82 120 358 637</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_2(\rho, \text{BNPD})$</td>
<td>82 120 354 634</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_2(\rho, \text{MCPD})$</td>
<td>82 120 374 712</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_3(\rho, \text{CCPD})$</td>
<td>115 168 523 833</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_3(\rho, \text{BNPD})$</td>
<td>115 169 520 833</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_3(\rho, \text{MCPD})$</td>
<td>115 170 594 959</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_4(\rho, \text{CCPD})$</td>
<td>146 251 730 980</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_4(\rho, \text{BNPD})$</td>
<td>145 250 730 984</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_4(\rho, \text{MCPD})$</td>
<td>147 249 794 1847</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Moreover, by comparing these results to the case of \((n, m) = (1298, 131)\), it is observed that the efficiency of group testing is higher as the number of clones is larger. In fact, when \(d = 4\), \(\rho = 99\%\) and error probability was \((P1)\), the ratios of clones to be tested in the second stage were about \(179/1298 \approx 0.137\) for \(n = 1298\) and \(980/10121 \approx 0.097\) for \(n = 10121\).

### 5.4. Simulation 2: Comparison of execution speed

We performed simulations with \(n = 1298, 3088, 6371, 10121, 30050\) by utilizing pooling designs generated by packing. For each \(n\) and for each algorithm, we compared computation times. Note that in this case the pooling design had no cycles of length four. We found that CCPD was about ten times faster than MCPD in the case when \(t = 10000\), and about thirty times slower than BNPD, as shown in Table 3 and Fig. 6. The execution speeds of these algorithms were proportional to the number of clones. Also, the execution speed of MCPD was proportional to the number of iterations \(t\). Note that the vertical axis of Fig. 6 is on a logarithmic scale. Table 4 indicates the number of operations in one iteration.

As a result, BNPD was the most efficient algorithm when there were no cycles of length four.

#### Table 3. Execution time.

<table>
<thead>
<tr>
<th>(n)</th>
<th>CCPD</th>
<th>BNPD</th>
<th>MCPD</th>
<th>MCPD 20000</th>
<th>MCPD 40000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1298</td>
<td>0.28</td>
<td>0.01</td>
<td>2.50</td>
<td>4.78</td>
<td>9.32</td>
</tr>
<tr>
<td>3088</td>
<td>0.85</td>
<td>0.02</td>
<td>5.97</td>
<td>11.42</td>
<td>22.23</td>
</tr>
<tr>
<td>6371</td>
<td>1.69</td>
<td>0.04</td>
<td>12.52</td>
<td>24.04</td>
<td>46.61</td>
</tr>
<tr>
<td>10121</td>
<td>3.35</td>
<td>0.06</td>
<td>19.67</td>
<td>37.6</td>
<td>73.44</td>
</tr>
<tr>
<td>30050</td>
<td>11.30</td>
<td>0.18</td>
<td>59.14</td>
<td>112.74</td>
<td>220.14</td>
</tr>
</tbody>
</table>

CPU: Intel\textsuperscript{®} Xeon\textsuperscript{TM} 3.06 GHz

OS: Red Hat Linux 9

![Figure 6. Execution time.](image)
Table 4. Numbers of operations in one iteration.

<table>
<thead>
<tr>
<th>Operation</th>
<th>CCPD</th>
<th>BNPD</th>
<th>MCPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>addition</td>
<td>$(2</td>
<td>E</td>
<td>+ n + 2m)\tau$</td>
</tr>
<tr>
<td>subtraction</td>
<td>$0$</td>
<td>$2</td>
<td>E</td>
</tr>
<tr>
<td>multiplication</td>
<td>$(10</td>
<td>E</td>
<td>+ 8)\tau + 2</td>
</tr>
<tr>
<td>division</td>
<td>$(2</td>
<td>E</td>
<td>+ 3n + m)\tau + 2$</td>
</tr>
<tr>
<td>exponentiation</td>
<td>$(2</td>
<td>E</td>
<td>+ 2n)\tau + 2$</td>
</tr>
<tr>
<td>square root</td>
<td>$2</td>
<td>E</td>
<td>\tau$</td>
</tr>
<tr>
<td>random number</td>
<td>$0$</td>
<td>$0$</td>
<td>$n$</td>
</tr>
</tbody>
</table>

$\tau$ is the iteration number of inner loop.

Table 5. Numbers of simulations that converged among 10000 simulations of BNPD ($n = 1298$, $m = 47$).

<table>
<thead>
<tr>
<th>Error probability</th>
<th>(P1)</th>
<th>(P2)</th>
<th>(P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d = 1$</td>
<td>8991</td>
<td>9777</td>
<td>9349</td>
</tr>
<tr>
<td>$d = 2$</td>
<td>5244</td>
<td>7677</td>
<td>6121</td>
</tr>
<tr>
<td>$d = 3$</td>
<td>1478</td>
<td>2724</td>
<td>2232</td>
</tr>
<tr>
<td>$d = 4$</td>
<td>232</td>
<td>373</td>
<td>511</td>
</tr>
</tbody>
</table>

5.5. Simulation 3: The case when the Tanner graph has cycles of length four

We considered what happens when the Tanner graph has cycles of length four. We performed simulations as follows. Choose $n = 1298$ as in the first simulation in Subsection 5.3. We compared the simulation result with the real pooling experiment reported in Knill et al. (1996) for 1298 kinds of clones of human chromosome 16 with 47 pools. In this case, the Tanner graph of the design had cycles of length four, as was mentioned in Subsection 5.3. That is, there were two clones that were included together in more than two pools. We utilized the same pooling design as Knill et al. (1996). And we performed 10000 simulations to examine the performance of the CCPD algorithm.

In this simulation, we could not expect good performance from the BNPD algorithm because the Tanner graph of the design had cycles of length four. Actually, when the BNPD algorithm was applied, the numbers of simulations that converged after 100 iterations among 10000 simulations decreased severely, as the number of positives $d$ increased (see Table 5). Thus, in this case it was not adequate to utilize the BNPD algorithm. On the other hand, CCPD always converges even if the Tanner graph of a pooling design has cycles of length four. We compared the performance of CCPD and MCPD.

Table 6 represents the detectability of positive clones for CCPD and MCPD. In Table 6, similar to the cases of $(n, m) = (1298, 131)$ and $(10121, 349)$, CCPD has some advantage when the iteration number $t$ was small. The accuracy of MCPD$t$ was getting better as the iteration number $t$ was getting larger. Also the iteration number $t$ necessary to give the same accuracy with CCPD increases as
Table 6. Detectability of positive clones \((n = 1298, m = 47)\).

<table>
<thead>
<tr>
<th>(\rho)</th>
<th>Error probability (P1)</th>
<th>Error probability (P2)</th>
<th>Error probability (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N_1(\rho, \text{CCPD}))</td>
<td>10 20 37 59</td>
<td>1 2 4 7</td>
<td>7 16 34 60</td>
</tr>
<tr>
<td>(N_1(\rho, \text{MCPD}))</td>
<td>10 19 36 60</td>
<td>1 2 4 7</td>
<td>7 15 35 64</td>
</tr>
<tr>
<td>(N_2(\rho, \text{CCPD}))</td>
<td>38 63 111 163</td>
<td>6 12 23 39</td>
<td>35 62 111 170</td>
</tr>
<tr>
<td>(N_2(\rho, \text{MCPD}))</td>
<td>37 65 122 211</td>
<td>5 9 20 37</td>
<td>33 61 117 211</td>
</tr>
<tr>
<td>(N_2(\rho, \text{MCPD20000}))</td>
<td>37 63 116 185</td>
<td>5 9 21 32</td>
<td>34 61 115 180</td>
</tr>
<tr>
<td>(N_2(\rho, \text{MCPD100000}))</td>
<td>37 63 111 166</td>
<td>5 9 19 31</td>
<td>34 60 111 166</td>
</tr>
<tr>
<td>(N_3(\rho, \text{CCPD}))</td>
<td>91 134 217 314</td>
<td>25 44 75 108</td>
<td>97 151 258 341</td>
</tr>
<tr>
<td>(N_3(\rho, \text{MCPD}))</td>
<td>91 139 251 458</td>
<td>22 37 72 133</td>
<td>97 161 327 684</td>
</tr>
<tr>
<td>(N_3(\rho, \text{MCPD20000}))</td>
<td>93 136 237 362</td>
<td>22 37 67 107</td>
<td>98 156 277 423</td>
</tr>
<tr>
<td>(N_3(\rho, \text{MCPD100000}))</td>
<td>89 132 223 319</td>
<td>22 36 67 99</td>
<td>95 153 260 365</td>
</tr>
<tr>
<td>(N_4(\rho, \text{CCPD}))</td>
<td>163 230 365 453</td>
<td>68 102 151 207</td>
<td>183 266 411 498</td>
</tr>
<tr>
<td>(N_4(\rho, \text{MCPD}))</td>
<td>165 232 473 839</td>
<td>66 104 168 267</td>
<td>185 284 519 863</td>
</tr>
<tr>
<td>(N_4(\rho, \text{MCPD20000}))</td>
<td>162 231 391 582</td>
<td>64 101 153 216</td>
<td>184 275 471 684</td>
</tr>
<tr>
<td>(N_4(\rho, \text{MCPD100000}))</td>
<td>160 229 371 476</td>
<td>62 96 146 194</td>
<td>182 270 420 522</td>
</tr>
</tbody>
</table>

\(d\) and \(\rho\) increase in each error probability.

In this case, the accuracy of MCPD seems to exceed that of CCPD after sufficient iteration, though the differences are not so large. This may be because of the magnitude of biases between the true value of \(P(X_c = 1 \mid S = s)\) and the convergent point of CCPD/BNPD. In general it is known that if there are many short cycles in a Bayesian network, the convergent point may have some bias between the true value. On the other hand, the convergent point of MCPD is not affected by the cycle structure of the Tanner graph.

Noting this result, Kanamori et al. (2008) theoretically examined that the first (dominant) term of the bias of \(P(X_c = 1 \mid S = s)\) between the true value and the convergent point of BNPD/CCPD is 0 when there are no cycles of length four in the Tanner graph, whereas it may not be 0 in the case when there are cycles of length four.

Hence, this simulation result indicates that it is better to use a hybrid algorithm of CCPD and MCPD in the case when we can not use a packing design. That is, firstly, we apply the CCPD algorithm to obtain values close to \(P(X_c = 1 \mid S = s)\) and secondly, we utilize the MCPD algorithm by setting the result of CCPD as a starting value of the algorithm and continue MCPD, as long as we can spend the time. By utilizing the result of CCPD we can pass only the clones which are likely to be positive to the MCPD algorithm. By using this procedure we can save computation time.

6. Conclusion

In conclusion, in the case when we want to find all positives with high probability, the MCPD algorithm needs a comparatively long execution time to assure the same accuracy as BNPD or CCPD. Thus to find all positives with high probability, BNPD or CCPD may be efficient in the sense of execution speed.
For the usage of the BNPD algorithm, a packing design is required to assure the convergence. When a packing design can be used, the combination of packing design and the algorithm BNPD provide good performance for detecting positive clones in a short period. But, in some cases, we may not be able to examine enough pools because of the cost of experimentation. In such cases we may not be able to utilize a packing design. If we cannot utilize a packing design, the CCPD algorithm is a good alternative to BNPD. Furthermore, when we need a more accurate result than that of CCPD, we can use the hybrid algorithm of CCPD and MCPD as was noted in Subsection 5.6.

**References**


