國 立 交 通 大 學

應用數學系

碩 士 論 文

最小覆蓋問題在測集設計的反物模型上的應用

The Minimum Covering Problem with Application to Pooling Designs with Inhibitors

研 究 生：陳 正 傑

指導老師：黃 光 明　教 授

中 華 民 國 九 十 三 年 六 月

最小覆蓋問題在測集設計的反物模型上的應用

The Minimum Covering Problem with Application to Pooling Designs with Inhibitors

研 究 生：陳正傑　　**Student**: Cheng-Jie Chen

指導老師：黃光明 教授　**Advisor**: Frank K. Hwang

國 立 交 通 大 學

應用數學系

碩 士 論 文

A Thesis

Submitted to Department of Applied Mathematics
College of Science
National Chiao Tung University
In partial Fulfillment of Requirement
For the Degree of Master
In
Applied Mathematics
June 2004
Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 三 年 六 月

最小覆蓋問題在測集設計的反物模型上的應用

研 究 生：陳正傑　　指導老師：黃光明　教授

國 立 交 通 大 學

應 用 數 學 系

摘　要

最小覆蓋問題在圖論中算是個大問題, 很多題目都可以轉換成這類形的問題. 在這篇文章中, 我們會討論到的是它的一個特別例子, 即發生在二分圖中的最小覆蓋問題, 我們將給出一個下界並將這個結果用在測集設計中.

一個 clone 是一小段 DNA 序列,clone library 是儲存了大量 clone 的地方,從 clone library 裡找出特定性質的 clones, 我們稱之為正物,便是 clone library 的檢測問題。將一群 clones 放在一起並稱此集合為一個測集(pool)。我們用檢測測集代替檢測所有 clone 。另外, 我們還希望同時檢測所有的測集, 以節省我們的時間。所謂測集設計就是同時檢測所有測集的方法,我們利用它有效的找出正物。

由於 DNA 分子間的作用,有時正物會與某些亦存在於 clone library 裡的 clones,我們稱之為反物.產生化學反應。如果受測試的測集中,同時包含了一些正物跟反物,我們會誤認此測集中並不含有正物.本論文利用了最小覆蓋問題的結果,提出了一個結果可以應用在測集設計上,達到檢測出所有正物的結果。

關鍵詞：最小覆蓋問題，測集設計，反物模型。

中 華 民 國 九 十 三 年 六 月

The Minimum Covering Problem with Application in Pooling Designs with Inhibitors

Student : Cheng-Jie Chen Advisor : F.K. Hwang

Department of Applied Mathematics
National Chiao Tung University
Hsinchu 300, Taiwan, R.O.C.

Abstract

Many problems in graph theory can be formulated as a minimum covering problem. In this thesis, we will discuss a special form of the minimum covering problem, based on the bipartite graph. We will give a lower bound of the cardinality of a vertex-cover and use this result in screening clone library.

A clone is a DNA subsequence and a clone library is a large collection of clones. Screening of a clone library is to identify all clones containing a specific subsequence, in the clone library. We refer to such a clone as a positive clone. We choose a set of clones to form a pool and screen each pool as a unit for the existence of a positive clone. Further, we screen all pools in parallel to save time. Such a set of pools is called a pooling design which helps us solve the clone library screening problem efficiently.

Due to the interaction between DNA molecules, positive clones might interact with some other clones, called inhibitors, to lose their positive effect. This thesis, using the result of the minimum covering problem, provides a result which identifies all positive clones even in the presence of inhibitors and errors for some inhibitor model.

Keywords: Minimum covering problem, Pooling designs, Inhibitor model.

致　　謝

首先，這篇論文的完成要特別感謝我的指導老師：黃光明教授，這兩年來一直有耐心的指導著我，也教我很多做研究的心態跟技巧，謝謝!

接著要感謝林琲琪學姐，張飛黃學長跟郭君逸學長，謝謝你們的提攜才能使我順利完成碩士學位。

當然還要謝謝我的同學，尤其是唐文祥跟陳建瑋，真的是幫了我很大的忙。還有要感謝家人的支持，讓我沒有後顧之憂的作研究。

最後，我還要感謝跟我一起相聚的朋友，沒有你們，也許我沒有辦法走過如此艱辛的一段路程，在此僅將此論文獻給所有支持和鼓勵我的人，謝謝你們!

Contents

List of Figures

List of Tables

1 Introduction

Many problems in graph theory can be formulated as an instance of the following set-covering problem (for ease of writing, we omit the word "set" from now on):

Problem (A minimum covering of R):

Let G be a bipartite graph with bipartition (D, R) where every vertex in R has degree at least s. We want to find a subset $X_0 \in D$ of minimum cardinality such that $N_G(X_0) = R$, where $N_G(G)$ is the set of neighbors of X_0 in G.

This problem is NP-hard (Karp(1972))[6]. In Chapter 2. We will use the greedy algorithm to find approximate solutions, which we will use in Chapter 3.

The emergence of pooling designs arose from the need to screen a collection of clones, say, from a clone library, against a *probe*. A clone represents a short DNA fragment cut from a molecule into storage size, and a probe is a specific DNA fragment (very short) that we are interested in. (For more information about clone library, see pp 83-85 in [7]). A clone is called positive if it contains the complement of the probe as a segment. Otherwise, it is called negative. Our goal is to identify all positive clones in the library efficiently. (For ease of writing, we use "positives" instead of "positive clones" and "negatives" instead of "negative clones" from now on).

Because the number of clones in a library is very large, screening each clone individually, of course, is not a good idea. An alternative strategy is to select a set of clones to form a pool and assay the pool as a unit. There are two possible outcomes: a pool with negative outcome is called a negative pool which signifies that no clone in it is positive; otherwise it is called a positive pool which signifies that there's at least one positive in this pool. This strategy is helpful because in most situations, relatively few clones are positive for the probe. Hence one negative pool identifies all clones contained in it as negative.

A pooling design is a collection of pools. It is a screening scheme which identifies all positives in the library efficiently. Figure 1 is an example for a pooling design, in which

C_2 is contained in pools 1 and 2; pool 1 contains C_1, C_2, C_3 and C_5. A pooling design with t pools for a library and n clones is represented by a $t \times n$ binary matrix M. Each column of M represents a clone and each row represents a pool. A 1-entry in cell (i,j) signifies that clone j is contained in pool i.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7
P_1	1	1	1		1		
P_2	1	1		1		1	
P_3	1		1	1			1

Figure 1: A t=3,n=7 pooling design.

After all pools have been assayed, let 1 represent a positive outcome and 0 represent a negative outcome. Then the t outcomes can be written as an outcome vector. While a column is a binary t-vector, it can also be viewed as a subset of 1,...,t depending on the locations of its 1-entries. So we can talk about a union of columns, which is simply their Boolean sum. Note that the union of all positives is the outcome vector. For example, in Figure 2, suppose C_2 and C_5 are the positives, then the outcome vector would be 1,1,0. We have to decode the outcome vector to infer what is the set of positives.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	**Outcome**
P_1	1	1	1		1			1
P_2	1	1		1		1		1
P_3	1		1	1			1	0

Figure 2: The outcome vector.

In the pooling design problem, all information we have consists of the matrix M and the outcome vector. Whether we can decode correctly, i.e, identifying the positive clones, from this outcome vector apparently depends on the structure of the matrix.

An example of a pooling design with good structure is the *d-disjunct* matrices. A matrix is *d*-disjunct if the union of any up to d columns can not *cover* any other column.

Figure 3 gives a 2-disjunct matrix. Note that a set of d columns can be viewed as a candidate set of positives. The union of this candidate set then yields the outcome vector. When there are at most d positives, the d-disjunct property guarantees that all negatives have at least one 1-entry not covered by the set of positive pools, or at least one 1-entry in a negative pool, hence it can be identified as a negative.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	outcome
P_1	1	1	1	1									1
P_2	1				1	1	1						1
P_3	1							1	1	1			0
P_4		1					1		1		1		1
P_5			1		1			1			1		1
P_6				1		1							0
P_7		1				1				1		1	1
P_8				1	1				1			1	1
P_9			1				1	1				1	0

Figure 3: A 2-disjunct matrix.

An error in pooling design changes the outcome vector of a pool($0- > 1$, $1- > 0$). For example, in Figure 4, suppose C_2 and C_5 are the positives. The outcome vector would be 1,0,0 if pool 2 incurs an error.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	Outcome
P_1	1	1	1		1			1
P_2	1	1		1		1		0
P_3	1		1	1			1	0

Figure 4: The outcome vector with error.

The inhibitor is a new category of clones whose presence in a pool dictates a negative outcome, regardless of the presence of positives in that pool. Farach et al. [4] first introduced this model. Let n denote the total number of clones including at most d positives and at most r inhibitors. They gave a randomized algorithm to identify all positives in O((d+r)logn) tests, assuming d+r$\ll$n. De Bonis and Vaccaro [2] gave a

deterministic algorithm in O($(r^2$+d)logn) tests. However, both algorithms are sequential in nature, namely, tests cannot be preformed in parallel. It is possible to convert the De Bonis and Vaccaro algorithm to a 3-stage algorithm (tests in a stage can be preformed in parallel) by increasing the number of tests to O($(r^2+d^2)log$n). Hwang and Liu[5] gave a pooling design which can tolerate e errors.

In a k-fold inhibitor model, the presence of an inhibitor in a pool dictates a negative outcome unless the number of positive clones in that pool is at least k. Our goal is still to identify all positive clones, figure 5 gives an example. Suppose C_1 , C_2 and C_6 are the positive clones, while C_7 is a 2-fold inhibitor. Then P_4 would have a negative outcome but P_2 would not, since P_2 intersects two positives clones, and P_4 intersects one.

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	outcome
P_1	1	1	1	1									1
P_2	1				1	1	1						1
P_3	1							1	1	1			1
P_4		1					1		1		1		0
P_5			1		1			1			1		0
P_6				1		1							1
P_7		1				1				1		1	1
P_8				1	1				1			1	0
P_9			1				1	1				1	0

Figure 5: A pooling design under the 2-fold inhibitor model.

We study the k-fold inhibitor model in Sec.3, using the results obtained in Sec.2.

2 Result of a minimum covering of R

In this section, we consider the minimum covering problem stated in Chapter 1. Consider a bipartite graph $G(D, R)$ with $|D| = m$, $|R| = n$, and every vertex in R has degree at least s. We assume the degrees are exactly s since that is clearly the worst case. In order to find X_0, we use the greedy covering algorithm which we give below.

Greedy covering algorithm

- **Begin** with bipartite graph G(D,R).
- **Let** C=$\emptyset$.
- **Repeat while** $R \neq \emptyset$

  Choose a vertex v of maximum degree in D.

  Set $D = D - v$, and $R = R - N_G(v)$.

  Add v to the set C.
- **End while**.

Let f=$\lceil sn/m \rceil$, and m_i be the number of vertices which covers i neighbors in the greedy algorithm. We have $\sum_{i=0}^{p} im_i = n$ immediately since C is a cover of R.

Lemma 1 *Suppose the first vertex chosen in C covers p neighbors, p≥f. Then*

$s(n - pm_p - (p-1)m_{p-1} \cdots - (p-i)m_{p-i}) \leq (p-i-1)(m - m_p - m_{p-1} \cdots - m_{p-i})$

for $0 \leq i \leq p-2$.

Proof. If not, then

$s(n - pm_p - (p-1)m_{p-1} \cdots - (p-i)m_{p-i}) > (p-i-1)(m - m_p - m_{p-1} \cdots - m_{p-i})$,

which says that after all vertices in D each covering at least p-i vertices in R have been chosen, there is still a vertex $x \in D$ of degree at least > p-i-1 which the greedy algorithm could choose to cover at least p-i vertices, contradicting our assumption of $m_p, m_{p-1}, \cdots, m_{p-i}$.

∎

We want to find an upper bound of $\sum_{i=1}^{p} m_i$ since $\sum_{i=1}^{p} m_i$ is the cardinality of a covering set C, and $\sum_{i=0}^{p} im_i = n$. Now, an upper bound of $\sum_{i=1}^{p} m_i$ can be considered as a lower bound of $\sum_{i=1}^{p}(i-1)m_i$ since their sum is $\sum_{i=0}^{p} im_i = n$, a constant.

Define $c(v)_{m,n,s} = \{\frac{v!(s-1)^v[(v+1)m-n]}{\prod_{w=1}^{v}[(w+1)s-w]}\} - m + n.$, when the context is clear, we use c(v) instead $c(v)_{m,n,s}$

Note: $c(1)_{m,n,s} = \frac{(s-1)[2m-n]}{2s-1} - m + n = \frac{(s-1)[2m-n]-2sm+m+2sn-n}{2s-1} = \frac{sn-m}{2s-1}$. If $c(1)_{m,n,s} < 0$, then $sn - m < 0 \Rightarrow f = 1$. In this case, clearly the worst case occurs when the degree

of each vertex of D is not more than 1. Then it takes m vertices in D to cover R. Thus we only need to consider $c(1)_{m,n,s} > 0$.

Lemma 2 *c(f-1) is a maximum of c(v) for 1 ≤ v ≤ p-1.*

Proof.

$$c(v+1) - c(v) = \frac{(v+1)!(s-1)^{v+1}[(v+2)m-n]}{\prod_{w=1}^{v+1}[(w+1)s-w]} - m + n - [\frac{v!(s-1)^{v}[(v+1)m-n]}{\prod_{w=1}^{v}[(w+1)s-w]} - m + n]$$

$$= \frac{v!(s-1)^{v}}{\prod_{w=1}^{v+1}[(w+1)s-w]}[(v+1)(s-1)[(v+2)m-n] - [(v+2)s-(v+1)][(v+1)m-n]]$$

$$= \frac{v!(s-1)^{v}}{\prod_{w=1}^{v+1}[(w+1)s-w]}[(v+1)(s-1)[(v+1)m-n] + (v+1)(s-1)m - [(v+2)s-(v+1)][(v+1)m-n]$$

$$= \frac{v!(s-1)^{v}}{\prod_{w=1}^{v+1}[(w+1)s-w]}[(vs+s-v-1-vs-2s+v+1)[(v+1)m-n] + (v+1)(s-1)m]$$

$$= \frac{v!(s-1)^{v}}{\prod_{w=1}^{v+1}[(w+1)s-w]}[(-s)[(v+1)m-n] + (v+1)(s-1)m]$$

$$= \frac{v!(s-1)^{v}}{\prod_{w=1}^{v+1}[(w+1)s-w]}[sn - s(v+1)m + (v+1)(s-1)m]$$

$$= \frac{v!(s-1)^{v}}{\prod_{w=1}^{v+1}[(w+1)s-w]}[sn - (v+1)m]$$

Since f=$\lceil sn/m \rceil \Longrightarrow sn > (f-1)m$ and $sn \leq fm \Longrightarrow$, $sn-(v+1)m > 0$ for $1 \leq v \leq f-2$ and $sn-(v+1)m \leq 0$ for $f-1 \leq v \leq p-1$. Then $c(v+1) \leq c(v)$ for f-1 ≤ v ≤ p-1, and hence c(f-1) is a maximum of c(v) for 1 ≤ v ≤ p-1. ∎

Theorem 3 *Consider a bipartite graph G(D,R) where $|D| = m$, $|R| = n$, and every vertex in R has degree at least s. We have*
$\sum_{i=1}^{p}(i-1)m_i \geq c(v) + \frac{v!(s-1)^{v}}{\prod_{w=1}^{v}[(w+1)s-w]}[[\sum_{j=v+2}^{p}[j-(v+1)]m_j]$ *for* $1 \leqq v \leqq p-1$.

Proof. By induction. Suppose v=1. Set i= p-2 in Lemma1, we have

$$s(n - pm_p - (p-1)m_{p-1} \cdots - 2m_2) \leq (m - m_p - m_{p-1} \cdots - m_2).$$

Then,

$$m_2 \geq \frac{sn - m - [\sum_{j=3}^{p}[sj-1]m_j]}{2s-1}.$$

We find

$$
\begin{aligned}
&\sum_{i=1}^{p}(i-1)m_i = m_2 + \sum_{i=3}^{p}(i-1)m_i \\
\geq\ & \frac{sn-m}{2s-1} + \frac{1}{2s-1}[\sum_{j=3}^{p}(2s-1)(j-1)m_j - \sum_{j=3}^{p}[sj-1]m_j] \\
\geq\ & \frac{sn-m}{2s-1} + \frac{1}{2s-1}[\sum_{j=3}^{p}[2sj-j-2s+1-sj+1]]m_j \\
\geq\ & \frac{sn-m}{2s-1} + \frac{s-1}{2s-1}[\sum_{j=3}^{p}(j-2)m_j] \\
=\ & c(1) + \frac{s-1}{2s-1}[\sum_{j=3}^{p}(j-2)m_j].
\end{aligned}
$$

So v=1 is true.

Suppose v=q ,$1 \leq q \leq p-3$, holds, i.e.,

$$
\begin{aligned}
&\sum_{i=1}^{p}(i-1)m_i \\
\geq\ & c(q) + \frac{q!(s-1)^q}{\prod_{w=1}^{q}[(w+1)s-w]}[\sum_{j=q+2}^{p}(j-(q+1))m_j].
\end{aligned}
$$

Now let v=q+1. Setting i=p-q-2 in Lemma1,

$$s(n-pm_p-(p-1)m_{p-1}\cdots-(q+2)m_{q+2}) \leq (q+1)(m-m_p-m_{p-1}\cdots-m_{q+2}).$$

We find

$$m_{q+2} \geq \frac{sn-(q+1)m-[\sum_{j=q+3}^{p}[sj-(q+1)m_j]]}{s(q+2)-(q+1)}.$$

Combining the two inequalities, we find

$$
\begin{aligned}
&\sum_{i=1}^{p}(i-1)m_i \\
\geq\ & c(q)+\frac{q!(s-1)^q}{\prod_{w=1}^{q}[(w+1)s-w]}[\sum_{j=q+3}^{p}(j-(q+1))m_j+m_{q+2}] \\
\geq\ & c(q)+\frac{q!(s-1)^q}{\prod_{w=1}^{q}[(w+1)s-w]}[[\sum_{j=q+3}^{p}(j-(q+1))m_j]+ \\
& \frac{(q!(s-1)^q)(sn-(q+1)m-[\sum_{j=q+3}^{p}[sj-(q+1)m_j]])}{\prod_{w=1}^{q+1}[(w+1)s-w]}] \\
=\ & c(q)+\frac{q!(s-1)^q}{\prod_{w=1}^{q+1}[(w+1)s-w]}[sn-(q+1)m]+ \\
& \frac{(q!(s-1)^q)}{\prod_{w=1}^{q+1}[(w+1)s-w]}[\sum_{j=q+3}^{p}[j-(q+1)][(q+2)s-(q+1)]m_j-\sum_{j=q+3}^{p}[sj-(q+1)]m_j] \\
=\ & c(q+1)+\frac{(q!(s-1)^q)}{\prod_{w=1}^{q+1}[(w+1)s-w]}[\sum_{j=q+3}^{p}[[(q+2)s-(q+1)][j-q-1]-sj+(q+1)]m_j] \\
=\ & c(q+1)+\frac{(q!(s-1)^q)}{\prod_{w=1}^{q+1}[(w+1)s-w]}[\sum_{j=q+3}^{p}(q+1)[sj-j-(q+2)s+(q+1)+(q+1)+1]m_j] \\
=\ & c(q+1)+\frac{(q!(s-1)^q)}{\prod_{w=1}^{q+1}[(w+1)s-w]}[\sum_{j=q+3}^{p}(q+1)(s-1)[j-(q+2)]m_j] \\
=\ & c(q+1)+\frac{(q+1)!(s-1)^{q+1}}{\prod_{w=1}^{q+1}[(w+1)s-w]}[\sum_{j=q+3}^{p}[j-(q+2)]m_j]
\end{aligned}
$$

So, by the induction hypothesis, theorem 2 is true for $1 \leq v \leq p-1$. ∎

Corollary 4 *Using the greedy algorithm,* $\sum_{i=1}^{p}(i-1)m_i \geq c(f-1)$.

Proof. By Theorem 3

$$
\begin{aligned}
&\sum_{i=1}^{p}(i-1)m_i \\
\geq\ & c(f-1)+\frac{(f-1)!(s-1)^{f-1}}{\prod_{w=1}^{f-1}[(w+1)s-w]}[\sum_{j=f+1}^{p}[j-f]m_j].
\end{aligned}
$$

Since $m_j \geq 0$ for $f+1 \leq j \leq p-1$, and $m_p \geq 1$, $\sum_{i=1}^{p}(i-1)m_i \geq c(f-1)$. ∎

By Corollary 4, we find a lower bound of $\sum_{i=1}^{p}(i-1)m_i$ to be $c(f-1)$, and an upper bound of $|X_0|$ is $n-c(f-1)$.

Now we give a relation of $n-c(f-1)$ and m.

Lemma 5 *n-c(f-1) is increasing in m.*

Proof. Since $n-c(f-1)_{m,n,s} = m - \{\frac{(f-1)!(s-1)^{f-1}[fm-n]}{\prod_{w=1}^{f-1}[(w+1)s-w]}\}$.

Now we consider $n-c(f-1)_{m-1,n,s}$, if f does not change, then

$n-c(f-1)_{m-1,n,s} = (m-1) - \{\frac{(f-1)!(s-1)^{f-1}[f(m-1)-n]}{\prod_{w=1}^{f-1}[(w+1)s-w]}\}$

and,

$n-c(f-1)_{m,n,s} - n + c(f-1)_{m-1,n,s} = 1 - \frac{(f-1)!(s-1)^{f-1}[fm-n]}{\prod_{w=1}^{f-1}[(w+1)s-w]} + \frac{(f-1)!(s-1)^{f-1}[f(m-1)-n]}{\prod_{w=1}^{f-1}[(w+1)s-w]} =$
$1 - \frac{(f-1)!(s-1)^{f-1}[f]}{\prod_{w=1}^{f-1}[(w+1)s-w]} = 1 - \prod_{w=1}^{f-1}\frac{[(w+1)s-(w+1)]}{[(w+1)s-w]} > 0$

else f increase 1 (note: f=$\lceil sn/m \rceil$), at this time, we give

$c(f)_{m-1,n,s} - c(f-1)_{m-1,n,s} = \frac{f!(s-1)^{f}[(f+1)(m-1)-n]}{\prod_{w=1}^{f}[(w+1)s-w]} - \frac{(f-1)!(s-1)^{f-1}[f(m-1)-n]}{\prod_{w=1}^{f-1}[(w+1)s-w]}$
$= \frac{(f-1)!(s-1)^{f-1}}{\prod_{w=1}^{f-1}[(w+1)s-w]}[\frac{fs-f}{(f+1)s-f}[(f+1)(m-1)-n] - f(m-1)+n]$
$= \frac{(f-1)!(s-1)^{f-1}}{\prod_{w=1}^{f-1}[(w+1)s-w]}[\frac{1}{(f+1)s-f}[(fs-f)(f+1)(m-1) + sn - [(f+1)s-f]f(m-1)]]$
$= \frac{(f-1)!(s-1)^{f-1}}{\prod_{w=1}^{f-1}[(w+1)s-w]}[\frac{1}{(f+1)s-f}[f(m-1)[(s-1)(f+1) - [(f+1)s-f]] + sn]]$
$= \frac{(f-1)!(s-1)^{f-1}}{\prod_{w=1}^{f-1}[(w+1)s-w]}[\frac{1}{(f+1)s-f}[f(m-1)+sn] > 0]$

then $c(f)_{m-1,n,s} > c(f-1)_{m-1,n,s}$. Hence $n-c(f-1)_{m,n,s} - n + c(f)_{m-1,n,s} > n - c(f-1)_{m,n,s} - n + c(f-1)_{m-1,n,s} > 0$, and we find $n-c(f-1)$ is increasing in m. ∎

Sapozhenko[1] also gave an upper bound $1+\frac{m}{s}(1+log\frac{sn}{m})$ by using the greedy algorithm. We will show our upper bound is less than $1+\frac{m}{s}(1+log\frac{sn}{m})$

His analysis can be summarized to: let U_k be the subset of vertices remaining in R after the kth iteration of the greedy algorithm. Then there is a vertex $x \in D$ of degree at least $\frac{s|U_k|}{m-k}$.

Let X be the first q chosen vertices in greedy algorithm, and let $\bar{n_q}$ be $|N_G(X)|$ in our analysis, and let $\bar{n_q'}$ be $|N_G(X)|$ in his analysis. Sapozhenko gave $\bar{n_q'} = \sum_{i=1}^{q}\frac{s(n-\sum_{v=1}^{i-1}n_v')}{m+1-i}$

Lemma 6 $\bar{n_q} \geq \bar{n_q'}$ *for all q.*

Proof. By induction. Suppose q=1. Obviously,

$\bar{n_1} = \lceil \frac{sn}{(m)} \rceil \geq \frac{sn}{(m)} = \bar{n_1'}$

Suppose q=p holds.

Condider q=p+1. $\bar{n_p} \geq \bar{n_p'}$, so $\bar{n_{p+1}} = \bar{n_p} + \lceil \frac{sn-\bar{n_p}}{m-p} \rceil$

$\geq \bar{n_p} + \frac{sn-\bar{n_p}}{m-p} = \bar{n_p}(1 + \frac{1}{m-p}) + \frac{sn}{m-p}$

$\geq \bar{n_p'}(1 + \frac{1}{m-p}) + \frac{sn}{m-p} = \bar{n_p'} + \frac{sn-\bar{n_p'}}{m-p} = \bar{n_{p+1}'}$ ∎

Thus our analysis produces better results than his.

We give some numerical comparisons of $|X_0|$, as a function of d, between our method(Table 1) and Sapozhenko's method(Table 2).

n \ s	2	3	4	5	6	7	8	9	10
0.5d	0.5	0.4	0.357143	0.316239	0.289773	0.265425	0.247335	0.230698	0.21743
0.75d	0.5833	0.485714	0.421429	0.372549	0.336601	0.308334	0.28526	0.265495	0.24884
d	0.6667	0.542857	0.465934	0.411621	0.370589	0.338207	0.311844	0.289872	0.27122
2d	0.8	0.659008	0.565337	0.498003	0.446871	0.406492	0.373658	0.346349	0.32321
3d	0.8571	0.716227	0.616951	0.544122	0.488307	0.439389	0.403695	0.37397	0.34877

Table 1: $|X_0|/d$ in our method

n \ s	2	3	4	5	6	7	8	9	10
0.5d	0.5	0.468488	0.423287	0.383258	0.349769	0.321823	0.298287	0.278231	0.26095
0.75d	0.702733	0.603643	0.524653	0.464351	0.417346	0.379747	0.34897	0.323283	0.30149
d	0.846574	0.699537	0.596574	0.521888	0.465293	0.420844	0.38493	0.355247	0.33026
2d	1.193147	0.930586	0.76986	0.660517	0.580818	0.519865	0.471574	0.432264	0.39957
3d	1.39588	1.065742	0.871227	0.74161	0.648395	0.577789	0.522257	0.477315	0.44012

Table 2: $|X_0|/d$ in Sapozhenko's method

Our result can be extended to non-bipartite graphs.

A *vertex dominating set* in a graph G is defined to be a subset of vertices $D \subseteq V(G)$ such that each vertex $x \in V(G) \setminus D$ is adjacent to some vertex of D. The problem is to

find a vertex dominating set D_0 consisting of as few vertices as possible. We give a lemma of this problem.

Lemma 7 Every n-vertex graph with minimum degree k has a dominating set of size at most $n - c(k)_{n,n,k+1}$.

Proof. By interpreting G as a bipartite graph G'(D,R) where D=R=V(G). Each vertex $x \in D$ covers itself in R and those vertices in R, which it is adjacent to in G. Hence s=k+1. ∎

3 Pooling design with k-fold inhibitor model

If k>d, then all tests will have a negative outcome. To avoid this uninteresting case, we assume k≤d in this section.

Definition: A column C_i in a 0-1 matrix is *isolated* if there exists one row that contains C_i only.

To avoid trivial discussion, it is often assumed that a pooling designs has no isolated column. We make this assumption in this section. D'yachkov and Rykov[3] proved that:

Lemma 8 *Every column should have column weight at least* $m+1$ *in a m-disjunct matrix.*

Lemma 9 *Let* K *denote a set of* k *row indices. There exists a set of at most* k *columns whose union covers* K*.*

Proof. For each row index in K, select any column with a 1-entry in that row (whose existence is guaranteed by the assumption of no isolated column). Then the set of at most k chosen columns covers K. ∎

Suppose there are at most d positives , r inhibitors and e errors.

Lemma 10 *A positive should have at least $d+e+1$ 1-outcomes in a $(d+r+2e)$-disjunct matrix.*

Proof. Suppose there is a positive C which has at most $d+e$ 1-outcomes. By Lemma 9, there exists a set of at most $d+e$ columns covering the row indices of these 1-outcomes. Further all its other (at least $r+e+1$) 1-entries are covered by the r inhibitors and the up to e errors. Since there is no isolated column, there exists a set E of e column, $C \nsubseteqq E$, covering the row indices of the up to e errors. Hence at most $(d+e)+(r+e) = d+r+2e$ columns cover C, contradicting Lemma 8. ∎

Lemma 11 *A negative should have at least $r+e+1$ 0-outcomes in a $(d+r+2e)$-disjunct matrix.*

Proof. Suppose to the contrary that a negative C has only $r+e$ 0-outcomes. Let E be defined as in the last lemma. Then all the 1-entries of C with 1-outcomes are covered by at most $d+e$ columns, and all with 0-outcomes by at most $r+e$ columns, leading to the conclusions that C is covered by $d+r+2e$ columns, a contradiction to Lemma 8. ∎

Let M denote a $(d+r+2e)$-disjunct matrix . Let I_i denote an inhibitor with weight w_i, and let W_i denote the set of pools i appears and $W_i^+ \subseteq W_i$ the subset of pools with positive outcomes. We will show that W_i can be covered by at most d+r+2e columns, there violating the $(d+r+2e)$-disjunctness of M (lemma 9). This is accomplished by observing that W_i^+ can be covered by a fraction of the positive clones, using the results in the last section. We first need to turn the problem into the covering problem format.

Define the graph G(D,R) by taking D so the set of d positives (by Lemma 5, d is the worst case among all d'¡=d). R as the set W_i^+ and an edge from u∈D to v∈R if u is a positive clone appearing in pool v. Then each v most have degree at least k since it takes at least k positive clones to yield a positive outcome. Figure 6 illustrates the construction of G(D,R) from M although only the columns corresponding to I_i and the d(6) positive clones in M are shown.

	I	D_1	D_2	D_3	D_4	D_5	D_6	Outcome
P_1	1	1	1	1				1
P_2	1				1			1
P_3	1					1	1	1
P_4	1		1			1		1
P_5	1	1		1				1
P_6	1		1			1	1	1
P_7	1				1			1
P_8	1							0
P_9	1							0

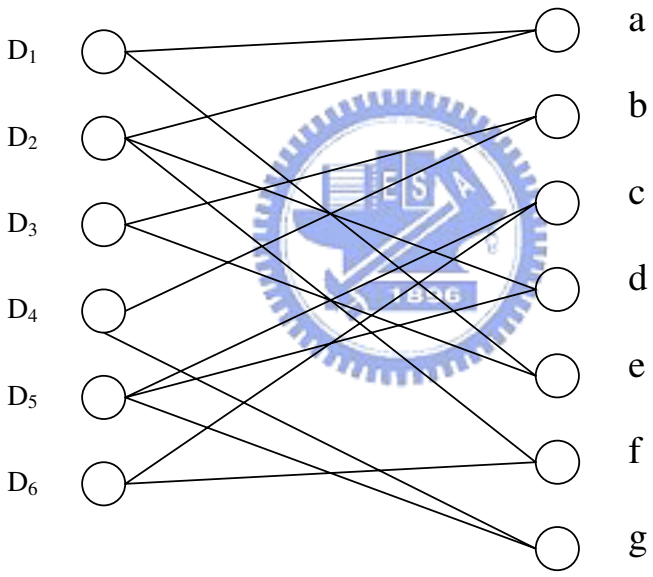
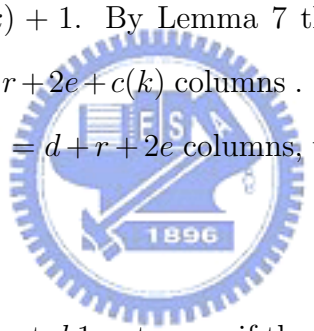


Figure 6: In a 2-fold inhibitor model, relation between positives and the 1-outcomes.

Now the main results.

Theorem 12 *A k-fold inhibitor with weight less than* $d+r+c(k)_{d,d+1,k}+2e+1$ *have at most* $d+e$ *1-outcomes in a d+r+2e-disjunct matrix.*

Proof. Since this matrix has at most e errors, it suffices to show that not counting errors, if an inhibitor I has at least $d+1$ 1-outcomes, then it will violate the (d+r+2e)-disjunctness. Let R denote a set of d+1 1-outcomes of I. Then each of these $d+1$ 1-outcomes has to be covered by k positive clones'. Hence we can define a bipartite graph G(R,D) with $|R| = d+1$. $|D| = d$, $s = k$, then f=$\lceil k(d+1)/d \rceil$=k+1 (recall $k \leq d$ obviously). By Corollary **??** it only needs $d-c(k)$ vertices in D to cover R, in other words it only needs $d-c(k)$ positive clones to cover the 1-outcome of I. Suppose weight of I is less than $d+r+2e+c(k)+1$. By Lemma 7 the other at most $r+2e+c(k)$ 0-outcomes of I can be covered by $r+2e+c(k)$ columns . Combining, I can be covered by $d-c(k)_{d,d+1,k}+r+2e+c(k)_{d,d+1,k} = d+r+2e$ columns, violating (d+r+2e)-disjunctness assumption.



So an inhibitor must have at most d 1-outcomes if there is no error. Considering error, an inhibitor must have at most $d+e$ 1-outcomes. ∎

Using the above result, we give a 1-stage method.

Pooling: Use a (d+r+2e)-disjunct matrix with column weight $\leq d+r+2e+1+c(k)_{d,d+1}$.

Decoding:

Step1. Partition clones into 3-sets: P consists of those with at most $r+e$ 0-outcomes. O consists of those with at most $d+e$ 1-outcomes, R consists of the rest.

Step2. If $R \neq \phi$ let the outcome vector be V and denote the union of an r-subset S of O as V'_S. Let $U_S = V \cup V'_S$. If clone C has at most e 0-outcomes under U_S, then put it into P. Do this for all S.

Step3. Output P as our positives.

Lemma 13 *After step1, P is contained in the set of all positives. O contains all inhibitors and no positive.*

Proof. By Lemma 11, a negative has at least $r+e+1$ 0-outcomes and can't be in P. Further, the column weight is at least $(d+r+2e+1)$ in a (d+r+2e)-disjunct matrix, so a clone in P has at least $(d+e+1) > d+e$ 1-outcomes. By Theorem 12, inhibitors will not appear in P, either. By Lemma 9, a clone with at most $d+e$ 1-outcomes can't be a positive. An inhibitor can't have more than $d+e$ 1-outcomes. Hence O contains all inhibitors but no positive. ∎

Lemma 14 *A clone C in R is positive $\Longleftrightarrow$ there exists at least one r-subset of O such that C has at most e 0-outcomes under U.*

Proof. ($\Rightarrow$) O contains all inhibitors(whose number is at most r). Some r-subsets chosen in step 2 should contain all r inhibitors. Then the vector V'_S corrects the false negative outcomes caused by the inhibitors.

($\Leftarrow$) Suppose a clone C is negative. U can be viewed as the union of $d+r$ clones. Then $|C \backslash U| > e$ for otherwise C can be covered by $U \cup E$, where E is a set of e columns, a contradiction to the (d+r+2e)-disjunctness since $|U \cup E| \leq d+r+2e$. ∎

Corollary 15 *After Step 2, the set P contains all positives and nothing else.*

References

[1] A.D.Bonis and U.Varraro. Improved algorithms for group testing with inhibitors, Information Processing Letters 67(1998), 57-64

[2] A.G.D'yachkov and V.V.Rykov, *Bounds of the length of disjunct codes*, Problem-scontrol Infoem.Thy.11(1982), 7-13.

[3] M. Farach, S. Kannan, E. knill, and S. Muthukrishnan, Group testing with sceqquences in experimental molecular biology. Proc. Compression and omplexity of sequences(B.Carpentieri,A.Desantis, J.Storer, and U.Vaccaro, Eds.),IEEE computer soc, 1997, pp.357-367

[4] F.K.Hwang and Y.C.Liu, *Error-tolerant pooling designs with inhibitors*, J. Comput. Biology 10(2003),231-236.

[5] R.M. Karp, Reducibility among combinatorial problems. in complexity of Computer computations, Eds. R.E. Miller and J.W. Tratcher, Plenum Press, New York, 1972, pp.85-103

[6] A.A. Sapozhenko. *On the complexity of disjunctive normal forms,obtained with the help of the gradient algorithm,Diskret.Analiz.21(1972), 62-71 (in Russian)*

[7] M. S.Waterman, *Introduction to computational biology; Maps,sequences and genomes*, Chapman and Hall, 1995