# Pooling designs for high-throughput biological experiments

by

**Raghunandan M. Kainkaryam**

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Chemical Engineering and Scientific Computing)
in The University of Michigan
2010

For Peter Woolf

*il miglior fabbro*

# Acknowledgments

It would be impossible for me to overstate the influence that my advisor, Peter Woolf, has had on my development as a researcher and as an individual. His attitude towards life and work is something I strive to emulate, so far only with partial success, and consider a worthwhile goal for the rest of my life. Instead of struggling to express in words my gratitude to Peter, I dedicated my dissertation to him using words borrowed from T. S. Eliot's dedication of *The Waste Land* to Ezra Pound, which roughly translate as, "the better craftsman."

I benefited from having a dissertation committee that actively shaped the direction of my work both as advisors and as collaborators. While I engaged in flights of mathematical fancy, Professors Linderman and Ziff ensured that I stayed true to the engineering aspect of my work and kept me grounded in the biological details of the problem.

Professor Neubig, apart from representing my target audience – practitioners of high-throughput biological experiments, was responsible for steering me towards the problem of quantitative analysis of pooling design strategies (Chapter 4). Along with his student Levi Blazer, he helped infuse a dose of reality into my dissertation work by providing an opportunity to experimentally test the pooling design in a high-throughput drug screen. These experiments were made possible thanks to the enthusiastic participation of Steve vander Roest and Martha Larsen of the Center for Chemical Genomics.

The application of pooling designs to improve microarray experiments (Chapter 5) was a direct consequence of my collaboration with Angela Bruex and John Schiefelbein of the Molecular, Cellular, and Developmental Biology department on a separate project. Apart from gaining valuable (read employable) experience working on bioinformatics-based approaches to understanding root-hair development in *Arabidopsis thaliana*, the collaboration alerted me to a new system in which to test pooling designs. Angela and John showed immense patience and faith as I explained that I wanted to experimentally test my half-baked and seemingly crazy idea of using pooling designs to perform microarray experiments.

While I attempted to impersonate a mathematician by trying to come up with proofs and lemmas to characterize the properties of pooling designs, it was Professor Gilbert who encouraged my efforts but tempered them by teaching me some real mathematics. Her collaboration, along with her student Paul Shearer, was essential in identifying and establishing a connection between the new field of compressive sensing and the pooling designs I was using for high-throughput biological experiments (Chapters 4 & 5).

I owe an intellectual debt to Nicolas Thierry-Mieg of CNRS (France), who came up with the wonderful Shifted Transversal Design pooling strategy and had the foresight to abbreviate it STD (see Chapter 1 for the connection between syphilis testing and the origin of pooling designs). The STD design came at an opportune time for me as I was heading down a dead-end with my own attempts to create an efficient and scalable pooling design strategy. Much of my research work stands on his giant shoulders.

My expository skills were honed at the weekly meetings of the Systems Biology group. Any clarity in my explanation of pooling designs owes much to the patient and detailed feedback of my group mates.

My graduate school experience was made enjoyable thanks to a great group of friends and the subtle delights of the city of Ann Arbor. Of course, being able to participate in (trivia) quizzes and play cricket regularly made it all the more fun.

I would like to thank my parents, Narasimhan and Radha, and sister, Hema, for their love and understanding. I am grateful for their support while I chose to take the seemingly endless and monetarily unattractive route of doctoral studies after college. They instilled a love of learning and open-minded inquiry that has stood me in good stead through this journey.

Finally, my most important finding during graduate school has been my (now) wife, Mekhala. It is a finding that I hope to spend the rest of my life trying to fully understand.

# Table of Contents

# List of Tables

# List of Figures

# List of Appendices

# Chapter 1
# Introduction

In this thesis I explore the application of pooling design strategies to improve the performance of high-throughput biological experiments. This work provides a strong theoretical framework for applying pooling designs to high-throughput biological experiments, grounds these designs in the realities of actual experimental platforms, and experimentally validates this approach in two specific biological applications. The thesis also brings together tools and problems from very diverse fields and builds a general framework to address these problems going forward, especially in the context of biological experiments.

## 1.1 High-throughput biological experiments

Consider these two examples:

1. Pharmaceutical industries test millions of chemical compounds each day in order to identify potential drugs for a multitude of diseases. The process of finding cures for diseases that once depended on serendipity is now automated.

2. Gene expression microarrays enable us to measure the level of all genes (more than 20,000 in humans) in a cell sample. This allows researchers to identify and link genes to very specific biological conditions like, say, disease states.

Such large-scale, high-throughput biological experiments are the engines that drive the biotechnology revolution. New applications appear almost on a daily basis, accelerating the pace of biological discovery by removing human hands from the process. These applications make a variety of experimental tasks – searching for active chemical compounds, investigating interactions between various proteins, identifying genes that behave abnormally under different cell conditions, finding disease-causing genetic mutations that people may carry, etc. – easier than ever before.

Although high-throughput experimental systems allow us to do a lot of work very quickly, they have a few drawbacks as well. Inevitably these experiment platforms are expensive. Further, the high-speed sometimes comes at the cost of accuracy. These experiments are also prone to experimental error. This thesis considers these two problems (or challenges) of high throughput biological experiments: efficiency (reduce cost) and robustness (improve accuracy).

Solutions to these two problems are studied for a variety of high-throughput biological experiments that fit this simplified framework:

1. A large number of items are tested systematically in a biological test.
2. A small number of items that are of interest are identified.
3. The testing procedure, when scaled up to test a very large number of items on a regular basis, is expensive and prone to errors.

The current approach taken by high-throughput biological experiments is to test each item individually in its own test. That means a million tests are needed to identify a few hundred, or less, items of interest. The problem is made manageable by automating and optimizing the testing procedures, usually at high cost. This approach is wasteful and still does not address the problem of testing errors. A solution to this problem would be invaluable to the progress of biotechnology in the 21st century.

## 1.2 Pooling designs

The central idea behind this thesis was posed as a puzzle by my advisor in one of our early meetings.

"A group of scientists want to host a party. In their cellar are six bottles of wine. Over time, however, the labels on the bottles have eroded and the six bottles of wine got mixed up with one identically shaped bottle of rat poison. To identify the one bottle of rat poison and rescue the six bottles of wine, the scientists decide to test the contents of the bottles on their pet rats. However, it would take a rat as much time to die of the poison as they have left before the party. Furthermore, they have only three pet rats at their disposal. How do the scientists identify the poison bottle?"

After some thought, I realized that the key to the puzzle was in using the three rats, called A, B, and C, to somehow represent each of the seven bottles in a unique way. By trial-and-error I figured out that using combinations of rats – A, B, and C individually, then the pairs, A-B, A-C, B-C, and finally A-B-C together would help achieve this goal. These combinations give us a way to uniquely identify the seven bottles using just three rats. The solution to the puzzle was then to feed each rat a mixture of samples from the bottles. As illustrated in Figure 1, bottles one, two, and three were present only in mixtures fed to rats A, B, and C, respectively. Bottles four, five, and six were fed to pairs of rats, A-B, A-C, and B-C, respectively. Finally, a sample of bottle seven was fed to all three rats. By the time of the party, the pattern of death among the three rats would point exactly to the poison bottle. For example, if bottle five had poison, only rats A and C would die and thereby help pinpoint the culprit.

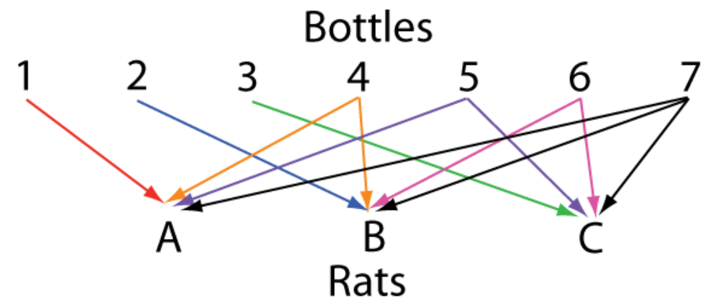

**Figure 1.1** Solution to pooling puzzle

This trick of using mixtures of samples to reduce the number of tests needed to get the necessary information is called pooling. The specification of which combinations of items are to be mixed is called the pooling design. Figure 1.2 shows the pooling design for the puzzle solution shown in Figure 1.1. The combination of items into mixtures or pools can be specified using a binary matrix. The rows represent the pools being tested and the columns the items being pooled in each test. A 1 in the matrix represents the presence of an item (along that column) in a pool (along that row), while a 0 represents its absence.

$$\begin{array}{c|ccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
A & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
B & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
C & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\hline
\end{array}$$

**Figure 1.2** Pooling design solution to pooling puzzle

In Figure 1.2, we can see that the pooling designs ability to use three rats to uniquely represent seven bottles is derived from the use of Boolean arithmetic. Each test with a yes-no outcome, the survival or death of a rat, represents a digit of a binary number. Three such digits can represent $2^3 = 8$ items uniquely. This allows us to represent seven bottles, or items, uniquely with just three rats, or tests. In general, when there is exactly one item of interest, say a poison bottle, out of n items to be tested, this can be achieved using just $\log_2 n$ tests. Therefore, a million bottles can be tested to identify the one poison bottle using just 20 rats! Such is the potential power of pooling designs.

It is this power that I want to exploit to solve the problem facing high-throughput biological experiments. The presence of only a small number of items of interest in a large number being tested, like one poison bottle among seven, allows for savings in the number of tests required to identify them. In a biological context, these items could be biological samples, chemical compounds, genes, proteins, or any collection of biological entities that constitute a set of similar items to be tested in an identical biological test for a response. A small subset of these items could be interesting due to a desired biological response in such a test. Next, the underlying biological test is repeated for every item being tested and therefore can be automated, allowing for high-throughput. Finally, the presence of errors in testing is a significant problem for high-throughput biological experiments. Such errors provide the strongest impetus to pursue pooling design approaches. When a large number of items are tested in high-throughput, it is usually unfeasible to retest these items for confirmation. Therefore, if an item of interest was wrongly identified in a test, it is essentially lost to us. Such a test is called a false negative. The opposite, called a false positive, would wrongly identify an item as interesting when in reality it is not. False positive tests can lead to erroneous conclusions or to an additional burden placed on retesting for confirmation interesting items that come out of a first round of tests. The reduction in tests provided by pooling designs can then be used to increase the number of times we test each item while still using fewer resources that present high-throughput approaches. It is the power of redundancy provided by pooling designs that allow it to become a potential solution to the problem facing high-throughput biological experiments.

However, there is cause to temper our enthusiasm. When there are multiple items of interest, the method of using $\log_2(n)$ digits to represent n items uniquely breaks down. This breakdown can be seen in the example shown in Figure 1.2. When we allow for two bottles of poison, say bottles one and five, still only rats A and C would die; leaving us with the erroneous conclusion that bottle five alone had poison in it. When multiple items of interest are to be identified, more advanced pooling designs are required to reduce the number of tests needed. Further, if these tests are subject to error, say a rat that was fed poison somehow survived thereby skewing the conclusions, then the problem gets even more complicated. Therefore, a solid theoretical framework is needed to address the problem of minimizing the number of tests needed to identify a small number of items of interest when a large number of items are subjected to identical testing methods, in the presence of testing error. Chapter 2 will describe the underlying theory of pooling designs that provides the framework required to solve this problem.

Pooling designs provide the means to test each item multiple times, always in pools with other items, across a collection of pooled tests, such that the total number of tests used is still lower than when each item is tested individually. The resulting redundancy, of testing each item multiple times, can allow us to overcome testing error. This feature can be easily demonstrated using the pooling puzzle.

Let's add an additional layer of complexity to the pooling puzzle by stating that we expect exactly one rat fed the poison to not die of it, thereby confounding our results. With this additional constraint, the simplistic method of testing each bottle on one rat individually would fail, as would the pooling design specified in Figure 1.2. A simple solution would be to perform the pooling design in Figure 1.2 twice, as shown in Figure 1.3. This trick would use 6 rats to test 7 bottles, still below the simplistic method of testing one bottle individually on a single rat (needs 7 rats) and well below the case where the whole experiment is repeated multiple times to overcome the error (needs 2×7=14 or 3×7=21 rats), while ensuring that the poison bottle is identified despite the single testing error.

$$
\begin{array}{c|ccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
A & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
B & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
C & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\hline
D & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
E & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
F & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\hline
\end{array}
$$

**Figure 1.3** An error-correcting version of the pooling puzzle. The dotted line bifurcates the replicate sets of pools.

This example gives a sense of the power of pooling designs to make high-throughput biological experiments both efficient and robust, even though these two goals are conflicting in nature.

To these two broad goals – efficiency and robustness – we can now add several layers of complexity that arise from the application of pooling designs to actual biological experiments. These complexities are listed below as a series of questions, still using the terminology of the pooling puzzle, to give a sense of the expanding scope of the problem.

1. *More than one item of interest* – The solution to the pooling puzzle is relatively simple

when there is exactly one item of interest among the many being tested – look for a unique representation for each item using the binary results of the tests. What if there are more than one items of interest? Unique representation does not work anymore and the exploding possibilities of multiple combinations of special items make it hard to visualize a similar solution.

2. *Varying strengths* – What if the items of interest occur in varying strengths, such that their response to the test is not simple a yes or no answer but a quantitative value – say, varying levels of poisoning of a rat?

3. *Dosage effect* – When feeding rats with mixtures of samples from bottles, should we worry about a dilution effect? As the number of bottles being tested increases the number of samples being pooled together also increases, while the consuming capacity of a rat remains fixed. This implies that smaller samples from each bottle need to be mixed together and fed to a rat. Would the poison still work in a lower dose? In response, if we had to limit the number of samples being pooled and fed to a single rat, how would the pooling design change?

4. *Errors* – What if one of the rats had a special immunity to the poison? Its unexpected survival would surely alter the results of the experiment. Can this be overcome?

5. *Interactions* – One of the wine samples pooled with the poison could act as an antidote and help the rat survive. This would confound the results of the experiment. Can this be identified and accounted for? Or the opposite case, where two wine samples that are harmless by themselves, turn into a deadly cocktail when mixed together.

6. *Additive effect* – If there are multiple bottles of poison among those being tested, but of varying strengths, there is a possibility that two bottles of poison that individually would not kill a rat, when pooled together become lethal. Can such an additive effect be detected?

Although extending the rat-poison analogy is sometimes limiting, the above set of problems capture the essence of some of the practical challenges of using pooling designs in biological settings. These problems can be seen as various cases of a general problem of error in testing and be dealt with accordingly, usually by increasing the number of pooled tests performed. However, they also present an opportunity to identify combinations of items that produce an effect of interest. Such unexpected benefits make the investigation of pooling designs for high-throughput biological experiments more interesting.

There are several real-life biological-testing problems that fit the three specifications for the application of pooling designs listed earlier. These are briefly described below, along with some historical context –adapted from the excellent introduction provided in the book by Du and Hwang (2000).

## 1.3 A brief history of pooling designs

Apart from the various versions of the pooling puzzle, the first known scientific treatment of the pooling approach was carried out during World War II, when U. S. army draftees were required to undergo a blood test for syphilis (Du and Hwang, 2000). Millions of blood samples were tested for the syphilitic antigen, resulting in the identification of a few thousand positive cases or less. Statisticians at the time hit upon the idea of pooling samples from multiple cadets and subjecting the pool to the laboratory test so as to reduce the number of tests to be performed. The central idea was that a negative test on a pool immediately eliminated all those cadets whose samples were in that pool, assuming no testing error, thus saving on several extra tests. The samples in a pool that produced a positive test would have to be retested individually in order to identify the offending sample or samples in that pool. The goal was to figure out the ideal pool size, or number of samples to be pooled together, so that the maximum savings in total number of tests could be achieved. Dorfman first proposed a solution to this problem (Dorfman, 1943). Although the solution was never used in practice, it became part of probability textbooks.

The pooling approach was generalized for a variety of applications and termed "group testing" by two Bell Laboratories scientists, Sobel and Groll (1959). Eventually, the field split into probabilistic and combinatorial group testing, based on the approach used to define the problem, and further into adaptive and nonadaptive group testing, based on the nature of the testing regime. These distinctions are discussed in greater detail in Chapter 2 of the thesis.

Independently, in a pattern that would repeat often for group testing problems, two computer scientists developed a mathematical construction called superimposed codes that allowed efficient retrieval of files in computer systems (Kautz and Singleton, 1964). The structure and properties of superimposed codes matched those of the pooling designs used for various testing problems. The connections between the two fields of group testing and coding theory were eventually identified and directed towards solving a range of related problems, which included – blood testing, chemical leakage testing, electric shortage testing, coding, multi-access channel communication, and many more (Du and Hwang, 2006).

Because this thesis focuses on biological testing problems, I describe below a variety of high-throughput biological applications that benefit from pooling designs.

1. Blood testing – From the early attempts of syphilis testing (Dorfman, 1943; Sobel and Groll, 1959; Du and Hwang, 2000) to more recent attempts at HIV testing (Hughes-Oliver, 2006; Kim et al., 2007; Westreich et al., 2008a) there is often the need to test a

large number of blood samples typically finding a small number of infections. The tests can be expensive and error-prone, hence the need for efficient and robust pooling strategies.

2. Clone library screening – Much of the human genome project required the identification of clones (long pieces of DNA) that contained a shorter, particular DNA sequence (Bruno et al., 1995; Knill et al., 1996; Balding, 1997). This required testing a library of clones against several such sequences repeatedly. Typically, a small number of clones would test positive for a given sequence.

3. Protein-protein interaction mapping – Large-scale efforts to identify interactions between proteins are a major part of proteomics. Pooling designs have been used to reduce the effort required to obtain such interaction maps (Jin et al., 2006; Thierry-Mieg, 2006b; Jin et al., 2007; Vermeirssen et al., 2007; Xin et al., 2009).

4. High-throughput drug screening – The first step in drug discovery is the identification of compounds that actively affect a biological target. However, there are millions of chemical compounds that need to be tested and the tests are often error-prone. In this thesis, I explore the use of pooling designs to reduce screening cost while ensuring that testing errors do not cause the loss of active compounds due to false negative test results (Wilson-Lingardo et al., 1996; Devlin et al., 1996; Chung, 1998; Snider, 1998; Ferrand et al., 2005; Jones and Zhigljavsky, 2001; Hughes-Oliver, 2006; Feng and Shoichet, 2006; Warrior et al., 2007; Kainkaryam and Woolf, 2008; Motlekar et al., 2008; Kainkaryam and Woolf, 2009).

5. Gene expression microarrays – Microarray technology enables the measurement of expression profile of all genes in a biological sample. However, they are an expensive and often noisy technology. In this thesis I show that pooling mRNA from samples before hybridization onto the microarry chip can provide savings in cost and means to overcome measurement noise (Kainkaryam et al., 2010).

6. Population genotyping – Next-generation sequencing technology makes massively parallel DNA sequencing possible (Mardis, 2008b,a). However, they are still expensive when used to sequence populations for rare genetic diseases. The nature of this problem allows for pooling designs to drastically reduce the cost of such endeavors (Erlich et al., 2009a; Prabhu and Pe'er, 2009; Patterson and Gabriel, 2009; Erlich et al., 2009b, 2010).

In this thesis, I develop and use state-of-the-art pooling design and decoding approaches for two specific applications, namely, high-throughput drug screening (Chapters 3 & 4) and gene expression microarrays (Chapter 5).

## 1.4 Thesis overview

Many of the applications listed above and related advances occurred in parallel and often oblivious of each other. This disconnect has often resulted in reinvention of the wheel. The primary reason is that the pooling problem is typically identified in the context of a particular application and a solution is sought within that context. Translating the solution to other applications that attempt to solve the same problem is made difficult by the language barrier between the fields involved in these applications. The very nature of the fields connected to these applications, namely mathematics, statistics, computer science, engineering, and a variety of subfields of molecular biology, give us an idea of the linguistic barriers that the pooling idea has to cross. This thesis is an attempt to bring some of these disparate threads together and reconcile the theory and practice of pooling designs. While attempting to do so, it also aims to apply this generalized approach to solve particular problems in high-throughput biological experiments. One of the goals of this thesis is to establish a channel of communication between the mathematical aspects of the problem and the practical constraints of actual experimental platforms. This would enrich the mathematical theory by supplying new problems to solve and correspondingly supply comprehensive solutions to address the practical challenges. Further, to make the pooling design approach useful for immediate application to various high-throughput experiments, the results in the thesis were packaged into a set of tools along with software implementation. Three of the chapters in the thesis share their title with the corresponding tool developed within them.

In Chapter 2, I provide an overview of the pooling design approach in the context of high-throughput drug screening (HTS). Using a standard example, I describe three different pooling design methods and make a case for one of them as being most suitable for HTS applications. Next, I review a variety of attempts to test pooling designs in practice and describe their success. Finally, I describe the challenges of implementing pooling design strategies with existing experimental systems.

In Chapter 3, I continue with a detailed description of a pooling design construction called the Shifted Transversal Design (STD) and its suitability for HTS applications. Next, I explore the mathematical properties of this design strategy and identify some of the limitations in its practical application. Finally, I adapt the STD strategy to fit the constraints of HTS and successfully apply it using existing experimental set-ups. The outcome of this effort was a tool called poolHiTS – pooling in high-throughout screening.

Chapter 4 has two aspects. First, I explore practical approaches to decoding the results of a pooled experiment. I describe two decoding methods  one that focuses on finding a general decoding problem in the pooling design framework; another that I developed to exploit the large amount of quantitative information inherent in high-throughput screens. The second method, called QUAPO – quantitative analysis of pooling, led me to identify connections between the pooling problem and a new development in the field of signal processing called compressive sensing. Using a biochemical model of pooling, I transformed the decoding problem to one that could use these novel quantitative decoding methods. Second, I tested and validated the pooling design described in Chapter 3 in a real high-throughput screen. I describe the implementation and performance of the two decoding methods using data from the screen and describe the advantages and challenges of using each method in the context of HTS.

In Chapter 5 I extend the use of the quantitative methods developed in Chapter 4 to a novel application – gene expression microarray experiments. Microarray experiments have become an important tool in advancing the genomics revolution. However, they are expensive to run and often provide noisy data. In this chapter I propose a novel pooling strategy to make certain types of microarray experiments cost-effective and robust to measurement noise. I tested the smart pooling approach in a proof-of-concept experiment. In the context of this experiment, I discuss the potential, challenges, and future directions of using pooling designs in mircoarray experiments. The tool that came out of this application is called poolMC.

Finally, in Chapter 6, I summarize the state-of-art in pooling design theory and application. I present challenges and future directions for the pooling design and decoding approaches, and highlight potential candidate solutions from recent literature. In addition, I discuss an interesting new application of pooling designs to population genotying using next-generation sequencing technology. Finally, I propose a set of potential applications of pooling in other, as yet untested, biological experiments.

# Chapter 2
# Pooling in high-throughput drug screening

## 2.1 Introduction

Traditional high-throughput screening (HTS) has provided impressive gains to drug discovery. However, advances in genomic biology and combinatorial chemistry have led to an increasing number of targets and compounds to screen. The past trend of reducing assay size is aiding HTS to keep pace with the increasing target lists, but miniaturization alone is insufficient (Dove, 1999). One of several ways to improve the efficiency of HTS is pooling, where mixes or pools of compounds are tested in each well of a HTS assay rather than individual testing of each compound. Because most compound libraries contain only a small fraction of active compounds, pooling can reduce the number of tests needed to identify them. The central rationale of pooling is that most compounds can be quickly identified as inactive via a negative result of a pooled test. Because of this potential, pooling has received intermittent attention from the screening community, which is evident from the references cited in this article.

This chapter discusses the complexities and solutions to pooled drug discovery, and focuses on the following five topics: (i) design of pooling schemes; (ii) experimental success of pooling; (iii) mixing constraints and cut-off selection of pooling; (iv) additive, synergistic and antagonistic effects of pooling compounds; and (v) recent advances in pooling. One of the goals of this thesis is to create a repository for literature collected from a very diverse set of journals and researchers in the hope of facilitating a fruitful collaboration of ideas among the HTS community.

## 2.2 Pooling Designs

Currently, most high-throughput screens employ the *one compound, one well* approach. This approach is simple in terms of both the physical implementation of the assay and the analysis of results (Walters and Namchuk, 2003; Janzen, 2002). However, given the usually small number of active compounds in a library that is typically comprised of a much large number of compounds, this approach can be wasteful. Also, HTS screens are prone to several forms of experimental error, which produce false positive (inactive compound classified as active) and false negative (active compound classified as inactive) results during analysis (Malo et al., 2006). False positive results are less of a problem because these compounds are retested in subsequent stages. In contrast, false negative results represent potential lead compounds that are lost to the development process because replicate testing of a compound library is prohibitively costly. Therefore, there is a need to improve the efficiency and robustness of screening, and pooling of compounds presents a solution to this.

The concept of testing pools of samples in biological experiments was conceived during World War II, when blood samples from cadets were pooled to optimize syphilis testing (Du and Hwang, 2000). Over several decades, the mathematical theory of pooling has undergone tremendous advances and has found applications in diverse areas (Du and Hwang, 2000; Bruno et al., 1995; Jones and Zhigljavsky, 2001; Eppstein et al., 2005; Hughes-Oliver, 2006). An excellent introduction to the subject can be obtained from the book written by Du and Hwang (2000). There have been several applications of pooling in high-throughput screens with varying success. These attempts used a variety of pooling schemes, which are reviewed immediately below while results of these schemes will be discussed in the following section.

Pooling schemes can be broadly divided into two categories: adaptive and nonadaptive. Adaptive schemes involve testing pools of compounds in stages, whereby information gained from one stage is used to design the next stage. The resultant gain in reduction in the number of assays required is, however, offset by the delay in obtaining the results of a screen. In contrast, nonadaptive schemes involve testing all of the pooled compounds in a single stage, typically testing each compound multiple times. Many of these nonadaptive schemes guarantee the accurate identification of a specified number of active compounds per library. Aspects of both of these strategies have been merged to form hybrid methods, details of which are available elsewhere (Du and Hwang, 2000; Bruno et al., 1995; Jones and Zhigljavsky, 2001). In the following discussion, four types of screen designs ('one compound, one well', adaptive pooling, orthogonal pooling and nonadaptive pooling strategies) using a hypothetical example of a 2400-compound library (25 plates of 96 compounds each) with 24 truly active compounds (1% hit rate) are presented. Furthermore, a detailed analysis of pooling designs for a subset of 25 compounds comprising 2 active compounds is presented to enable the properties of various pooling schemes to be highlighted. The probability of finding 0, 1, or 2 active compounds in a subset of 25 compounds selected from a larger set of 2400 compounds with 24 active compounds is 99.8%, based on the use of the hypergeometric distribution.

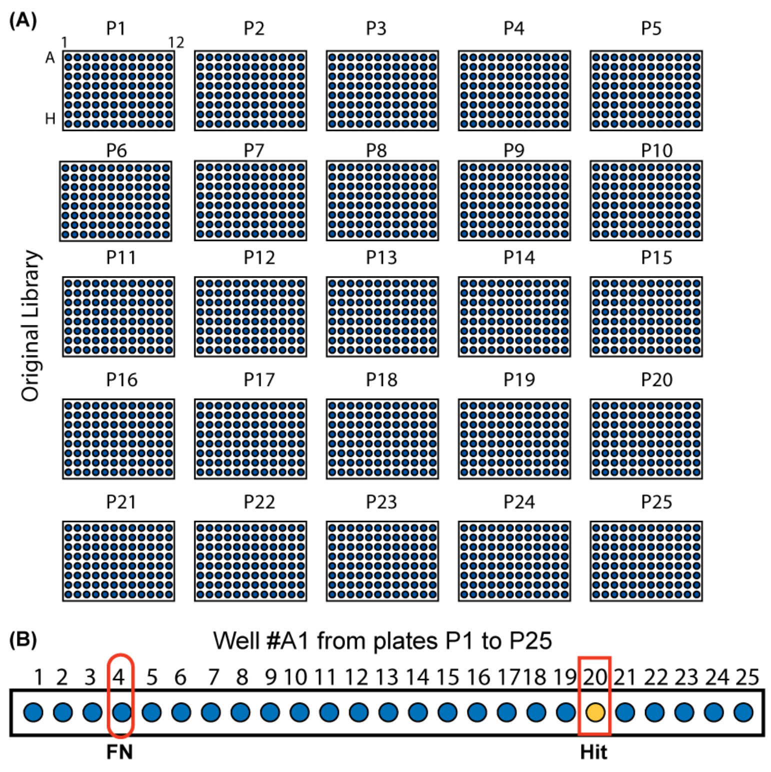

**Figure 2.1** One compound, one well strategy: Distribution of a 2400-compound library into 25 plates. (A) Distribution of 2400 compounds over 25 plates in the *one compound, one well* strategy for HTS. (B) The analysis for a subset of 25 compounds (position A1 on plates P1 to P25); compounds number 4 and number 20 (highlighted) are active. A false negative (FN) result in well number 4 results in the incorrect designation of compound number 4 as inactive. Compound number 20 is correctly identified as active (yellow-colored well).

### 2.2.1 *One compound, one well* strategy

The *one compound, one well* strategy tests each compound individually and hence requires 2400 wells for 2400 compounds. Figure 2.1 shows the original compound library distributed over 25 plates (P1 to P25), which contain 96 compounds each, and magnifies the results for a subset of 25 compounds in position A1 on these plates. If compounds number 4 and number 20 (position A1 on plates P4 and P20) were truly active but one false negative testing error occurred, the results from a standard detection method would indicate only one compound, for example, compound number 20, as active (highlighted in yellow in Figure 2.1). In this design, the benefit of a simple design and analysis method is obtained at the cost of testing a large number of inactive compounds. Additionally, any errors in testing cannot be detected or corrected in this primary stage of HTS. Therefore, there is a need for more efficient and robust screening methods, such as the three pooling designs described below.

### 2.2.2 Adaptive pooling strategy

The adaptive pooling designs exploit the presence of large numbers of inactive compounds by dividing a compound library of size *n* into pools of size *r* (ie, *r* compounds are mixed in each assay well) (Jones and Zhigljavsky, 2001; De Bonis et al., 2005; Bar-Lev et al., 2006). In subsequent stages, the pools that exhibited activity are retested to identify which compounds were responsible for the activity. The number of active compounds or *hits* is typically denoted by *d*, the abbreviation for defectives which is the term used in fault-testing literature.

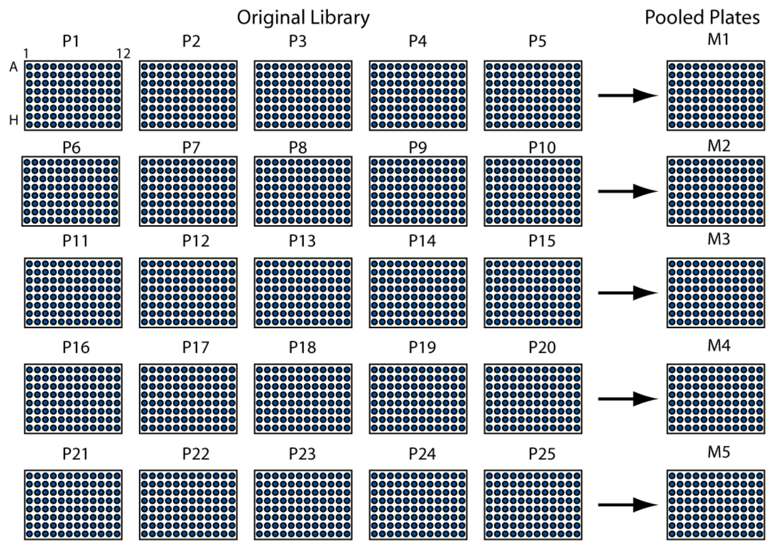

**Figure 2.2** Adaptive pooling strategy: A 2400-compound library pooled into five plates. The adaptive pooling strategy implemented at the plate level, by pooling groups of five plates of compounds into one pooled plate which is screened for activity.

As shown in Figure 2.2, the 25 plates are pooled in groups of five (M1 to M5). For example, plates P1 to P5 in the original library are pooled and tested in a single plate, M1. This translates to the pooling scheme shown in Figure 2.3 (Stage 1) for a subset of 25 compounds ($n = 25$) in position A1 of each plate (M1 to M5) tested in pools of five compounds ($r = 5$). Because compounds number 4 and number 20 (position A1 on plates P4 and P20) are the active compounds, pools M1 and M4 (wells A1 from plates M1 and M4) should produce a positive response and only compounds in these pools are carried over to the next stage. However, a false negative result in pool M1 would cause compound number 4 to be considered as inactive. Several similar stages can be conducted, but in this example only two stages are required to identify compound number 20. In the second stage, each of the five A1 compounds in pool M4 is tested in individual wells (M6 to M10, Stage 2; Figure 2.3) to identify compound number 20 as active. The total number of wells of compounds required to be tested to identify compound 20 as active is 10 (5 in Stage 1 and 5 in Stage 2), instead of 25 for the *one compound, one well* setting. Adaptive designs provide valuable savings in resources by minimizing the number of wells used, but the resultant procedure from such designs can be time consuming to conduct. In general, adaptive designs require the testing of a maximum of $d\log_2 n$ wells to identify $d$ active compounds out of $n$ compounds, when no limit on the number of compounds mixed in each well is imposed (see Chapter 2 of (Du and Hwang, 2000)). Simple adaptive designs such as the one adopted for this example are vulnerable to testing errors, where an error in a single well in an early stage can strongly influence the overall assay accuracy, as depicted in Figure 2.3. Hybrid designs that test pools multiple times in each stage can be used to reduce the sensitivity of an assay to testing errors.

### 2.2.3 Orthogonal pooling strategy

The orthogonal pooling or self-deconvoluting matrix (SDM) strategy can be considered to be an intermediate form of pooling and has been used in a number of screens that have been subsequently published (Devlin et al., 1996; Motlekar et al., 2008; Warrior et al., 2007; Ferrand et al., 2005). In orthogonal pooling, each compound is tested twice, each time in combination with a completely different set of compounds. Therefore, for an $n$ compound library tested in pools of size $r$, this strategy would require $\frac{2n}{r}$ wells to be tested. When more than two compounds are pooled in each well ($r > 2$) in orthogonal pooling, the number of wells that require testing is an improvement from using $n$ wells in the *one compound, one well* strategy. In orthogonal pooling, each compound is required to show activity in both wells in which it is tested in order to be classified as a hit. Although this method does not provide robustness to testing error, it does conserve both resources and time by conducting a reasonably efficient screen in a single stage. However, even in the absence of testing error, false positives occur when an inactive compound is tested both times with a different active compound (illustrated in Figure 2.5). This failure probability and the inability to correct or detect testing errors are the major drawbacks of the orthogonal pooling scheme.

The application of an orthogonal pooling scheme to the 2400-compound library example

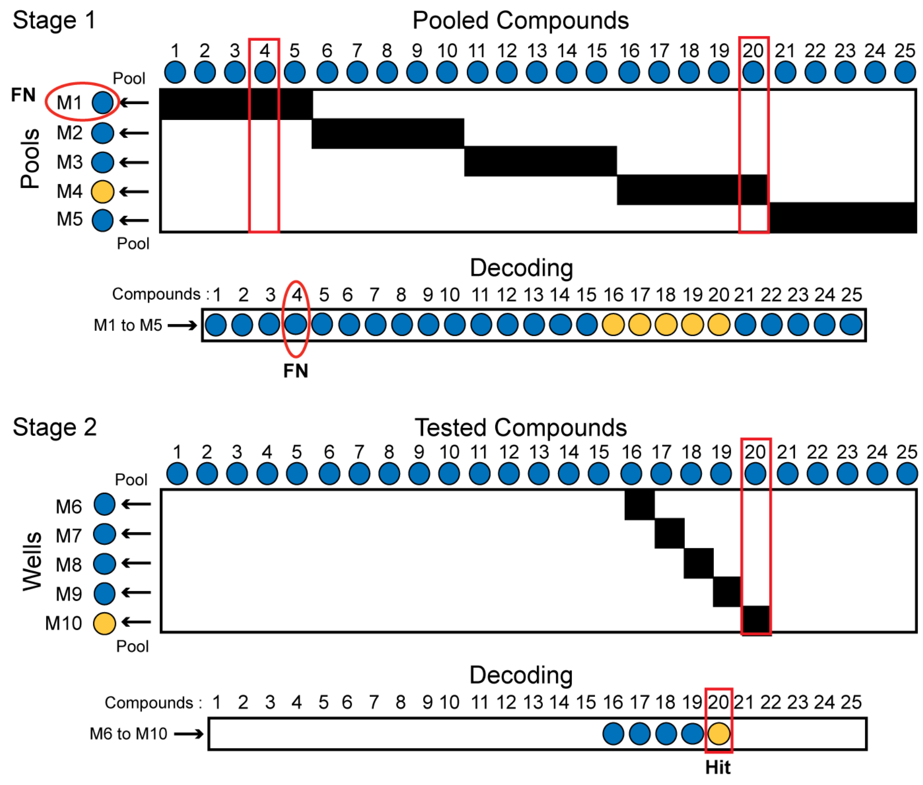

**Figure 2.3** Adaptive pooling analysis: The interpretation of adaptive pooling strategy is shown for a subset of 25 compounds present in well A1 of each plate in the original library in Figure 2.2. The pooling scheme is conducted in two stages. Stage 1 eliminates 20 compounds (via the negative results in M1, M2, M3 and M5). A false negative (FN) results in well M1 causes compound number 4 to be incorrectly designated inactive. Stage 2 tests the compounds from pool M4 individually to identify the active compound (20), via well M10.

is shown in Figure 2.4. In this design, plates are pooled once along each row and once along each column of the original library, allowing each compound to be tested twice but with a different set of compounds each time. The 25-compound subset in position A1 in all pooled plates (M1 to M10) translates to the design shown in Figure 2.5. Compound number 1 in the original library (in well A1 in plate P1) is tested once in pool M1 with compounds number 2, 3, 4 and 5 (row pools) and tested again in pool M6 with compounds number 6, 11, 16 and 21 (column pool). This method uses 10 wells (5 wells from the row pool, ie, M1 to M5, plus 5 wells from the column pool, ie, M6 to M10) compared to 25 in the *one compound, one well* strategy. The status of a compound is determined by examining the result from both sets of wells. A compound is considered to be active only when it is active in both sets of wells. As seen from the decoding diagram at the bottom of Figure 2.5, only those compounds with two active results (yellow circles) are considered as active. Therefore, compound number 20 is correctly identified as a hit (highlighted with a red rectangle), while compound number 4

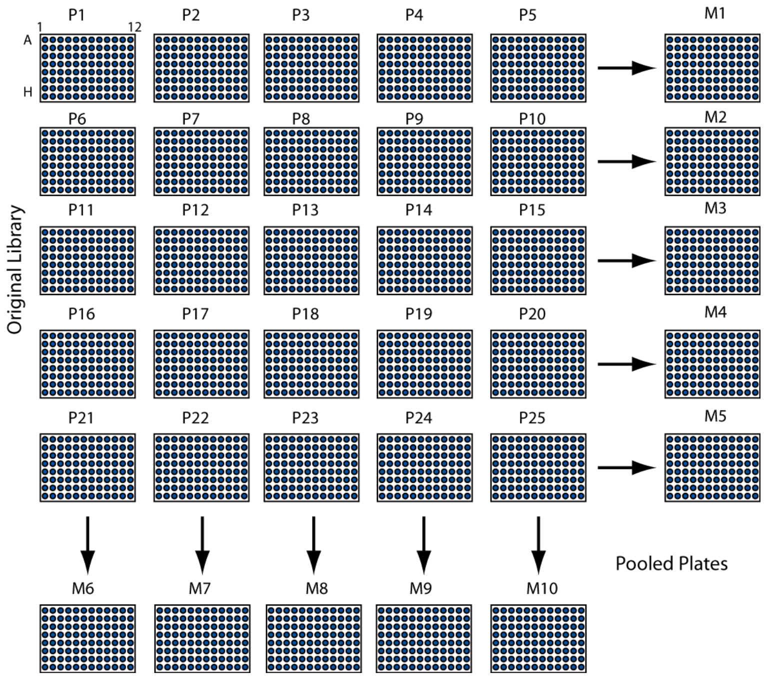

**Figure 2.4** Orthogonal pooling strategy: A 2400-compound library pooled into ten plates. An illustration of an orthogonal pooling scheme with a 2400-compound library. Each compound is tested in two pools, one created by mixing plates along each row of the original library and the other by mixing plates along each column.

is missed owing to a false negative result in pool M9. In addition, compound number 5 was pooled twice with active compounds and is wrongly identifies as a hit (false positive). This example illustrates the risk associated with orthogonal pooling. However, this false positive risk is minimized when the fraction of active compounds in a library is small.

## 2.2.4 Nonadaptive pooling strategy

The nonadaptive pooling schemes, which I regard to be the most suited for HTS, are more difficult to design and implement than the schemes of the three previously described strategies. However, the nonadaptive pooling strategy offers a reduction in the number of assays used and the ability to detect and correct testing errors in a single stage (Eppstein et al., 2005; Huang, 2004; Du and Hwang, 2006; Thierry-Mieg, 2006a). Nonadaptive pooling schemes test each compound multiple times across a single screen and in combination with a different set of compounds, using a low overall number of assays. The results from the pooled tests are then jointly decoded via an analysis algorithm to identify the active compounds. Testing each compound multiple times enables nonadaptive pooling methods to detect and

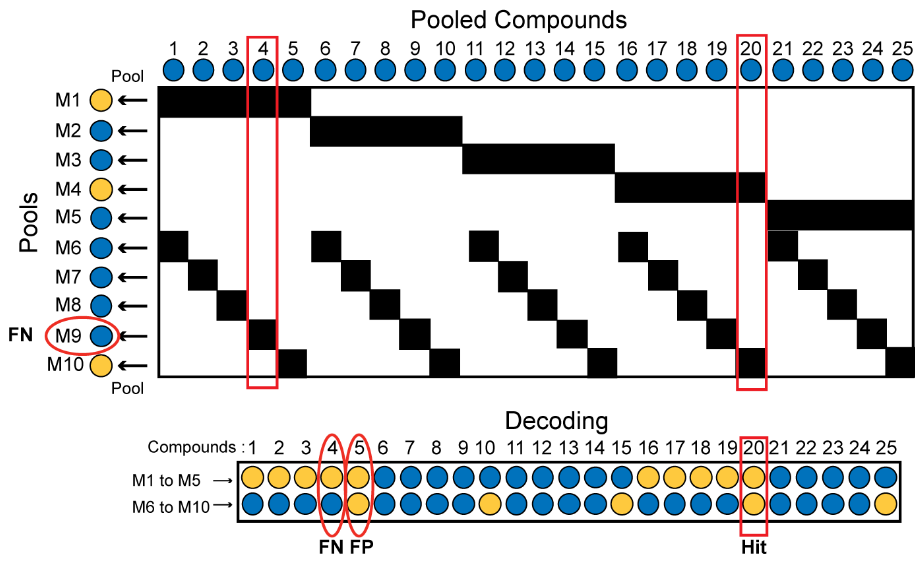

**Figure 2.5** Orthogonal pooling analysis: The analysis of the pooled wells is shown for the subset of compounds present in well A1 of plates P1 to P25 in Figure 2.4. Compounds present in two active wells are marked as active. However, owing to the false negative (FN) testing error in well M9, compound number 4 is incorrectly labeled as inactive. Furthermore, the design contains one false positive (FP) result with compound number 5 because the compound was tested twice in the presence of two different actives (ie, compounds number 4 and 20). Compound number 20 is correctly identified as a hit.

even correct experimental errors. Such schemes are designed using advanced mathematical algorithms that account for compound library size, expected hit rates, expected error rates, pool size, etc. Furthermore, these algorithms are used to construct mixing schemes that are guaranteed to identify all the active compounds successfully, even in the presence of testing errors. Because nonadaptive designs are mathematically complex, these schemes are typically represented as a binary matrix showing which compounds (represented along the columns of the matrix) are to be pooled in which wells (represented along the rows). Random nonadaptive pooling designs also exist that require fewer pools than constructed nonadaptive pooling designs and which can be created to guarantee a high, but not perfect, probability of success (Du and Hwang, 2006; Hwang and Liu, 2003).

An illustration of the application of a nonadaptive pooling scheme with the 2400 compound test case is shown in Figure 2.6. The black squares along a row correspond to the plates (P1 to P25) with compounds that are pooled together and tested on a single plate (M1 to M20). In this example, the plates are pooled according to a strategy called the Shifted Transversal Design (STD) created by Thierry-Mieg (2006a) and adapted to drug screening by Kainkaryam and Woolf (2008). The details of this pooling design are discussed in greater detail in the Chapter 3. The 25 compounds, corresponding to position A1 on these plates,

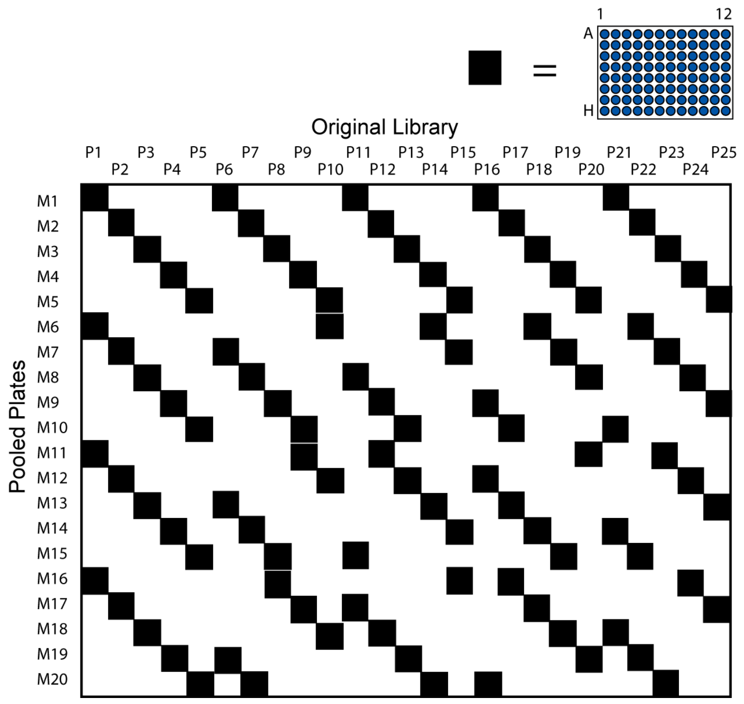
1
12
A
H
=
Original Library
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16 P17 P18 P19 P20 P21 P22 P23 P24 P25
Pooled Plates
M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13 M14 M15 M16 M17 M18 M19 M20

**Figure 2.6** Nonadaptive pooling strategy: A 2400-compound library pooled into 20 plates. A nonadaptive pooling scheme using the Shifted Transversal Design pooling strategy, where black squares represent the plates (P1 to P25; along that column) to be pooled together to form a pooled plate (M1 to M20; along that row). 20 pooled plates are used to screen 25 plates of compounds, resulting in a 20% reduction in the number of plates to be analyzed.

are arranged along the columns of the binary matrix shown with black and white squares in Figure 2.6 (and Figure 2.7). A black square indicates that the compound along that column is pooled in the wells corresponding to that row. Hence, compound number 1 is pooled in wells M1, M6, M11 and M16, along with other compounds specified by the mixing matrix.

In this example, each compound is tested four times using 20 wells, thus reducing the number of wells to be tested and analyzed by 5 (a 20% reduction), and each assay contains a pool of five compounds. The decoding method for such a pooling scheme is more involved (details available in Thierry-Mieg (2006a)) than the three previously described pooling schemes, but a simplified version is shown at the bottom of Figure 2.7. Any compound present in at least two wells showing inactivity (blue-colored wells) is considered as inactive. Furthermore, any compound present in at least two wells showing activity (yellow-colored wells), where all other compounds within the same wells are inactive, is considered as active. Despite the false negative error in pool M9, the nonadaptive design is able to accurately decode the true active compounds. This scheme has been explicitly designed to guarantee the identification of up to two active compounds out of 25 (a hit-rate of 8% across all plates)

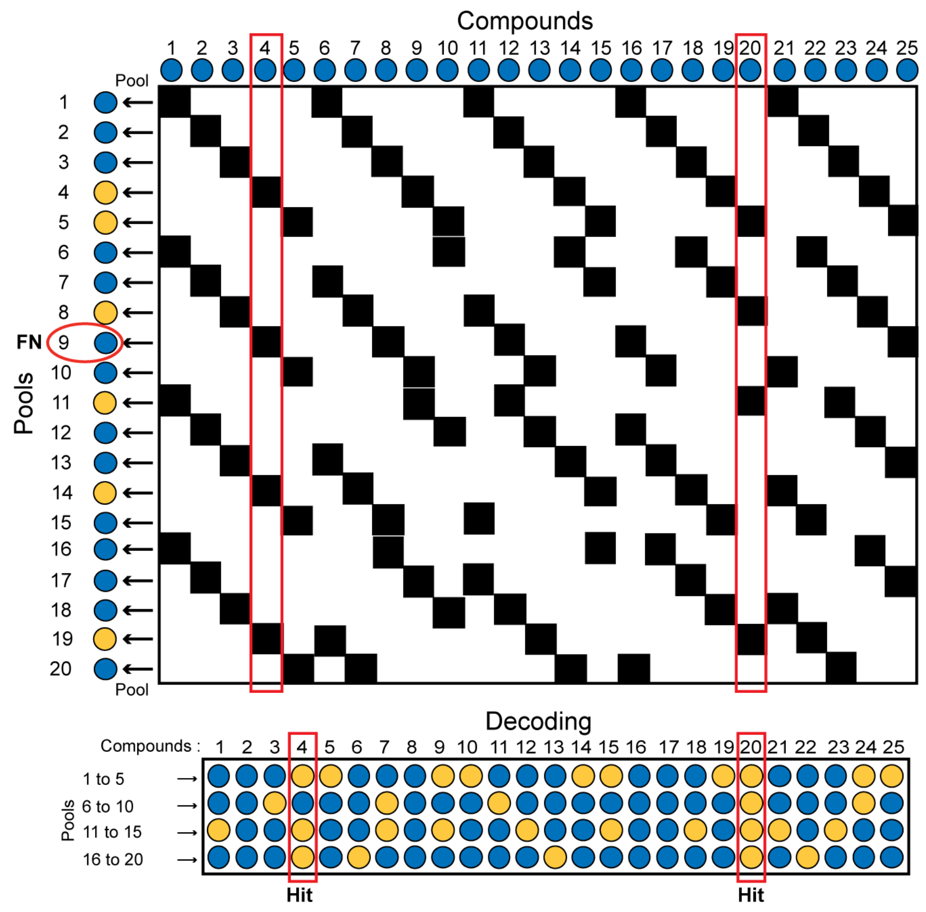

**Figure 2.7** Nonadaptive pooling analysis: The analysis of a nonadaptive pooling scheme for a subset of 25 compounds (well position A1 on plates P1 to P25 in Figure 2.6) is shown. Decoding the activity of compounds from pooled assay results is conducted by labeling those compounds present in at least two active wells in which all other compounds are inactive, as active. Despite the false negative (FN) testing error, compounds number 4 and 20 are correctly identified as active.

and to prevent the occurrence of false positive results, unlike the orthogonal pooling strategy at the cost of the need for ten extra wells. Of note is that despite testing each compound multiple times, the total number of assays needed in a nonadaptive pooling scheme can be lower than the *one compound, one well* strategy. Typically, nonadaptive schemes require approximately $d^2 \log_2 n$ wells to guarantee the identification of $d$ active compounds in an $n$-compound library (Porat and Rothschild, 2008).

## 2.3 Review of experimental results of pooling

This section reviews cases in which adaptive pooling, orthogonal pooling and nonadaptive pooling have been used in published drug screens. These examples are far from exhaustive, but highlight the reasoning that is used in selecting a pooling design for screening. Pooling has been successfully used for screening solid-phase libraries, combinatorial libraries, natural product extracts, etc (Ferrand et al., 2005).

One area where adaptive pooling strategies have been successfully used is in combinatorial library screening. This approach takes advantage of combinatorial synthesis of compounds to produce mixtures of compounds, which typically have some common functional unit. In one such study, 810 compounds synthesized from two-way and three-way combinations of monomer units, taken from a nine-monomer set, were tested in a human type II phospholipase ($PLA_2$) inhibition assay using an adaptive pooling strategy (Wilson-Lingardo et al., 1996). At each stage of the adaptive test, pools of compounds having a common monomer unit were tested together, resulting in the testing of 27 compounds per well in the first stage. The pool with the highest inhibition was progressed to the next stage, where the common monomer unit from the previous stage was preserved, while a second monomer position was varied for each pool. This process was repeated for all three monomer positions and the screen finally led to the compound with the three-monomer unit with the best $PLA_2$ inhibitory activity. Simulated measurement errors of approximately 7% for high-activity and 16% for low-activity pools were reported for this study. At these levels of experimental error, an adaptive strategy that does not use replicate measurements can be successfully used for screening. Similar combinatorial synthesis approaches to pooling have been conducted experimentally and reviewed elsewhere (Houghten, 2000).

A number of examples of the reasonable successful use of orthogonal pooling by the screening community have been reported (Devlin et al., 1996; Motlekar et al., 2008; Warrior et al., 2007; Ferrand et al., 2005). In a recent study, Motlekar et al. (2008) tested 64,000 compounds in a human cysteine protease cathepsin B assay, using both the *one compound, one well* approach and an orthogonal pool of ten-compound mixtures per well. Twenty active compounds were identified via the orthogonal pooling procedure, including three compounds that were missed (false negative) by the *one compound, one well* strategy. Similarly, Warrior et al. (2007) conducted a 10-compound per well orthogonal pooling study on three different screening formats (a GPCR functional assay, a GPCR binding assay, and a tyrosine kinase assay) and showed that the performance of pooling was equal to or better than *one compound, one well* screening in all cases. For cases in which the first stage results were ambiguous, the standard orthogonal pooling strategy was augmented with an additional retesting stage. Despite this additional feature, a cost comparison study showed that orthogonal pooling in this instance was significantly cheaper and faster than the *one compound, one well* strategy. Furthermore, in a comprehensive study published in 2005, Ferrand et al. (2005) rigorously tested a library of 26,400 compounds in a CXCR3 scintillation proximity assay by both *one compound, one well* (repeated twice) and orthogonal pooling (repeated once) approaches. Orthogonal pooling was found to significantly reduced resources and time, as well as false positive errors without increasing false negative errors. Data from all five screens (three in *one compound, one well* and two in orthogonal pooling) were used to identify 64 *true actives*, against which the statistical performance of the pooling approaches was evaluated.

Although the theoretical performance of nonadaptive designs are superior to orthogonal designs, nonadaptive pooling has not yet been tested in a drug screen to the best of the knowledge of the authors of this review. Strong theoretical and computational results suggest that nonadaptive pooling has the potential to be successfully applied to HTS. In 2006, as already mentioned, Thierry-Mieg published the general nonadaptive pooling strategy STD using a mathematical construction, which is the best-known algorithm to date (Thierry-Mieg, 2006a). The STD algorithm takes in, as inputs, the compound library size ($n$), the maximum number of active compounds expected ($d$) and the maximum number of false positive and negative testing errors expected ($E$) to design a pooling strategy that guarantees the successful identification, in a single stage, of $d$ active compounds in the presence of $E$ testing errors, while maintaining a lower number of tests than that required by the 'one compound, per well' approach. The STD algorithm achieves compression and accuracy by testing each compound multiple times in unique pools to guarantee the identification of all d active compounds, even when testing errors occur. The example in Figure 2.7 shows an STD strategy for $n = 25, d = 2$ and $E \sim 1$, using 20 pooled wells to test 25 compounds. The details of the STD design and its application to HTS are discussed in greater detail in Chapter 3. A positive feature of nonadaptive designs is their ability to complete a screen in a single stage, thereby avoiding errors caused by day-to-day variability. Ferrand et al. (2005) showed that day-to-day variability has a stronger influence on the screening results of standard procedures compared with any differences due to the use of *one compound, one well* or pooling strategies .

## 2.4 Note on pool size and choice of threshold

Most pooling schemes in HTS are constrained by the number of compounds that can be mixed in a well. The constraint arises from factors such as: (i) physical limitations of well size; (ii) effect of ionic strength of pool on compounds or solvent on assay; and (iii) side effects of pooling, such as synergistic and antagonistic interactions between compounds (discussed in detail in the next section). Therefore, pooling designs used for drug screening include the parameter $r$, which as already explained represents the limit on compounds that can be pooled in a well. In most designs, this limit on pool size also defines the upper limit of the possible compression and error correction that can be gained by pooling in HTS.

A second issue is the definition of a cutoff to define an assay result as a positive or negative result. In a pooled screen, due to the presence of multiple compounds, the concentration of each compound in the well may be different than if the assay were run using a *one compound, one well* design. Generally, for a compound showing inhibition (%I) at a certain concentration ($[C]$), the $IC_{50}$ value can be calculated by the following expression (equation 2.1) (Motlekar et al., 2008):

$$\mathrm{IC}_{50} = [C]\frac{100 - \%I}{\%I} \tag{2.1}$$

Assuming that pooled screens would test compounds at lower concentrations than a *one compound, one well* screen, for a compound tested in both formats, the following relationship (equation 2.2) exists between the compound concentration in both formats ($[C]_S$ and $[C]_M$ for single and pooled formats, respectively) and the corresponding cutoff inhibitions ($\%I_S$ and $\%I_M$ for single and pooled formats, respectively) (Motlekar et al., 2008):

$$[C]_S\frac{100 - \%I_S}{\%I_S} = [C]_M\frac{100 - \%I_M}{\%I_M} \tag{2.2}$$

The expression for the choice of cutoff inhibition in a pooled screen (equation 2.3) can be obtained by rearranging equation 2.2. $\%I_M$ corresponds to an equivalent choice of cutoff inhibition in a *one compound, one well* screen by accounting for the difference in the testing concentration of the two screening formats.

$$\%I_M = \frac{[C]_M}{[C]_S}\frac{\%I_S}{100 - \%I_S(1 - \frac{[C]_M}{[C]_S})} \times 100 \tag{2.3}$$

Equation 2.3 represents the cutoff inhibition above which a pooled well can be considered to display activity corresponding to the presence of at least one active compound tested individually. Using the example from Motlekar et al. (2008), if a compound tested individually at 10 $\mu$M ($[C]_S = 10$) is labeled as a *hit* for producing 33% inhibition ($\%I_S$), then the same compound in a well containing several other *inactive* compounds tested at 5 $\mu$M ($[C]_M = 5$) each, would have to show at least 20% inhibition to be termed a *hit*:

$$\%I_M = \frac{5}{10}\frac{33}{100-33(1-\frac{5}{10})} \times 100 = 20 \tag{2.4}$$

## 2.5 Additivity, Synergies and Antagonism

Some of the problems that could arise from pooling of compounds in HTS have featured predominantly in the debate about the use of pooling (Chung, 1998; Snider, 1998). The following three issues will be discussed: additivity, synergies and antagonistic effects. Additivity assumes independent action of compounds in pools, while synergy and antagonism refer to interactions between compounds in pools. Recent efforts to handle these three issues are described in section 2.6.

Pooling several moderately active compounds together in a well, such that the combined activity of the compounds mimics the appearance of an single active compound, leads to the production of an additive effect. From the previous example in the section *Pool size and choice of threshold*, if two compounds that individually produce only 20% inhibition at 10 $\mu$M were pooled together at 5 $\mu$M each, the result from the pooled well would be an observed inhibition of greater than 20%, assuming both compounds acted independently. Therefore, such a pooled well would be considered a *hit*, using the previous criteria for pooled wells, resulting in a false positive. Researchers who have published drug screens that used pooling have not searched systematically for this effect, although several have reported finding false positive results Motlekar et al. (2008); Warrior et al. (2007); Ferrand et al. (2005). An alternative and favorable view of such false positive results is that additive collaboration of non-interacting compounds could ultimately lead to potential multi-component therapies.

Pooling of compounds could also produce physiochemical effects, such as a reaction or aggregation. Consequently, such compounds could achieve a positive result (synergy) or negate the activity of each other (antagonism or blocking). Typically, pooling designs treat these interactions as false positive (synergy) and false negative (antagonism) testing errors and attempt to correctly identify them by including retesting steps. However, several research groups have systematically screened multi-compound mixtures to identify useful synergistic effects, with reasonable success (Borisy et al., 2003; Feng and Shoichet, 2006). For example, Feng and Shoichet (2006) studied synergistic and antagonistic effects caused by aggregation-based behavior of compound mixtures. They screened 764 compounds individually and in pools of 10 compounds and compared the observed pooled behavior with that predicted from individual tests. They found that synergistic effects predominated, although antagonism was also observed.

There is great potential for smart pooling design and analysis strategies that successfully identify individual hits while: (i) minimizing resources; (ii) providing error tolerance; and (iii) retaining the ability to identify synergistic effects. Some of the above goals are conflicting, but recent advances in pooling design theory promise reconciliation among them, as discussed in the following section.

## 2.6 Advances in Pooling

This section presents recent (mainly) theoretical advances that attempt to address or reconcile the multiple goals of pooling designs. These advances involve both the design of smart pooling strategies as well as the application of wide ranging methods to analyze the results from pooled screens. Typically, these analysis methods are computation-intensive techniques that shift the emphasis of the pooling from experimentation toward assay design and analysis.

Several approaches have been proposed to prevent interactions among compounds during pooling, such as: (i) reducing the number of compounds pooled; (ii) pooling dissimilar compounds; and (iii) systematically preventing chemical classes such as electrophiles and nucleophiles from mixing (Warrior et al., 2007; Hann et al., 1999). For adaptive pooling designs, computational methods have been designed that use chemical structure information to design optimal pools for maximizing coverage of the chemical space, while minimizing overlap (Remlinger et al., 2006). For orthogonal pooling, simulation techniques have been proposed that predict probabilities for the occurrence of synergy and blocking (antagonism) to help select the most suitable efficient pooling and retesting schemes (Xie et al., 2001).

Given the increasing complexity of the goals of pooling schemes, nonadaptive pooling designs appear to be the most promising strategy for screening, because such designs typically test each compound several times during a screen. The resultant increased amount of data allows for analysis schemes that explicitly account for possible antagonists and exhaustively search for pair-wise or higher-order synergies (see Chapters 6 and 8 of Du and Hwang (2006) for details). Furthermore, the mathematical guarantees of most non-adaptive designs are provided for worst-case scenarios, while on average these designs can identify more active compounds and correct larger testing errors (Thierry-Mieg, 2006a). A simulation-based approach has been proposed to exploit the abilities of nonadaptive pooling to design better pooling strategies and decoding methods (Thierry-Mieg and Bailly, 2008). In keeping with this trend of increasing computational efforts to address the complex goals of pooling, a Bayesian network pool decoder has been proposed that generalizes the analysis problem (Uehara and Jimbo, 2009). This Bayesian network decoder can incorporate prior knowledge of compound activity and various possibilities for measurement error and can potentially be extended to specifically identify synergistic or antagonistic effects. Together, these approaches promise to sustain the success of high-throughput drug screening.

## 2.7 Conclusions

The recent decline in success of the application of the *one compound, one well* strategy to high-throughput drug screening suggests that the strategy could benefit from further optimization. Pooling of compounds in HTS offers several potential gains, not limited to resource savings and error tolerance. Pooling also offers to address some of the critical issues experienced by HTS, such as the rapid increase in compounds and targets to be screened, the ever-present possibility of false negative testing errors, and the search for new classes of multi-compound, multi-target therapeutics. There are several challenges to the success of pooling in HTS, including the creation of optimal pooling design and analysis strategies, physical implementation of these pooling strategies, and management of synergistic or antagonistic behavior of compounds. Some of these challenges are addressed in Chapters 3 and 4 and represent opportunities to extend the current capabilities of HTS in drug discovery. Collaborations between the currently disjointed theoreticians and practitioners of these methods would greatly enable these challenges to be met and opportunities to be realized.

# Chapter 3

# poolHiTS – A shifted-transveral based pooling strategy for high-throughput drug screening

In 2006, a novel pooling design method called shifted transversal design (STD) was introduced for biological assay design (Thierry-Mieg, 2006a). STD is based on the dual objectives of (1) minimizing the number of times any two compounds appear together in a test and (2) maintaining the pool sizes roughly equal. When compared to other pooling designs, STD provides similar or better performance in nearly every area, as illustrated in Chapter 2.

In this chapter, I introduce a pooling strategy called poolHiTS, based on the STD algorithm, specific for the purposes of drug screening. First, I prove a limit to the error-correcting capacity of STD-based pooling strategies. This limit is important for drug screening as it determines the error rates that can be successfully overcome by a pooling design when applied to high-throughput screening (HTS). Second, I modified the pooling design algorithm to limit the number of drugs tested in each assay, thereby enforcing a realistic experimental constraint of HTS. Third, I introduce a block design method that both simplifies and improves the assay design.

## 3.1 Preliminaries

A STD-based pool construction starts with the specification of the compound library size ($n$), maximum number of active compounds expected ($d$) and maximum number of errors expected ($E$). The STD algorithm guarantees that the pooled design will be able to correctly identify up to $d$ active compounds in the presence of up to $E$ false positive and negative errors in the screen. STD is able to provide such guarantees because it uses a combinatorial procedure to ensure that no two compounds are pooled together more than a minimum number of times, to prevent confounding decoding results. Also, the number of compounds pooled in each test is roughly the same, ensuring correct intensity-concentration mapping for the test results. This implies that for any underlying structure of compound activities and testing errors, as long as their numbers lie within the specified experimental parameters, the STD design guarantees successful identification of the active compounds. It has been observed in Thierry-Mieg (2006a) and Thierry-Mieg and Bailly (2008) that pooling designs are capable of correcting errors much larger than those that they guarantee for. These input parameters $(n, d, E)$ are used to choose the design parameters of the STD construction algorithm $q$ and $k$. STD is a layered construction with $k$ layers, each of size $q \times n$. Each compounds appears only once in each layer. The STD construction algorithm produces a $t \times n$, 0-1 matrix $M = \text{STD}(n; q; k)$, with $t$ $(= q \times k)$ rows that are the assays to be performed and $n$ columns that represent the compounds in the library. The columns with entry 1 in a row are the compounds to be pooled in that assay. An example of such a pooling design is shown in Figure 3.1, created using $M = \text{STD}(20; 5; 3)$ for $n = 20$, $d = 2$ and $E = 0$. The process of mapping the experimental parameters to the design parameters is shown below in Algorithm 3.1.

**Algorithm 3.1** Basic STD pooling strategy : Inputs – $n, d$ and $E$

1. Choose a prime number $q$, with $q < n$. Start with the smallest prime, 2.
2. Find the *compression power*, $\Gamma = \min\{\gamma | q^{\gamma+1} \geq n\}$, therefore $\Gamma = \left\lceil \frac{\log n}{\log q} \right\rceil - 1$. Set $k = d\Gamma + 2E + 1$.
3. Check if this choice of $q$ and $k$ satisfy the *guarantee* requirements of identifying $d$ active compounds and correcting $E$ errors, using the inequality, $k \leq q + 1$.
4. If the inequality is satisfied continue to step 5, else choose the next prime in step 1 and repeat steps 2 and 3.
5. Once the smallest prime number $q_{\min}$ and its corresponding compression power $\Gamma_{\max}$ are found, all $q > q_{\min}$ will satisfy the inequality in step 3. Therefore, cycle through the values of $\Gamma$ in $\{1, \ldots, \Gamma_{\max}\}$ to find the corresponding $q$. For each $\Gamma$ find the smallest $q$ that satisfies, $q \geq n^{1/\Gamma+1}$.
6. Calculate the number of tests $(t)$ needed by each $q$ and $k$ pair from, $t = q \times k$.
7. Choose the $q$ and $k$ pair producing the least number of tests.
8. Design the pooling matrix, $M = \text{STD}(n; q; k)$.

A detailed description of the procedure used to construct the $t \times n$, 0-1 matrix $M$ can be found in the original paper (Thierry-Mieg, 2006a) and is reproduced in the Appendix A.1.

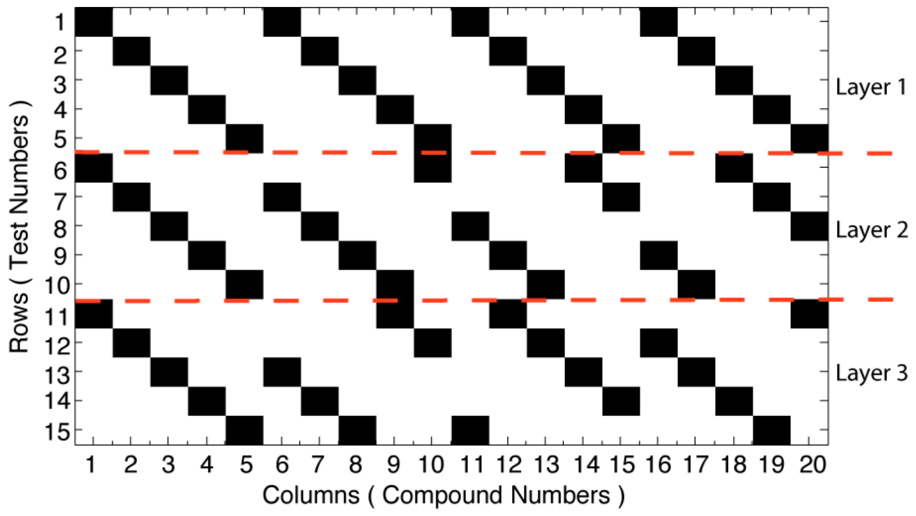

**Figure 3.1** An example of a pooling design for 20 compounds, expecting at most 2 active compounds and 0 errors in testing, requiring a $15 \times 20$ binary matrix, created using STD$(n = 20; q = 5; k = 3)$. A *black* square represents the presence of a compound $j$ (column index) in test $i$ (row index). The dashed line represents the separation of the STD design into its 3 layers ($k$), each of size $5 \times 20$ ($q \times n$). Each compound is present *only once* in each layer of the design. Each compound is present *at most* once ($\Gamma = 1$) with every other compound.

Having designed the pooling scheme, $M$, the decoding algorithm is given below. The pooled assays, $t$ in number, are carried out and the results classified into two states – positive or negative, using a chosen threshold.

1. A compound present in at least $E + 1$ negative tests is tagged inactive.

2. A compound present in at least $E + 1$ positive tests, in which all other compounds have been tagged inactive, is tagged active.

Note that each compound is present in $d\Gamma + 2E + 1$ tests and no two compounds are mixed together more than $\Gamma$ times. A more elaborate decoding algorithm designed to handle the presence of larger-than-designed-for values of $d$ and $E$ is provided in the original paper and its MATLAB implementation is provided in Appendix A.6.

## 3.2 Error rate

The STD-based pooling strategy creates a design for a specified number of errors, $E$, resulting in the addition of $2E$ extra tests to the total of $t$ pooled tests. Assuming random experimental errors, the number of errors present in the result will increase as the number of tests increase. In HTS, there can be instrument, biological, chemical, or human errors that increase (false positive, FP) or decrease (false negative, FN) the measurement from its true value (Malo et al., 2006). These multiple sources of error make it difficult to estimate the total number of errors, $E$, before knowing the number of pooled tests ($t$) that will be used. However, often in HTS we know the overall random error-rate for an assay, which we call $e$. For example, the pooling design for 100 compounds ($n$) expecting 3 active compounds ($d$) and 2 testing errors ($E$) needs 88 tests ($t$) using $STD(100;11;8)$ (Table 2 in Thierry-Mieg (2006a)). If we define the error-rate ($e$) as the percentage errors expected per test then $e = \frac{E}{t} \times 100$. For the example above, this error-rate $e$ is $\sim 2.27\%$. However, when the number of errors ($E$) is changed, keeping everything else the same, the corresponding number of pooled tests ($t$) and hence the error-rate ($e$) change. From Figure 3.2 it can be seen, that the error-rate ($e$) reaches a maximum and then drops off with respect to $E$. The STD construction is efficient for low values of $E$ but requires many more tests ($t$) for higher $E$ values, as seen in Figure 3.2 inset. This nonlinear relationship between $t$ and $E$ indicates that a construction based on the input parameter $E$ alone is not satisfactory. We propose the use of an expected error-rate $e$ as a input parameter instead. To do so we have modified the STD strategy as follows.

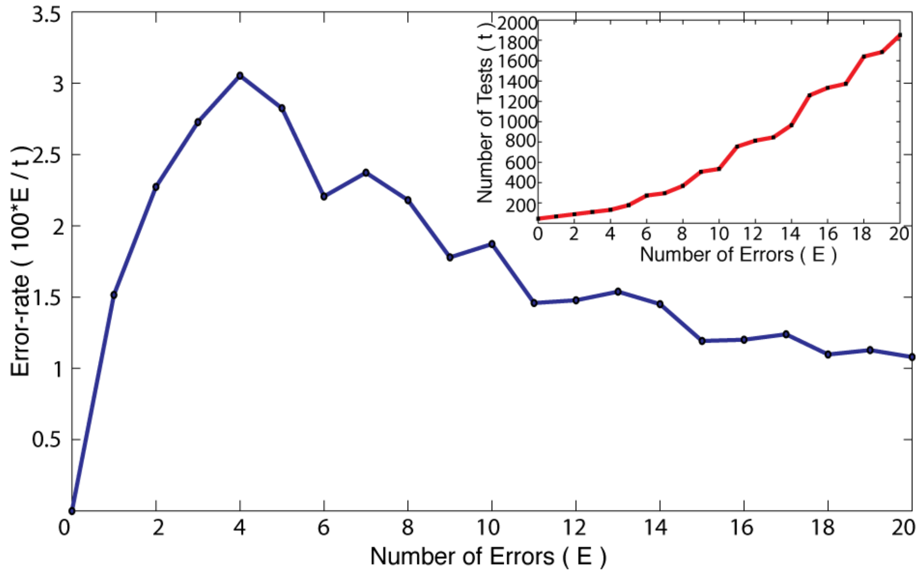

**Figure 3.2** Variation of error-rate ($e = \frac{E}{t} \times 100$) with choice of design parameter $E$ (number of errors) keeping compound library size ($n = 100$) and expected number of positives ($d = 3$) fixed. The inset shows the variation in number of test needed ($t$) with the same parameter $E$, for a $E = 0$ to 20 and $t = 44$ to 1852.

**Algorithm 3.2** Error-rate based STD strategy : Inputs – $n$, $d$ and $e$

Steps 1 and 2 same as (Algorithm 3.1)

3. If $d\Gamma \leq q$, calculate $E_{\max} = \left\lfloor \frac{q-d\Gamma}{2} \right\rfloor$, the maximum number of errors that can be corrected by this choice of $q$, for the given $n$ and $d$.

4. Check if the input error-rate $e$ is achievable using the inequality, $e \leq \frac{E_{\max}}{q(d\Gamma+2E_{\max}+1)} \times 100$. If $e$ is achievable then continue to step 5, else go back to step 1 and try the next $q$.

5. Cycle through all values for $E$ from 0 to $E_{\max}$ to find the minimum $E$ ($E_{\min}$) that satisfies the inequality in step 4. Use $E_{\min}$ to calculate, $k = d\Gamma + 2E_{\min} + 1$ and the number of tests, $t = q \times k$.

6. Similar to Algorithm 3.1, cycle through the values of $\Gamma$ smaller than the $\Gamma_{\max}$ to find the corresponding $q$ and hence $t$.

7. Use the design parameters ($q$ and $k$) that need the minimum number of tests.

8. Construct the pooling design, $M = \text{STD}(n;q;k)$, as usual.

From step 4 we find that there is an upper limit to the error-rate ($e$) for a given library size ($n$) and chosen maximum number of active compounds expected ($d$). This limit is a function only of the design parameter $q$ as shown in Equation 3.1 (elaborated in Appendix A.2).

$$e \leq \frac{\left\lfloor \frac{q-d\Gamma(q,n)}{2} \right\rfloor}{q(d\Gamma(q,n) + 2\left\lfloor \frac{q-d\Gamma(q,n)}{2} \right\rfloor + 1)} \times 100, \qquad \Gamma = \left\lceil \frac{\log n}{\log q} \right\rceil - 1 \tag{3.1}$$

For the simplest case of $d = 0$, the expression in equation 3.1 simplifies to $e \leq \frac{q-1}{2q^2} \times 100$, for all the odd prime numbers and $e \leq 16.67\%$ for $q = 2$. Thus any assay with an error rate greater than 16.67% cannot *guarantee* accurate results when screened using an STD-based scheme. In more realistic cases, the error-rates corrected by STD-based schemes are much smaller than this limiting case of $d = 0$ because $d$ and $\Gamma$ would have substantial values. This finding implies that no matter how large the specified number of errors ($E$) in the original STD-based pooling strategy, the corresponding number of tests needed ($t$) would adjust itself to keep the error-rate ($e$) corrected by the design low. An example of the effect of Equation 3.1 can be seen in Figure 3.3 for a library of 100 compounds ($n = 100$) and various values of expected active compounds ($d$).

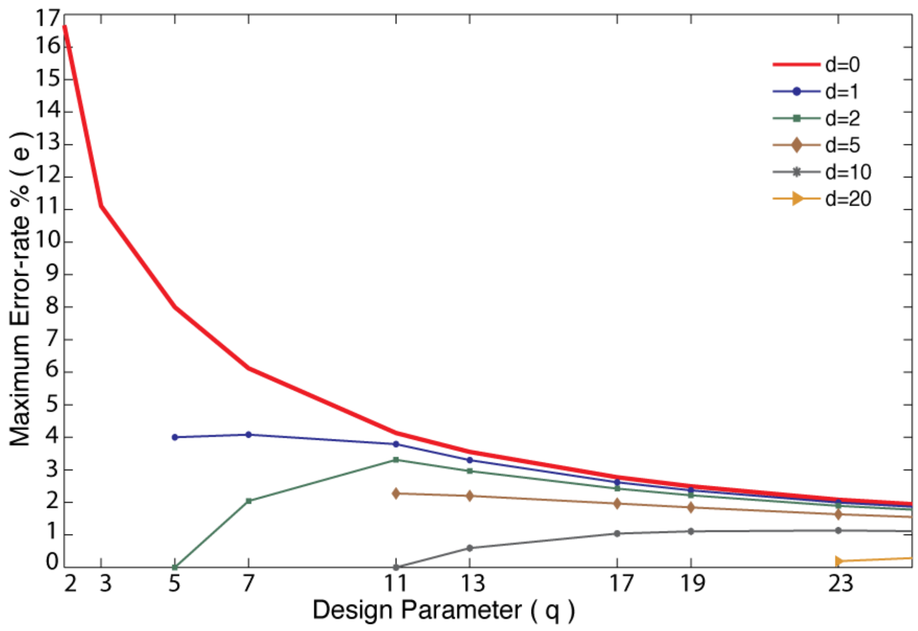

**Figure 3.3** The limits on assay error-rate ($e$) that can be handled by STD for a sample choice of library size ($n = 100$) and various choices of expected active compounds ($d$). The trend suggests that the ability to correct for experimental assay error-rates decreases as $q$ increases.

## 3.3 Mixing Constraint

To our knowledge, no pooling designs address the physical limitation that only a finite number of compounds can be mixed in each well. This limitation arises in drug screening for the following three reasons.

1. Each compound must be present at a sufficiently high concentration so as to be detectable by the assay within a physiologically reasonable range.

2. The total ionic strength of the test solution must be low enough to prevent precipitation of compounds or possible changes to the biological target.

3. The assay must be reasonably simple to physically construct.

From these constraints we can conclude that the total number of drugs that can be practically included in any experimental test well is relatively small. For example, for a relevant screening concentration of $\sim 10\ \mu$M, we can assume that each well can have $\sim 10$ compounds mixed in it. If more drugs are mixed, the cost of creating the assay increases and the concentration of the well mixture may become too high, resulting in inaccurate screening results. This limit can be specified as an input parameter to the pooling design, based on the high-throughput assay being implemented.

Here we extend the STD-based pooling strategy to introduce an explicit mixing constraint. Let $m$ be the mixing constraint, defined as the maximum number of compounds mixed in a well of the pooling scheme. Normally the STD design mixes at most $\left\lceil \frac{n}{q} \right\rceil$ compounds in each well Thierry-Mieg (2006a). However, this feature can break down when the input values of $n$, $d$ and $e$ (hence $E_{\min}$) are such that, for a certain choice of $q$ and $k$ $(= d\Gamma + 2E_{\min} + 1)$, $k = q + 1$ and $\left\lfloor \frac{n-1}{q^{\Gamma}} \right\rfloor < q - 1$, resulting in an unusually large number of compounds mixed in some tests and some tests with no compounds in them. The original STD construction can be easily modified by removing tests with no compounds in them. The details of the correction are provided in Appendix A.3. Having made this correction, we can implement the mixing constraint as follows (see Appendix A.4).

**Algorithm 3.3** Mixing constraint implementation : Inputs – $n$, $d$, $e$ and $m$
Steps 1 through 5 are the same as (Algorithm 3.2).

6. Using the $q$ and $k$ values obtained so far in the algorithm to choose one of the following options.
   a. If $k < q + 1$ and $\left\lceil \frac{n}{q} \right\rceil \leq m$, use Construction 3.1 (Appendix 3) requiring $t = qk$ tests.

   b. If $k = q + 1$, $n = q^{\Gamma+1} - 1$ and $q^{\Gamma} \leq m$, use Construction 3.1 requiring $t = q(q+1)$ tests.

   c. If $k = q + 1$, $\left\lfloor \frac{n-1}{q^{\Gamma}} \right\rfloor < q - 1$ and $q^{\Gamma} \leq m$, use Construction 3.2 (Appendix 3) requiring $t = q^2 + \left\lfloor \frac{n-1}{q^{\Gamma}} \right\rfloor + 1$ tests.
7. Similar to the previous algorithms, cycle through the values of $\Gamma$ smaller than $\Gamma_{\max}$ to find the corresponding $q$ and hence $t$.

8 Use the design parameters ($q$ and $k$) that require the minimum number of tests.

## 3.4 Block pooling

Limiting the number of compounds that can be mixed in a pooling strategy reduces the savings in tests that could otherwise have been obtained. For example, without the mixing constraint, screening a library of $10,000$ compounds with 3 expected actives and no error requires only 110 tests, using $\mathrm{STD}(10000;11;10)$, each mixing 910 (=$\lceil \frac{10000}{11} \rceil$) compounds in them. However, a mixing constraint of 10 compounds per test increases the required tests to 4036, using $\mathrm{STD}(10000;1009;4)$. Now, consider the STD-based strategy for 400 compounds while expecting only 1 active compound in a similar error-free setting and a mixing constraint of 10 compounds per well. The number of tests needed in this case is 82, using the STD$(400;41;2)$ design. If we now divide the original $10,000$ compound library into 25 blocks of 400 compounds each and used the STD$(400;41;2)$ design repeated on those 25 blocks, the total number of tests is only $25 \times 82 = 2050$, almost half the original requirement! The trade-off here is that the block assay design guarantees the detection of only 1 active compound out of every 400 compounds tested. However, given that we expected only 3 active compounds in a library of $10,000$ compounds, this implies that there is $\sim 99.5\%$ chance of finding, at best, 1 active compound among the 400 randomly selected compounds for a block (using the hypergeometric distribution). This block decomposition algorithm is inspired by a form of pooling used in NMR screening (Mercier and Powers, 2005). This example demonstrates that we can make an informed choice about a block size that can, not only, help reduce the number of tests needed while enforcing the mixing constraint but the smaller block size also implies that a better error-rate can be handled by the design (as seen in Figure 3.3).

For any block of $n_B$ compounds from the library of $n$ compounds, the number of blocks needed to cover the whole library is $B = \left\lceil \frac{n}{n_B} \right\rceil$. The creation of a STD for this block of $n_B$ compounds, using the error-rate $e$ and mixing constraint $m$ (assumed to stay the same for all blocks) requires the specification of the maximum number of active compounds expected for the given $n_B$, called $d_B$. The choice of $d_B$ is from a hypergeometric distribution such that, with at least a probability of $p_B$ (specified at the outset) the block of $n_B$ compounds contains at most $d_B$ active compounds. Using the hypergeometric distribution, the following inequality (elaborated in Appendix A.5) can be solved for $d_B$.

$$\sum_{i=0}^{d_B} \frac{\binom{d}{i}\binom{n-d}{n_B-i}}{\binom{n}{n_B}} \geq p_B \tag{3.2}$$

Based on this choice of $d_B$, a STD-based pooling design of size $t_B \times n_B$ can be generated using Algorithm 3.3. The number of tests needed for the whole library would be $B \times t_B$.

## 3.5 poolHiTS

We now define poolHiTS as an STD-based pooling strategy which takes in as input the compound library size ($n$), maximum number of active compounds expected ($d$), maximum error-rate expected ($e$), mixing constraint ($m$), and design confidence metric ($p_b$). poolHiTS$(n,d,e,m,p_b)$ produces a $t \times n$ mixing matrix $M$, which guarantees the success of the pooling scheme for the given input parameters. The algorithm for poolHiTS, which includes error-rate specification, mixing constraints, and optimal block size selection, is as follows. The MATLAB implementation of the poolHiTS algorithm is provided (see Appendix A.7.1).

**Algorithm 3.4** poolHiTS algorithm : Inputs – $n,d,e,m$ and $p_B$

1. Choose a value of $d_B$ in $\{1,\ldots,d-1\}$.
2. Find the set of $n_B$ that satisfy the inequality in Equation 3.2.
3. For each $n_B$ in this set use Algorithm 3.3 to evaluate STD$(n_B;q_B;k_B)$, if it exists, for the given $e$ and $m$. Calculate the total number of tests needed from $B \times t_B, \quad B = \left\lceil \frac{n}{n_B} \right\rceil$.
4. Choose the next value of $d_B$ and repeat steps 2 and 3.
5. After testing all values of $d_B$ in $\{1,\ldots,d-1\}$ and the corresponding $n_B$, select the $d_B,n_B$ pair that require the least number of tests. The whole library design (which corresponds to $n_B = n$, $d_B = d$ and $B = 1$) should also be included while making this choice.
6. Design the pooling matrix, $M = \text{STD}(n;q;k)$, for the choice of $q$ and $k$.

A typical example of a result of applying Algorithm 3.4 to the pooling design problem is shown in Table 3.1. Consider a case of a $10,000$ compound library ($n$), where we expect upto 3 active compounds ($d$) with 1% assay error-rate ($e$), a limit on mixing not more than 10 compounds in a test ($m$), and at least a 99% chance of finding the active compounds ($p_B$). From this problem specification we have two observations. First, a single whole library design is not possible because the mixing constraint permits only 10 compounds per assay, thus we must use a repeated block design. Second, we see that there is an optimal number of blocks ($B$) that requires the least number of tests and has a block size ($n_B$) between the two extremes ($n_B = n$ and $n_B = 1$). The best design chosen by poolHiTS$(10000,3,1,10)$ is to implement a block pooling scheme for $n_B = 110, d_B = 1, e = 1$ and $m = 10$, using STD$(10000;11;4)$, and repeat this design 91 times over to cover the whole library. The reason for choosing $n_B = 110$ rather than $n_B = 130$ (see Table 3.1), while both provide equal compression, is that the former provides a better actual error-correcting rate of 2.27%. It is useful to note that this design is capable of screening a library of $91 \times 110 = 10,010$ compounds vs. the specified $10,000$ compounds. However, because of the extra error-correction, we can take advantage of this design by ignoring the extra compounds or using some form of control compound in their place. The MATLAB implementation of this example is provided (see Appendix A.8).

| Number of blocks ($B$) | Number of compounds per block ($n_B$) | Number of active compounds expected per block ($d_B$) | Number of tests needed per block ($t_B$) | Error-rate handled ($e$) | Mixing constraint provided ($m$) | Total number of tests needed ($t$) |
|---|---|---|---|---|---|---|
| 1 | 10000 | 3 | – | – | – | – |
| 25 | 400 | 1 | 492 | 1.02% | 10 | 12300 |
| 50 | 200 | 1 | 92 | 1.09% | 9 | 4600 |
| **77** | **130** | **1** | **52** | **1.92%** | **10** | **4004** |
| **91** | **110** | **1** | **44** | **2.27%** | **10** | **4004** |
| 100 | 100 | 1 | 44 | 2.27% | 10 | 4400 |
| 500 | 20 | 1 | 20 | 5% | 4 | 10000 |

**Table 3.1** Block Design Example

Sample case of $n = 10,000, d = 3, e = 1\%$ and $m = 10$ showing the efficiency of selected block design parameters. The choice of error-rate ($e = 1\%$) does not permit a full library design (first row), thus it is left empty.

## 3.6 Conclusion

In this chapter I presented a pooling strategy called poolHiTS which implements tailored modifications and enhancements that make the shifted transversal design (STD) algorithm appropriate for drug discovery. First, I demonstrated how to switch from specifying the number of errors ($E$) for a STD-based strategy to an error rate ($e$), which is the percentage of errors expected in tests. Then, I showed that there is an upper limit to the error-rate that can be handled by this STD-based algorithm and this error limit (Equation 3.1) strongly constrains pooling designs to less noisy screening assays. I implemented error rate as an input experimental parameter via Algorithm 3.2.

Second, I introduced and implemented explicit mixing constraints to make pooling significantly more relevant to HTS assays via Algorithm 3.3. I also provided a necessary correction to the STD construction algorithm pertinent to the mixing constraint.

Third, I introduced the concept of repeated block designs that retains the efficiency of pooling strategies in the face of the mixing constraint and error-rate limitations. I showed that by using this block design, I am able to simplify the assay construction, increase the error tolerance and decrease the assay size.

The combination of these features produce the poolHiTS strategy described in Algorithm 3.4. The MATLAB implementations of the poolHiTS algorithm and an example of its use are provided in Appendix 3. poolHiTS provides a promising route to both reducing the cost and increasing the accuracy of high throughput drug screening. Although poolHiTS primarily focuses on drug screening, these same design methods can apply equally well to other screening environments where the number of perturbations to a system is finite or small (mixing constraint) and the error rate of the assay is approximately known.

# Chapter 4
# QUAPO – Quantitative analysis of pooling

This chapter describes two different approaches to the analysis of the results of a pooling design experiment. These approaches are then tested experimentally in a high-throughput drug screen performed using the pooling design strategy described in Chapter 3.

## 4.1 Interpool

The decoding procedure described in Chapter 3 uses the number of testing errors expected ($E$) to identify the active compounds correctly. This can be problematic for real experiments where the a good estimate of the error rate is hard to come by and using an estimated error rate ($E$) for decoding would reduce the scope of the pooling design to its design parameters which, as described in Chapter 3, are provided for worst-case scenarios.

A decoding procedure called Interpool that bypasses the problem of using pooling design parameters in decoding was proposed by Thierry-Mieg and Bailly (2008). This approach consisted of two parts. First, they provided a comprehensive mathematical treatment for the decoding problem, which is then used to solve it by finding a set of consistent interpretations that match the pooled observations. The interpretations would consist of positive, negative, and ambiguous calls for each compound's activity. Second, using the same mathematical framework the authors provided a tool to simulate the average case performance of the pooling design. This tool would allow the experimenter to find the pooling design most suitable for ranges of the two experimental parameters – hit rate and error rate – by running various simulations of the performance of several pooling designs under these conditions. Although computationally expensive when testing with large ranges of parameters, this method would provide some degree of control over the optimal choice of a pooling design and free the pooling strategy from the rigid confines of the worst-case pooling design methods described in Chapter 3.

### 4.1.1 Interpool decoder

The Interpool decoder starts with a set of observations from a pooling design experiment. Using a cutoff value these observations are transformed into two categories, positive and negative. In the case of high-throughput drug screening, the observations are the intensities from a set of pooled wells that have been categorized as positive or negative using a cutoff intensity value. With this as a starting point, the decoder applies the following rules to arrive at a canonical interpretation of the observations. Paraphrasing from Thierry-Mieg and Bailly (2008),

1. Any compound present in a negative well is tagged as inactive.
2. The remaining compounds are tagged active if they are the only compounds present in a positive well where all the other compounds are inactive; otherwise they are tagged ambiguous.
3. If every positive well contains at least one active or ambiguous compound, the interpretation is consistent.

For a consistent interpretation there is no further decoding to be done. The ambiguous compounds have to be retested to tag them as active or inactive. For inconsistent interpretations, the notion of conflicting variables and conflicting pools is applied. Again, translating the conditions to the language of high-throughput drug screening,

1. Any positive well is non-conflicting, if it contains at least one active or ambiguous compound, otherwise it is conflicting.
2. A compound is conflicting if it appears in at least one conflicting positive well.
3. A negative well is non-conflicting if it does not contain any conflicting compounds. Otherwise, it is conflicting.

Using the above categorization it is now possible to look for consistent interpretations of the observations, by switching wells from positive to negative and vice versa. However, this kind of switching has a cost associated with moving away from the actual observations.

The simplest case is to have an equal cost for switching a positive well to a negative and a negative well to a positive, say a cost of 1. Now, given an inconsistent interpretation, it is possible to switch a certain number of wells such that we get a consistent interpretation. The number of switches, in this case, would represent the cost of this interpretation. The goal of the Interpool decoder is to find the set of consistent interpretations. There can be more than one consistent interpretation when switching is allowed, that come at minimum cost. This set of consistent interpretations at minimum cost can then be used to arrive at a final consistent interpretation using any rule – union, intersection, majority, etc. The main goal is to find this minimum-cost set of consistent interpretations.

Equivalent to switching well states to find a consistent interpretation, sets of conflicting compounds can be switched from inactive to active, or vice versa, and the corresponding switches to the well states that make the interpretation consistent can be found and the cost of this switch calculated. It is easier to implement this approach to find the minimum-cost set of consistent interpretations. To identify this set, the Interpool decoder uses a branch-and-bound strategy; see Brusco and Stahl (2005) for details.

As illustrated in Figure 4.1, the Interpool decoder starts with the canonical interpretation from the observations and identifies the conflicting compounds and wells. Using this information it calculates, individually for each conflicting compound, the cost of switching and its associated positive and negative conflicting well switches. These form the *units* that start off the branch-and-bound strategy that helps identify the optimal solution set (or sets) of conflicting compounds that, when switched, would provide a consistent interpretation at minimum cost.

The following recursive algorithm implements the search for this set (or sets) of compounds:

**Algorithm 4.1** Interpool decoder : Inputs – *units, pervious, best*

1. Start with an empty solution set that serves as both *previous* and *best* solutions.

2. Sort the *units* by decreasing score and then by increasing size.

3. For each entry in *units* (in order of decreasing score), apply a pruning criterion (described below) to check if adding more compounds to the set of solution set would improve its score. If so, continue by going to step 4. Else, quit the algorithm.

4. Remove the entry from *units* and add it to the *previous* solution to get the current solution.

5. Use the *current* solution to update the other entries in *units* and rescore and resize them.

6. Compare the score of *current* solution against the *best* solution and update if required.

7. Go to step 2.

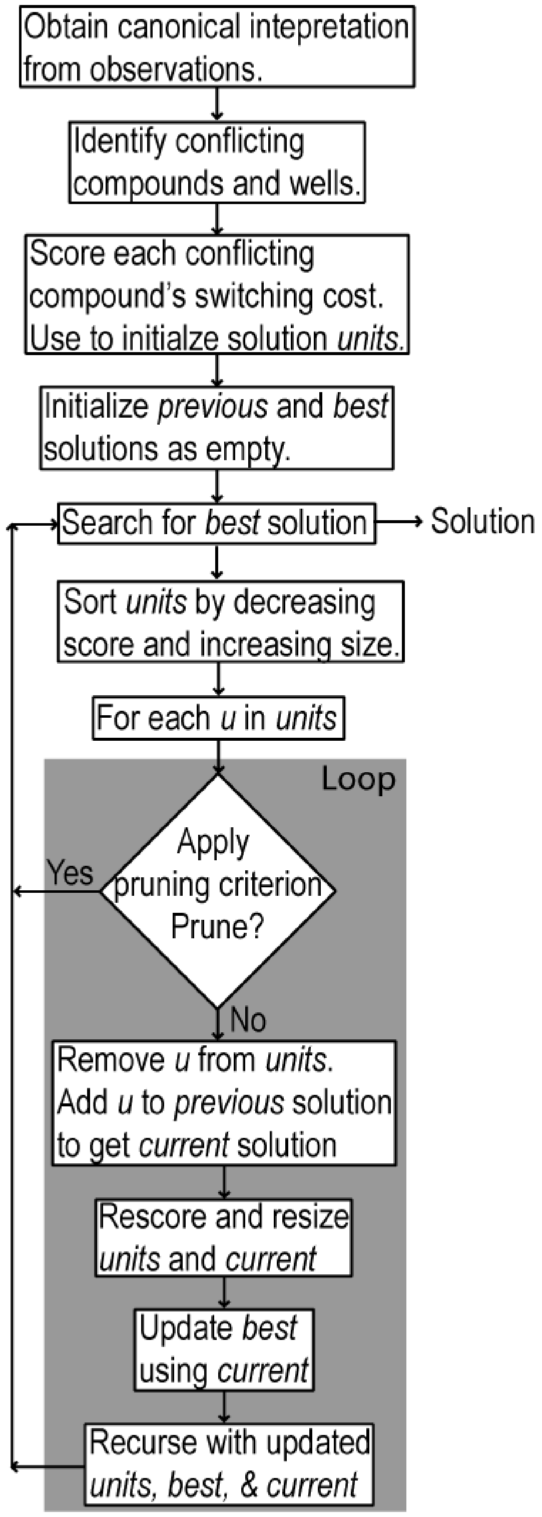
Obtain canonical intepretation from observations.
Identify conflicting compounds and wells.
Score each conflicting compound's switching cost. Use to initialze solution units.
Initialize previous and best solutions as empty.
Search for best solution
Solution
Sort units by decreasing score and increasing size.
For each u in units
Loop
Apply pruning criterion Prune?
Yes
No
Remove u from units. Add u to previous solution to get current solution
Rescore and resize units and current
Update best using current
Recurse with updated units, best, & current

**Figure 4.1** A flowchart of the Interpool decoder.

The pruning criteria allow the algorithm to limit the search space for the set(s) of conflicting compounds that can be resolved to provide a consistent interpretation to the observations, at minimum cost. The criteria provide a way to estimate in advance if adding any more conflicting compound(s) to a set under consideration would improve the score, and thereby minimize the cost. If not, the algorithm stops the search and returns the optimal solution(s). The details for the pruning criteria and their efficiency at solving the search problem can be found in Thierry-Mieg and Bailly (2008).

### 4.1.2 A note about the cost of switching

In the description above, the switching cost was shown to be equal for switching a negative well to positive (correcting a false negative) and vice versa (correcting a false positive). However, depending on the assay, the sensitivity for either form of error could vary. Further, the cost of switching a well result can also depend on its intensity measurement value. For example, the cost of switching a positive well that had a strong quantitative result could be higher than the cost of switching a positive well that had a relatively weaker quantitative result. Similarly for negative wells, there can be gradations in their negativeness. This would allow for more sensitive decoding. The Interpool decoder allows this gradation to be implemented with, usually, four levels – strong and weak for positive wells, and faint and none for negative wells. The score of switching each of these states can be chosen beforehand. These gradations can be extended to more levels as required.

## 4.2 QUAPO – Quantitative Analysis of Pooling

Despite the utility of the Interpool in solving the decoding problem from first principles it still gives us only a qualitative answer. Interpool provides a compound list categorized as active, ambiguous, and inactive, along with a measure of confidence in that categorization (via the cost metric associated with the active and ambiguous calls). However, it does not utilize all the quantitative data present in a high-throughput screen. It does not allow us to rank order the active compounds, which would be beneficial while designing the follow-up dose-response experiments for them. Typically, we are interested in compounds that show high activity at low concentrations, as these translate into better drugs. In order to obtain quantitative information from pooled screens we propose a new approach to decoding, one that is based on a biochemical model of pooling.

### 4.2.1 Biochemical model of pooling

A biochemical model that allows quantitative information from pooled screens is described in the context of a competitive binding screen using a scintillation proximity assay (SPA) type reporting technology (Wu and Liu, 2005). This model can be applied, almost identically, to most other types of assays and reporting technologies. Consider an SPA-based screen where a fluorescent bead that emits light when stimulated is used to anchor the receptors that form the target in the assay. As shown in the first panel of the Figure 4.2, a radio-labeled ligand bound to the receptor stimulates a signal which can be detected. However, as shown in the second panel of the figure, when there are compounds that compete with the labeled ligand in binding to the receptor, the presence of strong binding compounds decreases the signal, a process called inhibition.

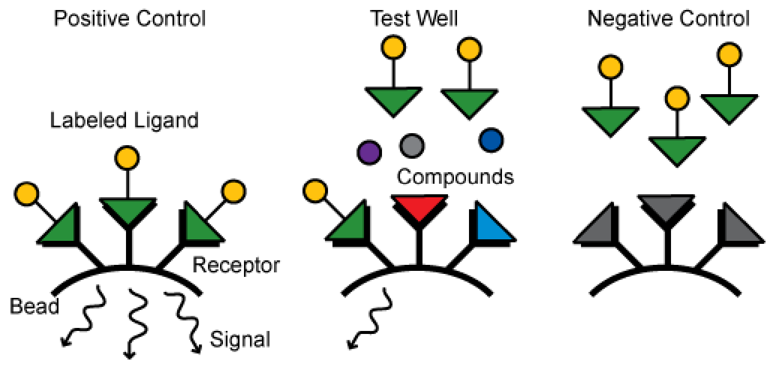

**Figure 4.2** Example of a scintillation proximity assay (SPA) to detect competitive binding molecules. The left panel shows the positive control well containing only radio-labeled ligands bound to the receptor, which is attached to a fluorescent bead. The middle panel shows inhibition of signal from the bead in response to chemical compounds displacing the labeled ligand bound to the receptor. The right panel shows the negative control well containing a known strong inhibitor displaces the labeled ligand.

This process can be modeled mechanistically as follows. The receptor (R) is the biological target and the compound (D) is tested for competitive binding against a labeled ligand (L). In the positive control well, only the ligand and receptor combine to form a complex (C).

$$\mathrm{R} + \mathrm{L} \leftrightarrow \mathrm{C}$$

At equilibrium, the dissociation constant of this reaction is,

$$K_d = \frac{[L][R]}{[C]} \tag{4.1}$$

The intensity measured ($\mathrm{I_{PC}}$) from this well, after removing nonspecific binding measured in the negative control well ($\mathrm{I_{NC}}$), corresponds to the concentration of the ligand-receptor complex,

$$\mathrm{I_{PC}} - \mathrm{I_{NC}} \propto [C] \tag{4.2}$$

Combining Eqs. 4.1 and 4.2 and accounting for a total balance on the number of receptors ($R_{tot} = [R] + [C]$), we find,

$$\frac{\mathrm{I_{PC}} - \mathrm{I_{NC}}}{R_{tot}} \propto \frac{[L]}{K_d + [L]} \tag{4.3}$$

Here, $R_{tot}$ is the total number of receptors present in the well (bound and free) and $[L]$ is the ligand concentration at equilibrium, which can be assumed to be taken in excess and hence the same as the starting ligand concentration. For pooling designs we mix multiple compounds in a well, resulting in the following reactions,

$$\begin{gathered} \mathrm{R} + \mathrm{L} \leftrightarrow \mathrm{C} \\ \mathrm{R} + \mathrm{D_1} \leftrightarrow \mathrm{C_1} \\ \vdots \\ \mathrm{R} + \mathrm{D_r} \leftrightarrow \mathrm{C_r} \end{gathered}$$

The intensity measurement from a pooled well ($\mathrm{I_{TEST}}$) will be reduced due to competition with the unlabeled compounds for the receptor.

$$\mathrm{I_{TEST}} - \mathrm{I_{NC}} \propto [LR]_{eq} \tag{4.4}$$

If each compound is assumed to act independently of others (no chemical interactions) in the well and that the concentrations of the labeled ligand and the compounds being tested are not depleted by binding to the receptor, we arrive at the following expression for the intensity ($\mathrm{I_{TEST}}$).

$$\frac{\mathrm{I_{TEST}} - \mathrm{I_{NC}}}{R_{tot}} \propto \frac{[L]/K_d}{1 + [L]/K_d + [D_1]/K_{d1} + \ldots + [D_r]/K_{dr}} \tag{4.5}$$

Using Equations 4.4 and 4.5 we can arrive at an expression relating the percentage inhibition observed in a well to the activity of the compounds placed in it.

$$\%\text{Inhibition} = \%I = 100 \times (1 - \frac{\text{I}_{\text{TEST}} - \text{I}_{\text{NC}}}{\text{I}_{\text{PC}} - \text{I}_{\text{NC}}}) = 100 \times \frac{[D_1]/K_{d1} + \ldots + [D_r]/K_{dr}}{1 + [L]/K_d + [D_1]/K_{d1} + \ldots + [D_r]/K_{dr}} \tag{4.6}$$

Here, $[L]$, $K_d$ and $[D_i]$s are known for the assay.

We can simplify Equation 4.6 by assuming that all the compounds in a well have the same concentration ($[D]$) and use the association constant $K_a$ to produce a linear sum. For each pooled well in the screen, the following equation would hold true. We now define a quantity $T$, which represents the summation of association constant ($K_a$) of the compounds present in each well, in terms of the experimental measurement.

$$T = \frac{1}{[D]} \cdot \frac{1 + [L]/K_d}{\frac{100}{\%I} - 1} = \frac{1}{K_{d1}} + \ldots + \frac{1}{K_{dr}} = K_{a1} + \ldots + K_{ar} \tag{4.7}$$

This expression can be written for each of the pooled wells in the screen to obtain the simplified linear algebra problem.

$$\overset{\mathbf{T}}{\begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_t \end{pmatrix}_{t\times 1}} \propto \overset{\mathbf{M}}{\begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 1 \\ 0 & 1 & & 0 & & 0 \\ & \vdots & & \vdots & & \vdots \\ 1 & 0 & \ldots & 1 & \ldots & 0 \end{bmatrix}_{t\times n}} \times \overset{\mathbf{K}}{\begin{pmatrix} K_{A1} \\ K_{A2} \\ \vdots \\ K_{Ai} \\ \vdots \\ K_{An} \end{pmatrix}_{n\times 1}}$$

**Figure 4.3** Linear relationship between compound activities and quantities measure from a pooled screen.

As shown in Figure 4.2.3, $M$ is a $t \times n$ binary matrix (where $t$ is the number of pooled tests and $n$ the total number of compounds being screened) that represents the pooling design used to mix compounds and $T$, a $t \times 1$ vector, represents the quantity obtained from experimental measurements. It remains to solve for activities of the compounds $K$, a $n \times 1$ vector, given $M$ and $T$. Moreover, due to the savings in tests achieved from pooling, $t << n$, making this linear system of equations *underdetermined*. The measurement noise, inherent in $T$, makes the process of solving for $K$ harder. However, using insights from a new field of research called *Compressive sensing* (Candès et al., 2006; Donoho, 2006), it has been shown that a good choice of pooling scheme ($M$) can guarantee the correct solution for $K$, even in the presence of measurement noise. The details of the design of such pooling schemes and the subsequent decoding procedure are discussed in the following section.

### 4.2.2 Compressive Sensing

Compressive Sensing is a relatively new area of signal processing that deals with the problem of recovering a compressible signal from noisy measurements.

### 4.2.3 Compressive sensing guarantee

Suppose we wish to recover a vector $x \in \mathbb{R}^n$ from incomplete and noisy measurements $y = Mx + e$; $M$ is a $m \times n$ matrix with far fewer rows than columns ($m << n$) and $e$ is an error term. It is possible to recover $x$ accurately based on the data $y$ by solving the linear program,

$$\hat{x} = \min_z ||z||_1 \qquad \text{subject} \quad \text{to} \qquad ||Mz - y||_2 \leq \varepsilon \tag{4.8}$$

where $\varepsilon$ is the size of the error term $e$. If $M$ obeys a Restricted Isometry property (RIP) and $x$ is sufficiently sparse, then the solution is within the noise level.

$$||\hat{x} - x||_2 \leq C \cdot \varepsilon$$

See Candès et al. (2006) for details.

In the context of QUAPO, this implies that when there are a small number of compounds that are *active*, the vector of compound activities $K$ in Figure is reasonably sparse, i.e., has very few large entries. Therefore, even with much fewer and noisy measurements, we can successfully recover the activities of the compounds by solving the linear program (LP), provided we choose a good mixing scheme.

### 4.2.4 Pooling scheme construction

Accurate recovery of the compound activities is incident upon the choice of a good pooling design, one that obeys the Restricted Isometry property (RIP). The RIP property requires that the pooling scheme $M$ preserve the length of $k$-sparse vectors. This implies, for some $k$-sparse vector $v$ and $\delta > 0$,

$$1-\delta \leq \frac{\|Mv\|_2}{\|v\|_2} \leq 1+\delta \tag{4.9}$$

However, to obtain a stable solution to Equation 4.8, it is sufficient that we use a pooling scheme $M$ that satisfies Eq. 4.9 for an arbitrary $3k$-sparse vector $v$. So, if there are only a small number of active compounds ($k << n$) in the library being screened, then using a pooling scheme that satisfies Equation 4.9 for $3k$-sparse signals will ensure that the recovered or decoded activities ($\hat{x}$) are within the noise level of the measurements. Fortunately, there are good deterministic constructions of pooling schemes that satisfy the RIP property. One such construction was provided by R. DeVore using finite fields (DeVore, 2007). To this construction is added another constraint arising from experimental implementation.

Pooling in a drug screening assay needs to satisfy the following three constraints:

1. Each compound must be present in sufficiently high concentration to be detectable.
2. The total concentration of the test solution must be low enough to prevent precipitation of compounds or possible changes to the biological target.
3. The assay must be reasonably simple to construct.

These constraints imply that the total number of compounds that can be practically pooled in any assay well is relatively limited. Assuming a reasonable physiological concentration of $\sim 10\mu$M, we can assume that each assay well can have at most $\sim 10$ compounds pooled in it. If more compounds are mixed, then the cost of creating the assay increases and the total concentration of materials in the well mixture becomes too high, resulting in inaccurate screening results. Thus, a pooling constraint must be placed on the design, limiting the total number of compounds in each well to a preselected number, $r$ (the maximum number of 1's in each row of $M$).

Therefore, given a compound library size $n$, the maximum number of active compounds expected $k$ and pool size limit $r$, a $t \times n$ binary matrix $M$ can be constructed and implemented that guarantees the successful decoding of the compound activities from noisy measurements. The details of the construction are provided in Appendix C.1. The linear program (LP) is solved using a MATLAB package called $\ell_1$-magic (Candès and Romberg, 2005).

## 4.3 Pooled high-throughput screen

In collaboration with the laboratory of Dr. R. Neubig (University of Michigan), a small proof-of-concept screen was run to test the pooling design strategy described in Chapter 3 along with the decoding schemes described.

The goal of the original study was to identify small molecule inhibitors of the RGS4-G$\alpha_o$ protein-protein interaction and was undertaken by Blazer et al. (2010). Regulators of G protein Signaling (RGS) are proteins that negatively regulate G protein-coupled receptor signaling pathways, which are important to many physiological processes (Berman and Gilman, 1998). Therefore, RGS proteins are an important target in pharmacological research (Neubig, 2002; Blazer and Neubig, 2008). In particular, the RGS4 protein is of interest because it inhibits several pain modulating receptors (Garnier et al., 2003; Traynor and Neubig, 2005). Hence, small molecules that inhibit binding of RGS4 to the G$\alpha_o$ protein were sought. Further, those small molecules whose mechanism of action did not involve reactions with the cysteine residues on the surface of the RGS protein were of interest. The cysteine-reacting mechanism makes the small molecule less desirable as a potential lead compound (Blazer et al., 2010). Therefore, cysteine-less versions of the RGS protein (RGS4C) were used for the assay.

A time-resolved fluorescence resonance energy transfer (TR-FRET) mechanism was used to monitor the binding of RGS4C and G$\alpha_o$. Fluorescence resonance energy transfer (FRET) is a well-known technology that allows the measurement of binding between proteins (Selvin, 2000). Each binding partner is tagged with a fluorophore. When the two proteins are bound and the donor fluorophore is excited by incident light, the proximity allows energy transfer to the acceptor fluorophore. This increase in the acceptor fluorophores emission intensity confirms the binding between the proteins. TR-FRET helps increase the signal-to-noise ratio by reducing background fluorescence at the time of signal measurement (Leifert et al., 2006). In the assay, the RGS4C was labeled with AlexaFluor-488 while the G$\alpha_o$ was labeled with LanthaScreen™Tb thiol. As shown in Figure 4.4 (adapted from Blazer et al. (2010)), Tb-G$\alpha_o$ was excited at 340 nm and the resulting fluorescence emission at 490 nm for Tb-G$_\alpha$o and 530 nm for RGS4C-AF488 was measured after a delay of 100ns. The ratio of emission 520nm/490nm was used to detect binding between RGS4C and G$\alpha_o$. A strong inhibition of this signal implies that a small molecule is successfully inhibiting RGS4C and G$\alpha_o$ binding.

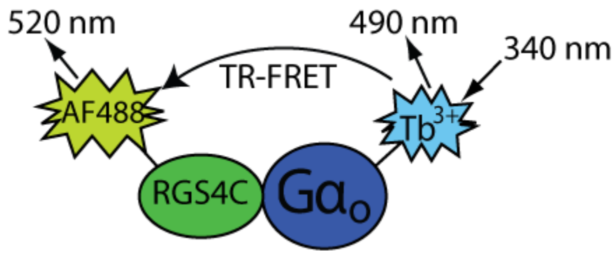

**Figure 4.4** An illustration of the RGS4C-G$_\alpha$o FRET-based assay

The assay and detection mechanism shown in Figure 4.4 can be easily adapted to the biochemical model of pooling described in Section 4.2.1 and Figure 4.2. The positive control would contain only RGS4C and G$\alpha_o$. The test well would contain RGS4C and G$\alpha_o$ along with a mixture of chemical compounds (multiplex). The negative control would contain RGS4C and G$\alpha_o$ along with a strong inhibitor of the FRET signal. Using the intensity measurements from these wells a %Inhibition (%I) of signal can be calculated which can be used in Equation 4.7 to arrive at the quantitative pooled measurements required by QUAPO. For simplicity, an approximation of Equation 4.7, shown in Equation 4.10, was used in the QUAPO analysis. This allows the calculation, for each pooled well, of an approximate quantitative value (LHS of Equation 4.10) of the binding strength ($K_a$) of the compounds pooled together ($\{1,\ldots,r\}$), which allows the transformation of the decoding problem to a linear system of equations, shown in Figure 4.2.3. Although this approximation does not provide exact binding coefficients, it allows the active compounds identified in the screen to be rank ordered by binding strength.

$$\frac{\%I}{100-\%I} \propto K_{a1}+\ldots+K_{ar} \tag{4.10}$$

2000 compounds from the MicroSource Spectrum 2000 collection were screened at the University of Michigan Center for Chemical Genomics in the assay described above to identify small molecules that inhibit the binding of RGS4C with G$\alpha_o$. From these data, a single plate that contained 3 hits (at a cutoff of 3 standard-deviations (SD) from the mean percent inhibition measurements of the negative control and compound wells taken together) was used to run the pooled screen along with a control screen where the compounds were not pooled.

For the sake of symmetry of the pooling design, 316 compounds from this plate were used in the proof-of-concept pooling experiment (320 compounds are available on a 384 well-plate; the remaining wells are used for positive and negative controls). These 316 compounds were run in singleplex (each compound tested individually) and in multiplex. The multiplex screen used the pooling design shown in Figure 4.5. This design corresponds to the shifted transversal design that tests 316 compounds in pools of 4 with each compounds tested 4 times in total. To test each of the 316 compounds once in pools of 4 requires 79 pools that together represent a layer of the design in Figure 4.5. Testing each compound 4 times in total required 79×4=316 pools. This symmetry in the design (4 compounds per pool and 4 tests per compound) determined the choice of testing 316 out of the 320 compounds on the plate. The use of 316 pools to test 316 compounds meant that the multiplex did not represent any saving in tests when compared to the singleplex. A mosquito robot (TTP LabTech, Melbourn UK) was used to perform the singleplex and pooling for the multiplex.

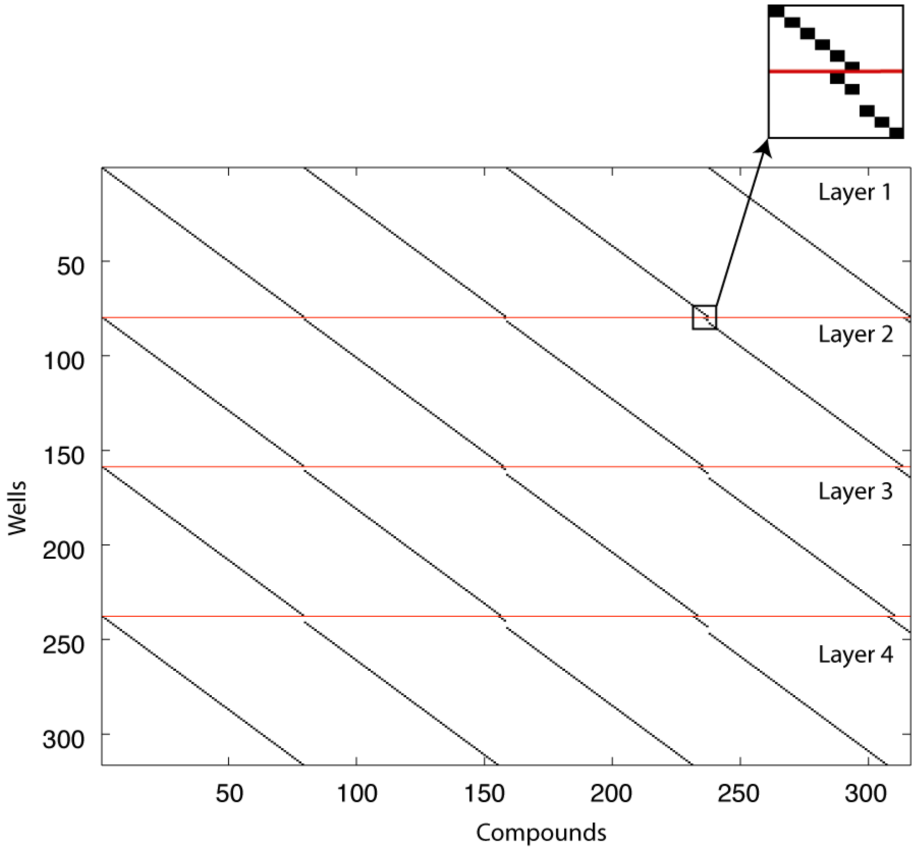

**Figure 4.5** The STD pooling design used for small proof-of-concept screen consisting of 316 compounds screened 4 times across 316 pools with each pool consisting of a mixture of 4 compounds. A layer represents the set of pools that contain each of the 316 compounds exactly once.

The activities of the 316 compounds tested individually in the singleplex, titled Singleplex 1, which was run alongside the multiplex, was compared to the activities of the compounds in the larger 2000 compound screen described earlier, titled Singleplex 2. Figure 4.6 shows the result of this comparison.

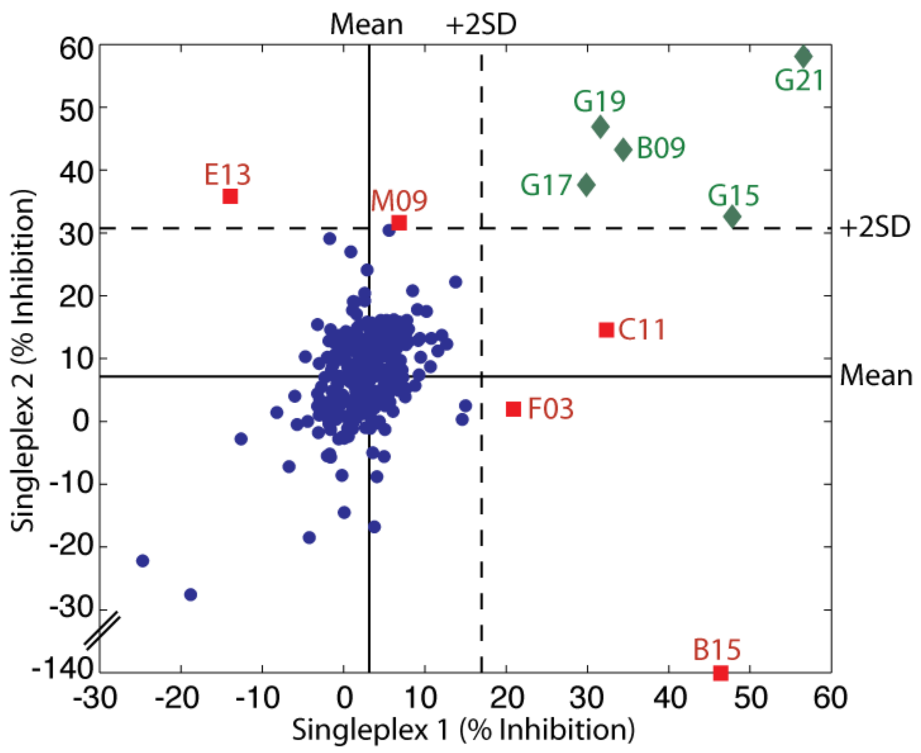

**Figure 4.6** The %Inhibition (per-plate) for 316 compounds from the MS-2000 library from two separate singleplex (one-compound-one-well) runs are shown in blue circles. The dashed lines represent a 2 SD cutoff applied to each singleplex run separately. The consensus hits are highlighted with green diamonds and the dissimilar hits are highlighted with red squares. The well numbers for these hits are displayed.

Compounds that inhibited the TR-FRET signal $\geq$ 2 SD from the mean percent inhibition measurements of the negative control and compound wells, taken together, were classified as active compounds or "hits". The cutoff applied to data from the two singleplex runs is shown in Figure 4.6. The cutoffs were applied separately to both the singleplex runs. Compounds that showed greater than 2 SD inhibition (dashed line) in both runs were classified as "consensus hits" (highlighted with green squares) – these included compounds in wells G21, G15, B09, G19, and G17. Compounds that showed greater than 2 SD inhibition in only one run were classified as "dissimilar hits" (highlighted with red squares) – these included compounds in wells B15, C11, F03, M09, and E13. The differences between the two runs give a sense of the false testing errors that can occur in automated screening. A false positive result wrongly classifies an inactive compound as active and a false negative result wrongly classifies an active compound as inactive.

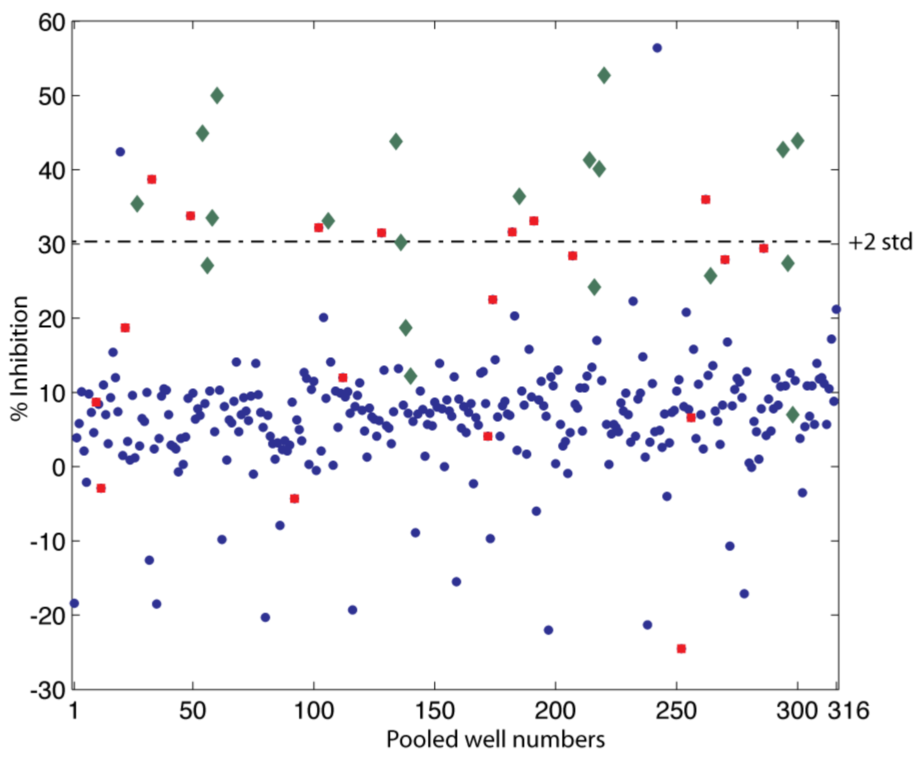

**Figure 4.7** The %Inhibition data for 316 pooled wells, each containing a mix of 4 compounds, from the multiplex run are shown in blue circles. As in the case of the singleplex runs, a 2 SD cutoff was applied to the observations to identify the wells that contained active compounds. Pooled wells that contained a compound classified as a consensus hit (green diamonds) or a dissimilar hit (red square) in the singleplex runs (Figure 4.6) are highlighted.

The multiplex run consisted of 316 pooled wells each containing a mixture of 4 compounds at the same concentration as the singleplex run. The %Inhibition measurements for each of the pooled wells in the multiplex run are shown in Figure 4.7. The figure shows the 2 SD cutoff and the corresponding classification of pools as “positive” or “negative.” As seen in the figure, most of the positive pools contain at least one compound classified as a consensus hit or a dissimilar hit in the two singleplex runs (see Figure 4.6). Several pools that did contain such compounds were classified as negative in the multiplex run, implying false negative testing errors. The next section describes the results of applying the two decoding procedures described earlier in this chapter – Interpool and QUAPO – to the multiplex data.

| Compound | Singleplex 1 | Singleplex 2 | Interpool | Compressed Interpool 1 | Compressed Interpool 2 | Compressed Interpool 3 | Compressed Interpool 4 | QUAPO |
|---|---|---|---|---|---|---|---|---|
| Consensus hits | | | | | | | | |
| G21 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ |
| G15 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| B09 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| G19 | ✓ | ✓ | | | | | | |
| G17 | ✓ | ✓ | | | | | | ✓ |
| Dissimilar hits | | | | | | | | |
| B15 | ✓ | | ✓ | | ✓ | | ✓ | ✓ |
| C11 | ✓ | | ✓ | | | ✓ | ✓ | ✓ |
| F03 | ✓ | | ✓ | | | ✓ | ✓ | ✓ |
| M09 | | ✓ | | | | | | |
| E13 | | ✓ | | | | | | |
| False positive (FP) and negative (FN) calls | | | | | | | | |
| FP | 3 | 2 | 4 | 2 | 1 | 4 | 6 | 8 |
| FN | 0 | 0 | 2 | 2 | 2 | 2 | 3 | 1 |

**Table 4.1** The decoding results for the consensus and dissimilar hits selected from the two singleplex runs are shown. A ✓indicates a hit call in a given run. False positive and negative call numbers are given relative to the consensus hits identified via the two singleplex runs. Therefore, by definition, the two singleplex runs have no false negative calls.

## 4.4 Multiplex decoding

The multiplex data was decoded using three approaches:

1. Interpool decoding – using data from all 316 pooled wells.

2. Compressed Interpool decoding – using data from only three layers of the pooling design, out of the four shown in Figure 4.5, thereby using 237 pooled wells and achieving 25% compression compared to singleplex. Removing one layer at a time, this analysis was performed four times.

3. QUAPO decoding – using data from all 316 pooled wells but with a quantitative approach to decoding.

The consolidated result of these decoding approaches is shown in Table 4.1. The result from each decoding approach is discussed in greater detail in the following subsections.

## 4.4.1 Interpool decoding results

As described earlier, the Interpool decoder analyses the results of a multiplex screen by finding an interpretation that is consistent with the observations at minimum cost. A consistent interpretation is found by switching the states of pools in the observation from positive to negative, or vice versa. An interpretation cost is only incurred when there are false testing errors. The cost structure used by the Interpool decoder, based on the observations in Figure 4.7 are given below.

1. Strong – A positive well with inhibition $\geq 3$ SD. The cost of switching this well to a negative state is 4.

2. Weak – A positive well with inhibition $\geq 2$ SD but $< 3$ SD. The cost of switching this well to a negative state is 2.

3. Faint – A negative well with inhibition $\geq 1$ SD but $< 2$ SD. The cost of switching this well to a positive state is 1.

4. Negative – A negative well with inhibition $< 1$ SD. The cost of switching this well to a positive state is 2.

The higher cost of switching a well result from positive to negative tunes the Interpool decoder to be highly sensitive to active compounds, at the risk of finding more false positives. The justification for this approach is that false negative results in high-throughput come with the risk of missing out on potential lead compounds. False positive results only add to the burden of secondary screening where each compound is tested retested for confirmation. However, depending on the screen, the cost structure can be altered easily.

The Interpool algorithm shown in Figure 4.5 was applied to the full multiplex dataset (all 316 pooled wells) using a software implementation provided by the authors (Thierry-Mieg and Bailly 2008) in the C language. The decoding procedure took less than 1 second to run on an Apple G5 (4 x 2.5 GHz). As seen in Table 4.1, the full Interpool decoder was able to identify three out of the five consensus hits (hence 2 false negative calls) along with four false positive calls, three of which were hits in Singleplex 1. The theoretical guarantee for the pooling design used in the multiplex run guarantees the identification of only three active compounds in the presence of no testing errors. However, according to the consensus hit list there were five active compounds and, using the 2 SD cutoff, this corresponded to nine false positive and eight false negative testing errors in the multiplex data, as shown in Figure 4.7.

It is interesting to note that here Singleplex 1 and the decoded multiplex agree more closely (when both consensus and dissimilar hits are taken into account) than the two singleplex runs do. This could be attributed to the fact that Singleplex 1 and multiplex were run in the same batch using the mosquito robot, whereas the Singleplex 2 data was extracted from a larger screen run separately. However, if we take the consensus hits as the gold standard it is worthwhile to ask why the error rate is higher in the multiplex (5.38% as opposed to 0.95% and 0.63% in the two singleplexes). One explanation could be that pooling compounds produces unexpected effects – unwanted reactions, concentration effects, etc. This could be partially addressed by optimizing the concentration of a pooled assay or avoiding mixtures of known reactants. Another explanation is that false testing results could be compound specific. Some active compounds may be more prone than others to produce false negative results; same goes for certain inactive compounds and false positive results. Such behavior could be attributed to their chemical properties. In a multiplexed screen each compound is tested multiple times across all the pooled wells. In the present case, each compound was tested four times across 316 pooled wells. Such repeated testing might also be the cause for higher false testing results in multiplex screens, as each compound gets more opportunities to behave. Either way, pooling designs may have to overcome higher testing errors in practice.

It is useful to note that the Interpool decoder also provides an interpretation cost for each active or ambiguous compound (data not shown). The cost provides a degree of confidence in the compounds call. However, it does not provide a quantitative or qualitative estimate of each compounds activity. Further, the Interpool decoder provides a short (2 to 4 compounds) list of “ambiguous” calls (data not shown). These call imply that in trying to come up with a consistent interpretation these compounds could not be classified as either positive or negative and hence need to be retested.

### 4.4.2 Compressed Interpool decoding

One of the primary intentions of using pooling design strategies in high-througput screening was to reduce screening effort by using fewer wells to test a compound library than the one compound, one well approach. However, as discussed in Chapter 3, the mixing constraint limits the amount of savings that can be achieved by pooling compounds. The pooling design used in the multiplex screen tested compounds in pools of four. That implies one layer (defined as the number of pooled wells that are needed to test each compound exactly once) required 316/4=79 pooled wells. Pooling designs also require that each compound is tested multiple times overall so as to enable decoding and to overcome testing error. However, the additional goal to use fewer wells than singleplex cannot be met even when using just four layers (79$\times$4=316), each testing the compound once. But it is possible to analyze post-hoc the performance of a compressed version of the pooling design. This can be done by removing a layer, and the corresponding pooled well measurements, from the pooling design and using the Interpool decoder on this compressed design. Due to the presence of a large number of active compounds and testing errors it is not worthwhile to remove more than one layer from the design in Figure 4.5. Therefore, the compressed decoding approach consisted of four runs each obtained by removing one of the four layers of the pooling design – this corresponded to a savings of 25% compared to the singleplex. For each layer removed, the corresponding wells were removed from the analysis and the pooling design, the 2 SD cutoffs recalculated, the cost structure readjusted according to the rules described earlier, and decoded using the Interpool algorithm.

The results of the compressed decoding approach are described in Table 4.1 under the heading *Compressed Interpool*. As shown in Table 4.1, it was still possible to identify several of the strongly active consensus hits correctly. As for the differing hits, the compressed runs did not have the same success, except in the case of layer 4. However, the Compressed Interpool decoding did not show more false positive or negative calls than the full Interpool decoding.

### 4.4.3 QUAPO

Although the Interpool decoder provides us with classifications of compounds – active, inactive, and ambiguous – and a confidence in that classification via the cost level, it does not provide a quantitative estimate of the compounds activities. An active call at zero cost in the Interpool decoder does not imply that the compound was quantitatively more active than an ambiguous call at cost three, say. However, as seen in Figure 4.7, there is lot of quantitative information generated in a pooled screen. The QUAPO decoding approach , described earlier, was developed to exploit this information.

For the QUAPO approach to succeed two principle requirements need to be satisfied:

1. Linearity – The measured quantity in a pooled well should map linearly to the activities of the compounds pooled in that well. This is required to enable the use of the linear algebra model described earlier.

2. Sparsity – The number of active compounds should be small. In the quantitative case this implies that the activity values of many if not most compounds should be close to zero. This is required to guarantee the success of the linear programming decoder described earlier.

If these requirements are satisfied, the QUAPO approach could successfully identify the active compounds along with quantitatively ordering these compounds by their activity. First, the linearity assumption was tested. As described earlier, the assay shown in Figure 4.4 can be modeled using Equation 4.10.

This model allows the transformation of the decoding problem to a linear algebra problem of the form shown in Figure 4.2.3. Using data from Singleplex 1 and the linear algebra model, a synthetic multiplex can be performed. First, the percent inhibition data was converted to an equivalent “association constant” measure or “activity” using Equation 4.10. This list of activities of all 316 compounds obtained from the Singleplex 1 data via this method was then multiplied with the pooling design shown in Figure 4.5 (whose entries are 0 or 1) arriving at the synthetic multiple measurements, as shown in Figure 4.2.3. The synthetic measurements can be compared to the actual multiplex measurement in order to check the linearity hypothesis, to within measurement noise. The results of this comparison are shown in Figure 4.8. As seen in the figure, there is a lot of scatter at the higher values. This is primarily due to large false positive and negative testing errors primarily in the multiplex, as discussed earlier. However, this comparison provides the starting point for the QUAPO analysis.

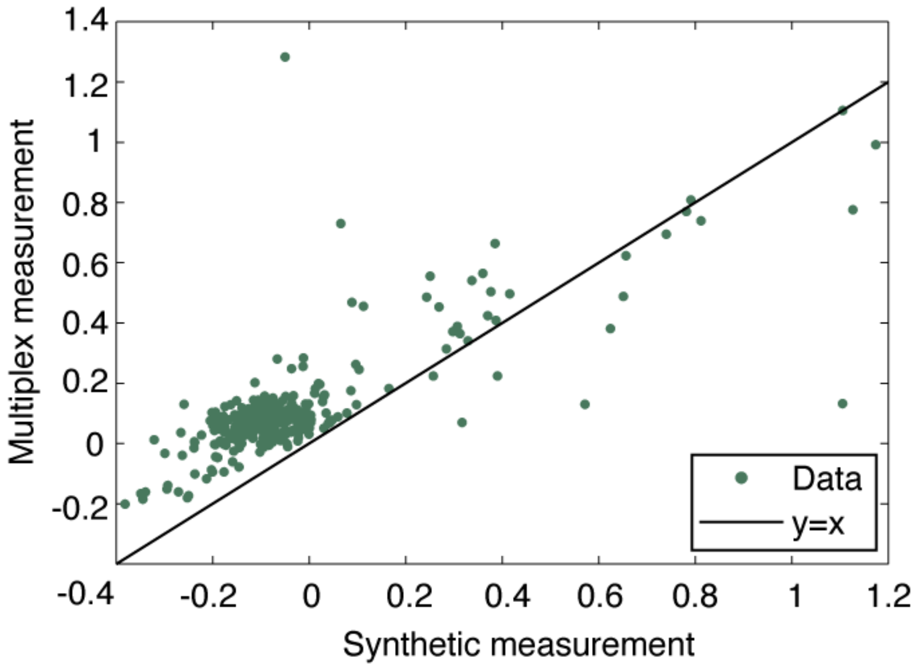

**Figure 4.8** Comparison of the synthetic measurements, obtained via an in-silico multiplex experiment performed using data from Singleplex 1 and the pooling design shown in Figure 3.5. This comparison is useful in estimating the success of the QUAPO decoding algorithm.

The QUAPO algorithm described earlier was applied to the multiplex data (from all 316 wells) using MATLAB software program (see Appendix C.2) using the $\ell_1$-magic linear programming function (Candès and Romberg, 2005). The decoding time on an Apple G5 (4 x 2.5 GHz) machine was on the order of a few seconds. The multiplex data on which the decoding was performed had been transformed according to Equation 4.10 and was retransformed to the percent inhibition scale using the reverse transform of the equation. The corresponding 2 SD cutoffs were applied and the consensus and differing hits were identified. The results of the QUAPO decoder are shown in Figure 4.9 and presented in Table 4.1.

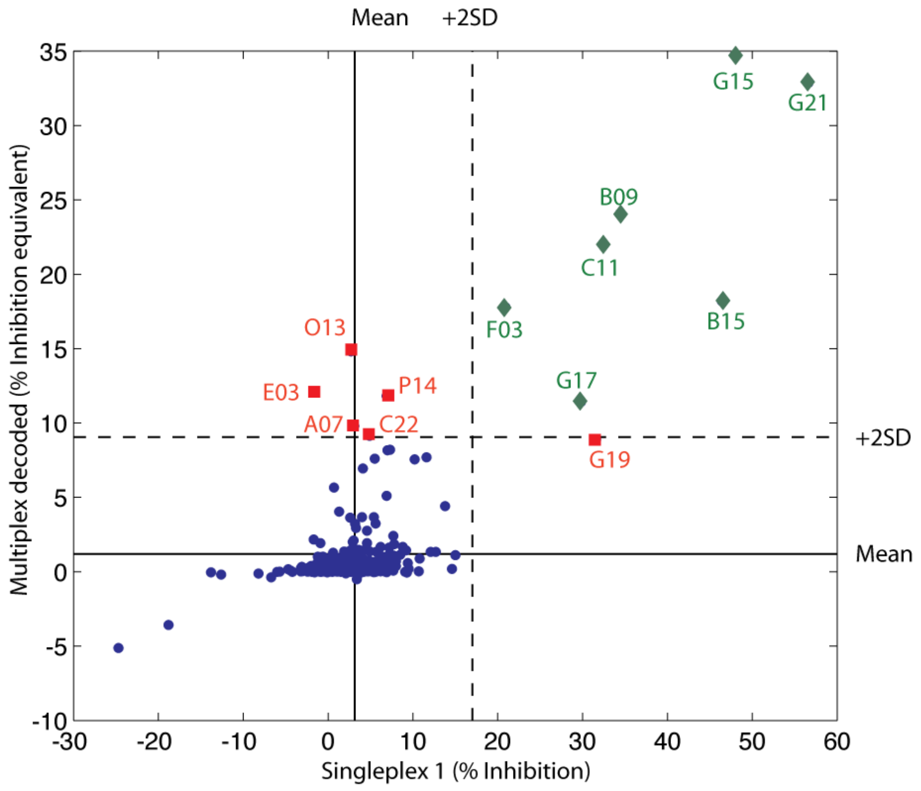

**Figure 4.9** Results of the QUAPO decoding procedure comparing the percent inhibition values of Singleplex 1 and to their % Inhibition-equivalent values in the decoded multiplex. As earlier, 2 SD cutoffs were calculated and applied. The well numbers corresponding to singleplex for the hits are provided. The consensus hits from both runs are highlighted with green diamonds, while the dissimilar hits are highlighted with red squares.

The figure shows the strong overlap in consensus hits between the singleplex and multiplex. As in the case of the Interpool decoder, a much larger list of consensus compounds was found between Singleplex 1 and Multiplex. However, due to the sensitivity of the QUAPO decoding procedure to large errors, which are non-gaussian in nature, there are a large number of false positive calls. However, this can be adjusted using a tuning parameter for the linear programming engine of QUAPO just as the cost structure for the Interpool decoder can be adjusted to balance sensitivity and accuracy. However, the accurate quantitative ordering of the active compounds, as seen in Figure 4.9, demonstrates the utility of the QUAPO decoding algorithm. It is worth noting that the scale of quantitative values for the singleplex and multiplex runs are different due to the application of the approximate transform shown in Equation 4.10.

## 4.5 Conclusions

This chapter discussed two different approaches to decoding the results of pooling design strategies in high-throughput drug screening. These approaches provide different advantages. The Interpool decoder accounts strongly for false testing results and provides the means to analyze the results of the pooling design independent of the pooling design parameters. The QUAPO decoder provides the means to exploit the quantitative nature of high-throughput screening data to arrive at a rank ordering of the active compounds. Modifications to these decoding methods, in order to improve their performance, are discussed in the concluding chapter of the thesis.

The chapter also provided a proof-of-concept experiment to test the validity of pooling design strategies using a small compound library of 316 compounds run in replicate using the "one compound, one well" strategy and once in a multiplex run using the shifted transversal design strategy.

# Chapter 5

# poolMC – Smart pooling of mRNA samples in microarray experiments

In this chapter, I extend the use of the quantitative methods developed in Chapter 4 to a novel application – gene expression microarray experiments. I propose a novel pooling strategy to make certain types of microarray experiments cost-effective and robust to measurement noise. I then describe a proof-of-concept experiment to test the smart pooling approach. In the context of this experiment, I discuss the potential, challenges, and future directions of using pooling designs in mircoarray experiments. The tool that came out of this application is called poolMC.

## 5.1 Background

Presently, pooling in microarray experiments refers to the act of mixing messenger RNA (mRNA) collected from several biological-replicate samples, before hybridization onto a microarray chip (Peng et al., 2003; Shih et al., 2004; Kendziorski et al., 2005; Zhang and Gant, 2005; Mary-Huard et al., 2007; Zhang et al., 2007). This form of pooling may be used to reduce biological variation, to lower costs by reducing the number of microarray chips used, and to overcome the problem of limited sample availability.

In this chapter, I describe a different pooling strategy; a smart pooling strategy based on compression algorithms from digital communication theory. The smart pooling strategy is applied to a large number of diverse biological samples, not necessarily biological replicates, which are pooled and tested on several microarray chips based on a pre-specified pooling design. The mathematical properties of smart pooling designs ensure that each sample is tested on multiple chips, but always in pools made up of a different set of samples, such that, data from all the chips taken together capture the same information as the standard one-sample-one-chip approach. Because of the convolution step involved in testing pools of samples on multiple chips, the measurements made from the smart pooling strategy must be decoded to obtain the gene expression value of each gene in every sample. To save cost and to accurately transmit information across digital communication channels, where bandwidth is limited and the channel is noisy, a similar compression and recovery strategy is used. Similarly, smart pooling can achieve an overall savings by using fewer microarray chips than samples being tested. The built-in redundancy of testing each sample on multiple microarray chips can also provide robust expression measurements. The gains of compression and robustness from using smart pooling strategies motivated us to investigate this method. Smart pooling strategies have been used in other high-throughput biological applications such as blood testing (Westreich et al., 2008b), drug screening (Kainkaryam and Woolf, 2008, 2009), protein-protein interaction mapping (Jin et al., 2007; Xin et al., 2009), genotyping (Erlich et al., 2009a; Prabhu and Pe'er, 2009), and others (Du and Hwang, 2006). By using commercially available gene chips to implement smart pooling, our method also differs significantly from other attempts to design compressive sensing DNA microarrays that apply smart pooling at the level of probes (Dai et al., 2009).

The pooling and decoding strategy is called poolMC – pooled microarray. poolMC is based on theoretical ideas from the field of compressive sensing, which has already demonstrated its utility in signal and image processing applications (Candès et al., 2006; Candes and Wakin, 2008). Compressive sensing takes advantage of the intrinsic compressibility of a data stream or set of experiments to produce experimental designs that require fewer experiments and provide greater robustness. Other experimental design strategies, derived from the field of group testing (Du and Hwang, 2000, 2006), have similar goals as compressive sensing but are not designed to obtain continuous-valued, quantitative measurements. A key requirement for any compression based experimental design is sparsity in the underlying data. The definition of sparsity in the context of gene expression data will be discussed in more detail in the following subsection.

To test the poolMC strategy, a small pooling experiment was carried out using validated biological samples on commercially available gene chips. In the context of this pooled experiment, we explain poolMC's pooling and decoding strategy. The results of this pooled experiment are then compared to the standard one-sample-one-chip method for the same set of samples. Finally, I propose the ideal setting for using poolMC and suggest its potential benefit for large microarray experiments.

## 5.2 Sparsity

A key requirement for any smart pooling strategy is that the data must be sparse to allow compression. In this paper, the data describing a single gene's expression pattern across several biological samples is called its expression profile. A gene's expression profile is said to be sparse if, across several samples, there are only a small number of samples in which the gene's expression is significantly different from its median expression value in these samples. In this context "different" is ill defined, so for practical purposes, "different" can be viewed as meaning differential expression. If a gene is differentially expressed in only a small number of samples across many being tested, then it is possible to exploit this sparsity to reduce the number of microarray chips needed to obtain the gene expression profile by pooling or multiplexing samples.

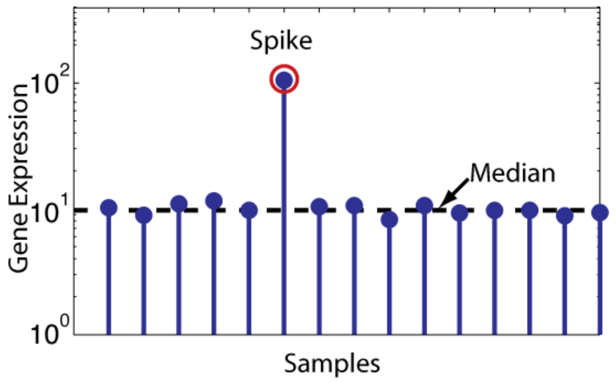

**Figure 5.1** Example of a gene showing only one spike (red circle) across 15 samples. The dotted line marks the median value for the samples.

Figure 5.1 shows an example of a gene with a sparse expression profile, because out of 15 samples the gene is differentially expressed in only one sample. By subtracting the median expression value from all samples only one sample, in this example, is left with a significant value while the rest of the expression values are close to zero. The significant value that is of interest is called a spike, as labeled in Figure 5.1. A spike can be a positive or negative deviation from the median depending on whether the gene was up or down regulated in a sample. Depending on the experiment there could be more or fewer spikes. For a particular experiment, different genes may display a wide variety in the number of spikes in their gene expression profile. However, most, if not all, genes in an experiment have a sparse expression profile across a given set of samples, this sparsity can be exploited via smart pooling to compress the experiment into a smaller and more robust design. One way to achieve these savings while maintaining robustness is by pooling samples in microarray experiments.

Before describing the details of the pooling method, I first identify two cases of microarray experiments that could produce sparse gene expression profiles. A first example of sparsity could be mRNA samples obtained from similar cell samples, but with different, non-overlapping gene knockouts. This example case would bias the expression profiles toward sparsity because many of the genes would have no difference in expression (zero spike) across the various knockout samples. Those genes that do change would likely change only in one genetic background. A second example of a sparse gene expression study is in biomarker discovery where the samples, classified as treatment (or control), would have few genes differentially expressed in the samples within each classification, though they may show great variation when compared across each other. The next section describes the concept of smart pooling and how it exploits sparsity in gene expression profiles.

## 5.3 Smart pooling

The central idea of smart pooling is to exploit sparsity in a gene's expression across several samples to obtain robust estimates of the gene's expression in all samples, while using fewer chips. This concept differs significantly from earlier microarray pooling attempts by not focusing on reducing the number of microarray chips used or increasing statistical power by estimating a single average expression value for each gene across multiple biological replicate samples. The goal of smart pooling is to recover the unique expression value of each gene in each of the samples being tested. To achieve this goal, samples are pooled in a systematic way across several chips while keeping the number of chips used lower than the number of samples tested. The pooling design ensures that each sample is tested on multiple chips, always in pools with different samples. The mathematical properties of the pooling design and decoding algorithm, discussed in more detail in the next section, guarantee that the expression of each gene in all samples is decoded accurately. For this strategy to succeed, two principle requirements need to be satisfied. First, as described in the previous subsection, gene expression profiles need to be sparse. Second, the intensity of pooled measurements should map linearly to the contribution from each sample being pooled in the measurement.

If the data are sparse and mixing is additive, the pooling strategy can be reduced to a linear system of equations for each gene. As shown in Figure 5.2, the pooling design specifies the samples to be mixed and tested on microarray chips. However, in reducing the number of chips, we make fewer measurements than we have samples (e.g. 12 measurements of 15 samples in Figure 2). This mismatch between measurements and unknown variables (the gene's expression value in the samples) results in an underdetermined system of equations, which has an infinite number of solutions. However, by assuming that the underlying data

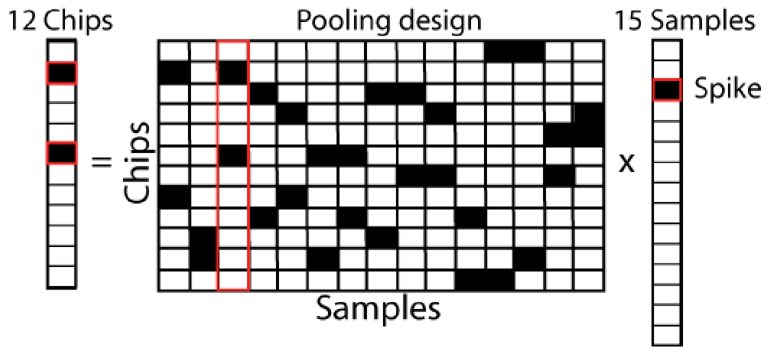

**Figure 5.2** Schematic showing the pooling process and the utility of a sparse expression profile. The right column shows a gene's expression across 15 samples, with only 1 spike (highlighted dark square). Samples are mixed according to the pooling design in the middle. The columns of the pooling design represent the samples being pooled and the rows represent the microarray chips used to test them. A black square in the pooling design represents the presence of the sample (along that column) on the corresponding chip (along that row). The highlighted column in the pooling design shows the sample that corresponds to the spike. The left column shows the resulting measurements that contain only two significant values (dark squares), those coming from the sample with the spike.

are sparse, the number of possible solutions are reduced and the measurements contain enough information to uniquely solve the system of equations. Figure 2 illustrates this process. The right column shows a gene with only 1 spike (black square) in its expression profile, corresponding to sample number 3. When the samples are pooled according to a pre-specified pooling design shown in Figure 5.2 as a matrix, where a black square represents the presence of a sample (represented by a column) in a specific pool along that row, sample number 3 appears in pool numbers 2 and 6. The left column in Figure 5.2 shows the spike (black square) appearing in measurement numbers 2 and 6, as expected. Thus, the sparsity of an expression profile affects the number of useful measurements obtained by the pooled experiment thereby allowing the system of equations to be solved to obtain a unique solution. The redundancy attained through multiple measurements – two in this case – also provides a way to improve the robustness of the predictions.

## 5.4 poolMC

This section describes the pooling design and decoding algorithm that together form the poolMC smart pooling strategy. The details of the pooling design construction, the laboratory protocol for pooling, and the decoding of the results are described in the context of a small pooling experiment that was performed to test poolMC experimentally.

### 5.4.1 Experimental details

The current one-sample-one-chip microarray experiment is termed the monoplex. The pooled microarray experiment is called the multiplex. Because the data measured by the monoplex experiment are directly used, these data serve as both the monoplex measurements and the monoplex results. However, the multiplex experiment has two parts. The multiplex measurements refer to the data obtained from pooling samples on microarray chips, hence fewer measurements than the monoplex, and multiplex result refers to the data obtained from decoding the multiplex measurements. The multiplex result can be directly compared with the monoplex result. A synthetic mutliplex was performed using the monoplex results and multiplying them with pooling design for each gene in the system. This produced a synthetic measurement, comparable to the multiplex measurement, and by decoding the synthetic measurements produced the synthetic result, comparable to the monoplex and multiplex results. The synthetic multiplex helped investigate the linearity assumption, by comparing multiplex and synthetic measurements, and the sparsity assumption by comparing the synthetic and monoplex results.

A proof of concept test was carried out using 15 mRNA samples obtained from the root epidermis of the plant Arabidopsis thaliana. These samples included 4 pairs of biological replicates, hence 8 samples, and 7 independent knockout samples. The monoplex measurements were performed at the usual concentration using 15 microarray chips. The multiplex measurements were obtained by mixing fractions of individual samples and using 12 microarray chips, employing the pooling design shown in Figure 5.3 (right). The choice of this pooling design is explained in the next subsection.

The mixing of fractions of individual samples was conducted at the mRNA level. Following isolation of mRNA from plant material (using the Qiagen RNeasy kit), the concentration of each sample was determined (Dinneny et al., 2008). For the monoplex experiment 3-3.5 $\mu$g mRNA was used. For the multiplex the samples were pooled based on the pooling design described in the subsection below, such that a total of 3.5 $\mu$g mRNA per chip was used. The cDNA was generated and labeled using the NuGen Ovation v2 and the NuGen Ovation FL kit. A total of 4 $\mu$g of the labeled cDNA were hybridized on the Affymetrix ATH1 Genechip.

The resultant mircoarray hybridization signal data corresponding to monoplex and multiplex measurements were preprocessed separately using RMA (Irizarry et al., 2003) and annotated using Brainarray custom CDF (Dai et al., 2005). Each chip simultaneously measured the expression value of 21,505 genes. The microarray measurements obtained from the monoplex and the multiplex were normalized separately to ensure that the data were handled as though they were obtained from independent experiments, as typically expected.

### 5.4.2 Pooling design

Several pooling methods have been discussed in the literature (DeVore, 2007; Berinde and Indyk, 2008). The design shown in Figure 5.3 (right) and used in the experiment was based on an expander graph construction used by Berinde et al. (2008), which is defined as follows.

#### Expander graph definition

A $(k,\varepsilon)$-unbalanced expander is a bipartite simple graph $G - (A,B,E)$ with left degree $d$ such that for any $X$ in $A$ with $|X| \leq k$, the set of neighbors $N(X)$ of $X$ has size $|N(X)| \geq (1-\varepsilon)d|X|$, where $|\cdot|$ represents the cardinatlity of a set.

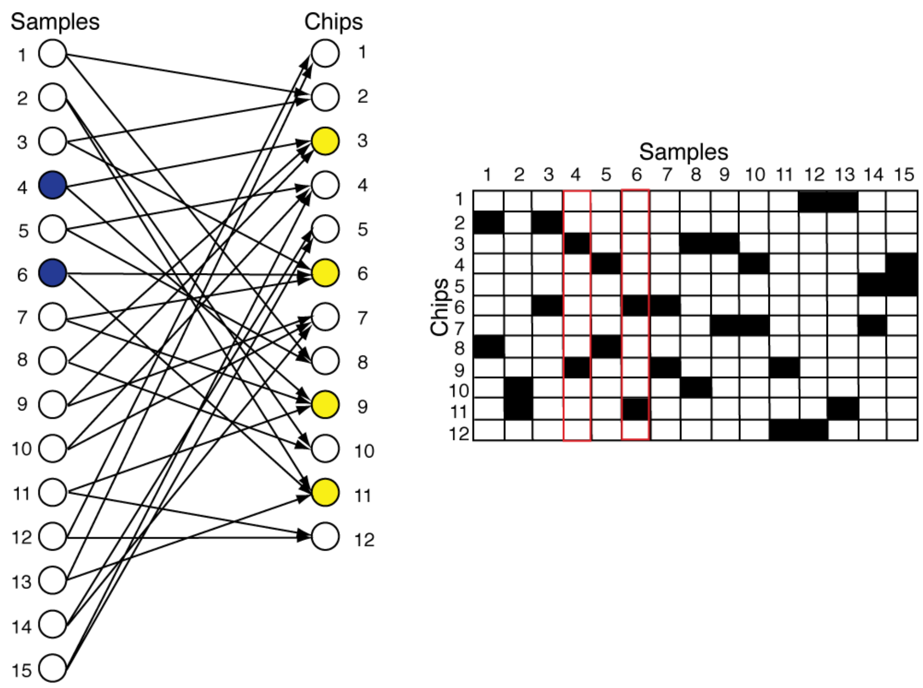

**Figure 5.3** Demonstration of expansion property of an expander graph. Two dark-colored nodes (samples) on the left of the graph expand to four different light-colored nodes (chips) on the right of the graph, thus providing 100% expansion ($\varepsilon = 0$) for this particular pair of samples. The binary matrix on the right of the figure is the matrix representation of the expander graph on the left of the figure. Black squares in the matrix represent the presence of a sample (along the column) on a chip (along the row).

Figure 5.3 illustrates this property of an expander graph in the context of the pooling design used in this paper. The pooling design tests 15 samples using 12 pooled chips. The expander graph shown on the left of the figure can be represented as a binary matrix, shown on the right. The presence of a sample on a chip is represented by the value 1 (black square in Figure 5.3). The left degree ($d$) of this graph is 2, that is, there are 2 edges emanating from each node on the left, which means each sample is tested on exactly 2 chips.

Definition 5.4.2 is illustrated with the example of samples 4 and 6, the left nodes shaded in blue in Figure 5.3. The right neighbors for these two samples ($|X| = 2$ in Definition 5.4.2) are the chips numbered 3, 6, 9, and 11. Thus, $|N(X)|$ in Definition 5.4.2 is equal to 4. Therefore, $\varepsilon$ in Definition 5.4.2 for this example is equal to 0. Similarly, the expansion coefficient $\varepsilon$ can be calculated for other pairs of samples. For the design in Figure 5.3, $\sim$ 77% of all pairs have $\varepsilon = 0$, the rest have $\varepsilon = 0.25$. Therefore, for simplicity, the design used in this experiment can be considered an ($k = 2, \varepsilon = 0$)-unbalanced expander graph, as defined in Definition 5.4.2. This simplification does not affect the demonstration of the theoretical guarantee of decoding. The utility of the expansion property of the pooling design is illustrated in the next section on the theoretical guarantee of decoding.

The pooling design is constructed by randomly placing $d$ ones in each column of an $m \times n$ matrix; the rest of the entries are zeros. This method guarantees a good expander graph, with high probability, when choice of d and m are made appropriately for a given $n, k$, and $\varepsilon$. Proposition 6 in Berinde et al. (2008), suggests that $d = O(\log(n/k)/\varepsilon)$ and $m = O(k \log(n/k)/\varepsilon^2)$. The MATLAB software implementation of the pooling design construction is provided in Appendix B.1.

In practice, another constraint is imposed on the pooling design. The number of samples that can be mixed on each chip is limited, thereby placing a cap on the number of ones in each row of the pooling design matrix. However, since sparse matrices with good expansion properties are needed, this cap is automatically satisfied, especially for large values of $m$ and $n$. However, in the software implementation, checks are imposed on the row weight of the pooling design and reject designs with high row weights. Further, instead of pooling each sample at full concentration, fractional concentrations of all samples are used in practice. The fractions are determined by row weights such that they sum up to a concentration equivalent to testing one sample, at full concentration, on each chip. Therefore, in practice, the pooling design is not a binary matrix, but has entries that are fractions summing up to one in each row. Finally, due to the small experiment size (12 chips for 15 samples), it was possible to exhaustively calculate the expansion property of the pooling design for low values of $k$ (up to 2), thereby allowing us to choose a “good” design (where $\sim$ 77% of all pairs of spikes had $\varepsilon = 0$).

### 5.4.3 Decoding strategy

The poolMC decoder solves a linear program of the form:

$$\min ||\hat{x}||_1 \text{ subject to } ||\Phi\hat{x} - y||_2 \leq \delta \qquad \text{(P1)}$$

Here, $\Phi$ is the pooling design, $y$ is the pooled measurement, and $\delta$ is a constraint relaxation parameter. poolMC uses the l1-magic software package (Candès and Romberg, 2005) to solve the linear program. If the gene's expression profile is sparse, then Theorem 5.4.4 guarantees accurate decoding. This guarantee is discussed in detail in the next subsection.

As seen in Figure 5.3, gene expression profiles are not sparse in the strictest sense. They can, however, be sparse when their deviation from the median value is considered. Therefore, poolMC's decoder operates on the median-subtracted gene expression. An approximate median value is arrived at from the median value of the pooled measurements. Since the pooling design used in practice preserves the magnitude of the gene's expression values (its rows sum up to 1), taking the median of measurements gives a good approximation to the actual median value. The decoder then operates on the median-subtracted version of the measurements. The MATLAB software implementation of the poolMC decoder is provided in Appendix B.2.

### 5.4.4 Theoretical guarantee of decoding

poolMC provides the following decoding guarantee (Berinde et al., 2008).

#### Decoding guarantee theorem

Let $\Phi$ be a $m \times n$ matrix of an unbalanced $(2k, \varepsilon)$ expander. Consider any two vectors $x, \hat{x}$, such that $\Phi x = \Phi\hat{x}$, and $||\hat{x}||_1 \leq ||x||_1$. If $S$ is the set of $k$ largest (in magnitude) coefficients of $x$, then,

$$||x - \hat{x}||_1 \leq C(\varepsilon) \times ||x - x_S||_1$$

As shown in the previous section, the pooling design used in the experiment described in the paper is an (2,0)-unbalanced expander. Therefore, $k = 1$, in Theorem 5.4.4, which implies that genes with a single spike in their expression profile are guaranteed to be recovered with an error bounded by the right hand side expression. The error bound consists of two parts – the deviation from the one-spike assumption of the design and the scaling constant dependent on the design. The scaling constant is a function of the expansion coefficient $\varepsilon$ of the expander graph, $C(\varepsilon) = 2/(1-2\alpha(\varepsilon))$ where $\alpha(\varepsilon) = (2\varepsilon)/(1-2\varepsilon))$. Figure 5.4 shows the relationship between these parameters. It would be advantageous to have a low value for $\varepsilon$ as it would imply a low scaling factor. For $\varepsilon = 0$, the scaling factor is 2. The effect of deviation from the sparsity assumption on the error guarantee of the decoding strategy is discussed next.

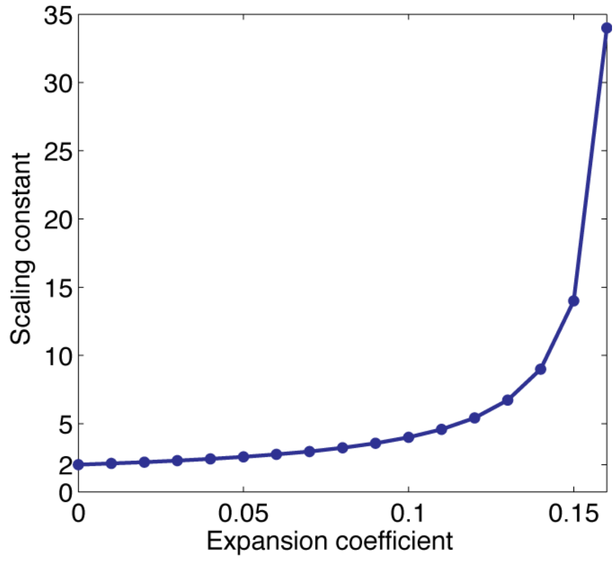

**Figure 5.4** Variation of the scaling constant in Theorem 5.4.4 as a function of the expansion property of the expander.

As shown in Figure 5.2, the success of smart pooling depends on the sparsity of gene expression. Therefore, the theoretical guarantee for accurate recovery, Theorem 5.4.4, is stated in terms of the sparsity, or lack thereof, of the underlying gene expression. Conventionally, the mathematical arguments treat non-spikes as having a value close to zero. However, in the case of gene expression, the non-spikes have a value close to the median, as seen in Figure 5.1. Therefore, a median-subtracted version of the gene expression profile is used in our analysis, a sample of which is shown in Figure 5.5.

Theorem 5.4.4 can be restated as follows:

$$\%\text{Decoding} \quad \text{error} \leq \text{Constant} \times \%\text{Deviation} \quad \text{from} \quad \text{sparsity} \tag{5.1}$$

The decoding error refers to the total difference, across all samples, between decoded and actual (true) expression for a gene and is scaled to magnitude of the expression profile. According to equation 5.1, the percent decoding error will always be less than or equal to the percent deviation from the one-spike sparsity assumption made by the pooling scheme. The scaling constant depends on the pooling scheme used and is greater than or equal to 2, as described in the previous section. The guarantee provided by a smart pooling method, stated in equation 1, helps guide the choice of parameters for the pooling scheme design.

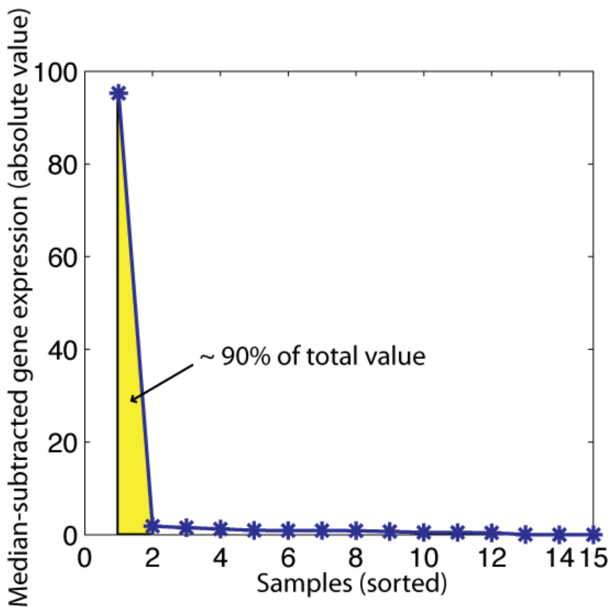

**Figure 5.5** Absolute value of median-subtracted gene expression for an example gene, sorted by magnitude.

This result is illustrated using the example of the monoplex data for gene AT5G54740, shown in Figure 5.5. For this gene, 90.41% of expression value is contained in a single sample. Therefore, we can use the pooling scheme described in the previous section, which was designed for a single spike.

Therefore, for the gene shown in Figure 5.5, the deviation from the one-spike scenario is 100%-90.41%=9.59%. Therefore, the right-hand side (RHS) of equation 1 takes a value of 2 x 9.59% =19.18%.

A synthetic pooling experiment was performed using the data shown in Figure 5.5, using the design shown in Figure 5.3, in a noise-free setting. The decoded result of the synthetic pooling experiment is shown in Figure 5.6. This example can be recreated by running the MATLAB code provided in Appendix B.3.

The percent decoding error, left-hand side (LHS) in equation 5.1, for the gene shown in Figure 5.6 is equal to 9.39%. While the scaling constant times the percent deviation from sparsity, the right-hand side (RHS) of equation 5.1, was 2 $\times$ 9.59% =19.18%. Therefore, the theoretical guarantee in equation 5.1 is satisfied.

The error guarantee in Theorem 5.4.4 is universal. Figure 5.7 demonstrates this for the decoded results from the synthetic pooling experiment for all 21,505 genes in the system. As shown in the figure, the decoding error is always below the theoretical error bound guarantee.

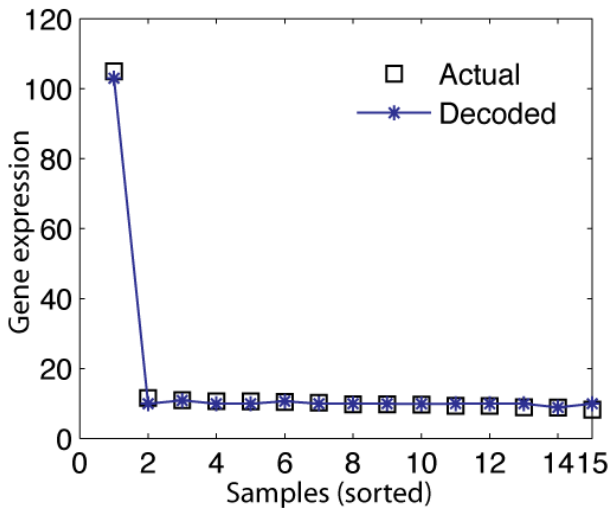

**Figure 5.6** Overlay of actual expression profile and decoded result of a synthetic pooling experiment for the gene shown in Figure 5.5.

It also shows that the decoding error scales linearly with the deviation from sparsity.

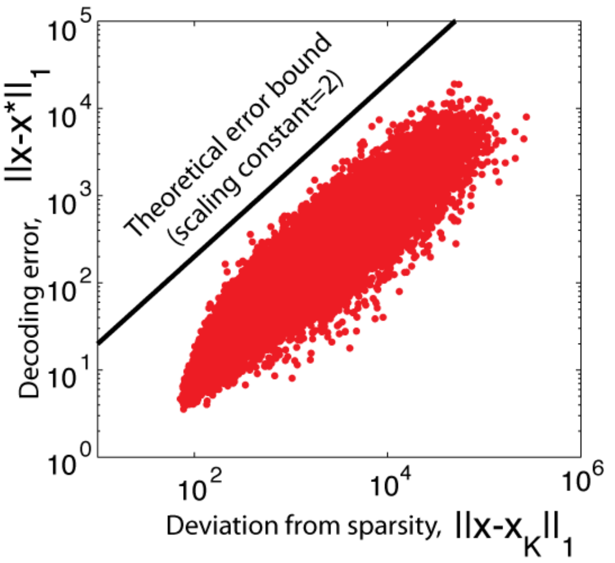

**Figure 5.7** A demonstration of Theorem 5.4.4 via a synthetic pooling experiment using the monoplex data, showing that the decoding error (y-axis) is always below the theoretical guarantee.

However, the conditions for the error bound in Theorem 5.4.4 are violated in practice in the following three ways:

1. In real systems there is noise in measurements. The effect of measurement noise is similar to deviation from sparsity in increasing the error bound on the decoded results.

2. As described in the pooling design section, the pooling design used in practice is not

strictly a binary matrix (rescaled by a constant factor) as each row is rescaled so that it sums to 1. The effect of this deviation from the conditions under which Theorem 5.4.4 holds is not clear. Although, the synthetic and experiments described in the paper were not affected by this change. For large enough pooling designs constructed randomly as in Berinde et al. (2008), the row weights cluster tightly about their mean value and our rescaling procedure produces a matrix which deviates only slightly from a truly binary matrix.

3. The poolMC decoder operates on the median-subtracted version of the measurements. However, the discrepancy in the estimation of the median adds to the decoding error. The decoding error due to median-approximation increases as the deviation from sparsity, hence the error in median estimation, increases.

### 5.4.5 Analysis procedure

The decoding procedure described in the previous sub-section was applied individually to each of the 21,505 genes in the system to obtain each gene's expression in all 15 samples.

Both the multiplex and synthetic pooled measurements were decoded and analyzed using the same procedure. The only source of noise in synthetic measurements is the noise resulting from measurements of the individual samples, and as such can be used to identify the sensitivity to experimental errors.

The success of poolMC was analyzed by directly comparing the monoplex and multiplex results. To test the linearity hypothesis, the multiplex measurements were compared with the synthetic measurements. To check if both datasets behaved similarly under standard microarray data analysis techniques, six differential expression analyses were performed between each of the 4 pairs of biological replicate samples present in the dataset for all three results – monoplex, multiplex, and synthetic. The overlap of significant genes obtained from the analysis in each of the six cases was used to evaluate the similarity of the monoplex to multiplex results.

## 5.5 Experimental results

poolMC depends on two key assumptions – linearity and sparsity. To test the linearity assumption the multiplex measurement was compared with the synthetic measurement. Figure 5.8 shows the alignment between synthetic and multiplex measurements, indicating that mixing samples produces measurements that are linearly additive. In Figure 5.8, we observe a greater disagreement between the synthetic and multiplex data in the low expression range.

This disagreement is likely due to measurement noise, because the synthetic measurements were simulated using the monoplex results, while the multiplex measurements were directly measured. As the expression level drops, the signal to noise ratio of the assay decays, generating discrepancies even between technical replicates. The slight deviation from the 45 degree line in Figure 5.8 is due to the monoplex, hence synthetic, and multiplex measurements being preprocessed separately.

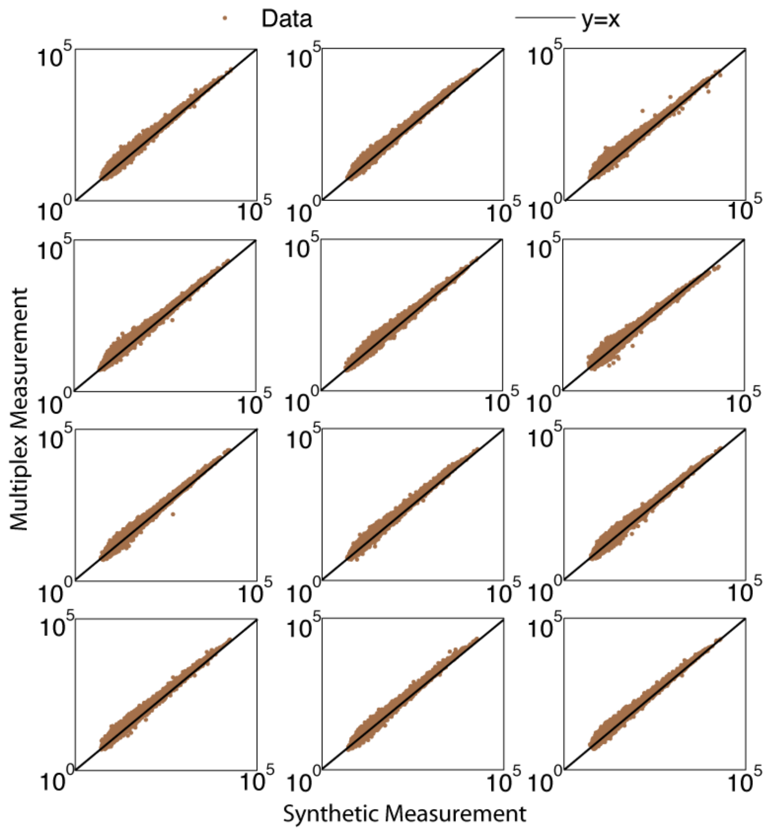

**Figure 5.8** Comparison between synthetic and multiplex measurement for all 12 pooled samples, showing all 21,505 genes.

Having confirmed the linearity assumption, the poolMC linear programming decoder was applied to both the synthetic and multiplex measurements. Overview figures for all samples are shown in Figures 5.9 and 5.10. These results demonstrate that overall the synthetic case provides a better fit to the monoplex result. This better fit is expected because the synthetic measurements have no measurement noise, relative to the monoplex. However, where the synthetic and multiplex results show large deviations from monoplex results, they do so in similar patterns, implying that the decoding error is due to deviation from the sparsity assumption of the design.

Next the data at the individual gene level was examined to determine how well the multi-

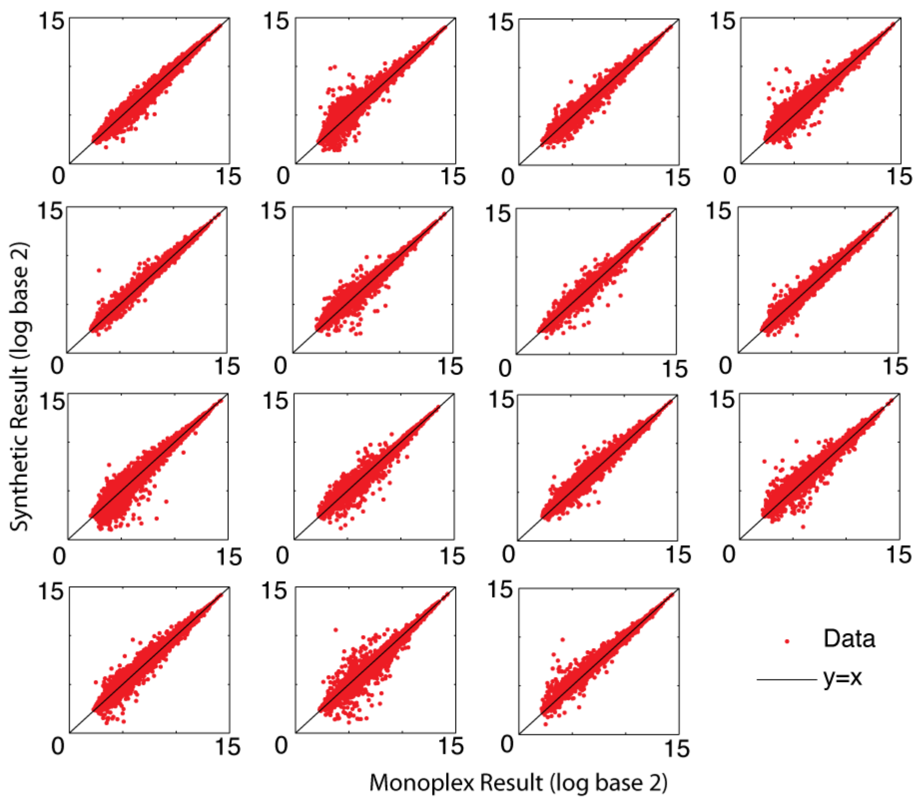

**Figure 5.9** Comparison between monoplex and synthetic pooling results for all 15 samples, showing all 21,505 genes.

plexed result could recover the monoplexed result. Figure 5.11 shows four representative examples of individual genes, under different conditions of sparsity and measurement noise.

The four cases in Figure 5.11 are described below, numbered according to the figure.

a A gene with exactly 1 spike and close to no measurement noise (strong alignment between synthetic and multiplex measurements in inset) is decoded accurately, as guaranteed by the pooling design used.

b A gene with exactly 1 spike but with significant noise in a measurement (far right data point in inset is not aligned), is not decoded accurately.

c A gene with multiple spikes but low noise is decoded with moderate accuracy, even though the number of spikes exceeds the algorithmic guarantee.

d Gene with multiple spikes and low noise is not decoded accurately due to a larger deviation from the sparsity assumption than in (c).

Although, the four examples are only a small sampling, the patterns shown in Figure 5.11

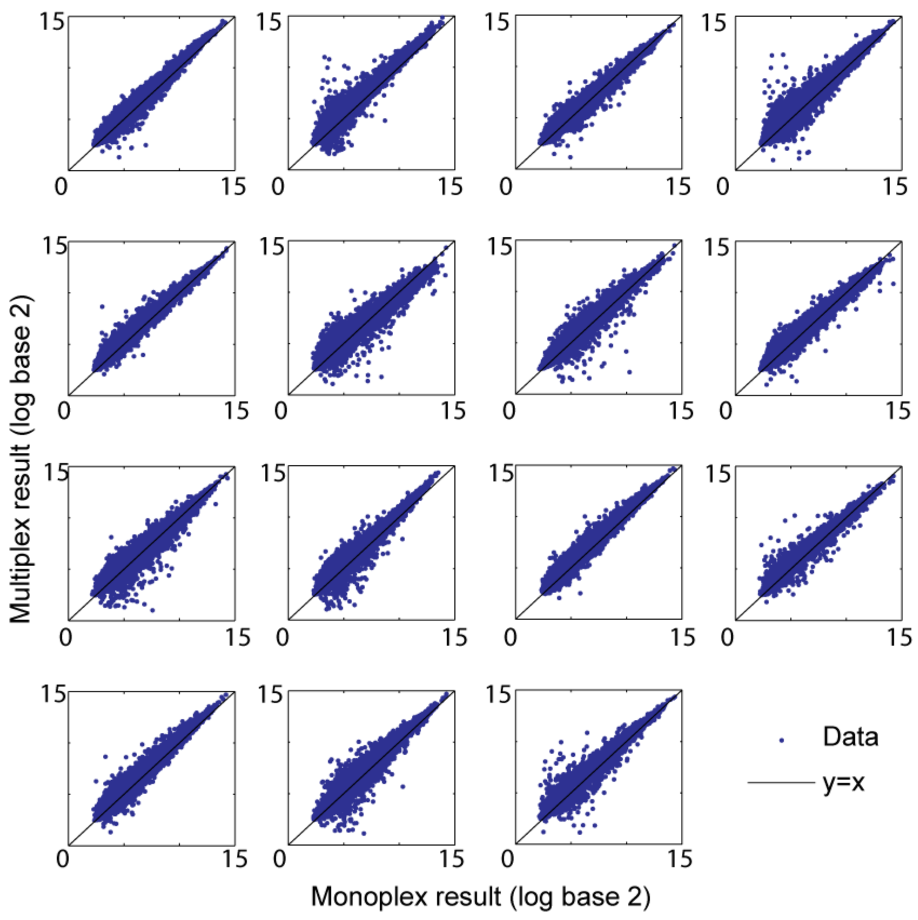

**Figure 5.10** Comparison between monoplex and multiplex results for all 15 samples, showing all 21,505 genes.

are representative of the properties of the decoding algorithm. The suggested experimental settings that will maximize the utility of smart pooling are discussed in the next section.

Given that the monoplex and multiplex results often produce the same pattern of expression, we next compared the lists of differentially expressed genes obtained from both lists. Because the 15 samples contain 4 pairs of replicate sample measurements (samples 2-4, 3-5, 7-8, and 12-15), differential expression analysis was performed between pairs of biological replicates using the Significance Analysis of Microarrays (SAM) method (Tusher et al., 2001) to obtain lists of significantly expressed genes. The SAM analysis compares the expression data for two or more sample types (treatments or conditions) to identify genes that were differentially expressed among them. Further, it uses a permutation-based method to identify is the differential expression was significant. Hence the need replicate measurements for each sample type. The SAM analysis was carried out independently within each dataset – monoplex, synthetic, and multiplex – and the resulting lists of differentially expressed genes were compared across the datasets (Table 5.5). Before carrying out

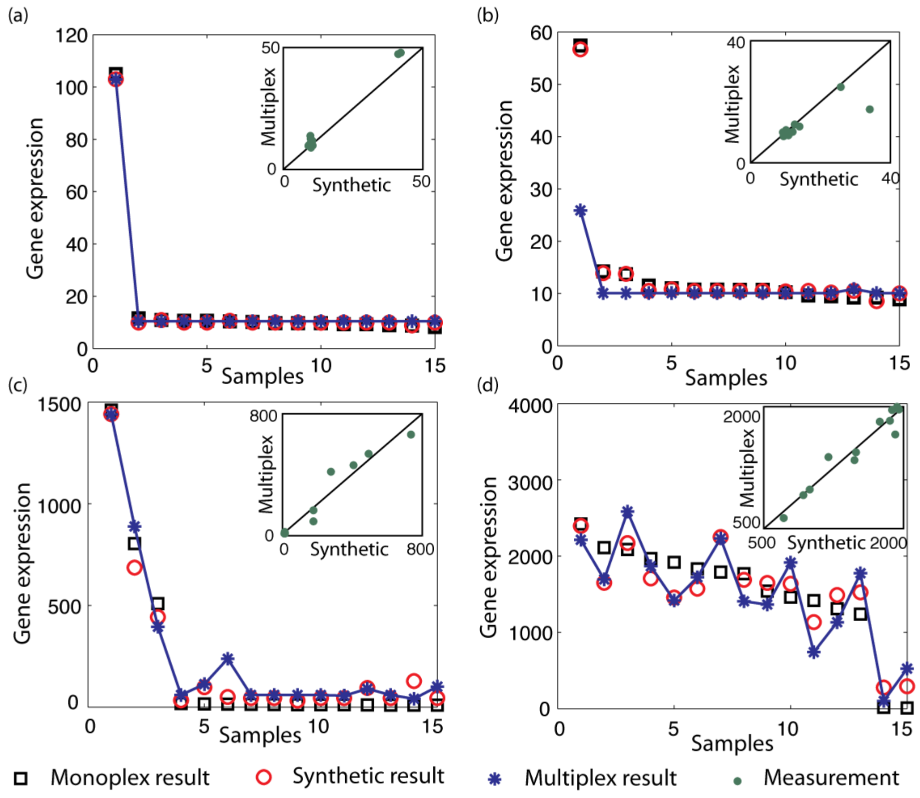

**Figure 5.11** Four examples of decoding performance. (a) Low spike, low noise case, (b) Low spike, high noise case, (c) High spike, low noise case, and (d) High spike, high noise case. For each gene, expression profiles from monoplex result (black square), decoded synthetic result (red open circle), and decoded multiplex result (blue star with lines) are shown. Inset shows the alignment between synthetic and multiplex measurements (green dots) for the gene across the 12 pooled samples. Raw gene expression values are shown.

the SAM analysis 50% of the 21,505 genes were filtered out based on variance to increase statistical power (Hackstadt and Hess, 2009).

In all six comparisons, the longest significant-gene lists were produced by the monoplex dataset. The multiplex and synthetic datasets identified fewer differentially expressed genes at the 10% q-value cut-off. As expected, the synthetic dataset shows more overlap with the monoplex than the multiplex does. This difference in overlap can be attributed to the decrease in noise in the synthetic dataset. The significant disparity in number of significant genes common to both monoplex and synthetic datasets can be attributed to the lack of sparsity in a large number of genes in the 15 samples chosen for the experiment, resulting in decoding errors.

| Sample pairs tested | # of significant genes at q=10% | | | # Overlap with Monoplex | |
|---|---|---|---|---|---|
| | Monoplex | Multiplex | Synthetic | Multiplex | Synthetic |
| 2-4 v. 3-5 | 761 | 43 | 13 | 27 | 13 |
| 2-4 v. 7-8 | 1545 | 3 | 5 | 3 | 5 |
| 2-4 v. 12-15 | 163 | 69 | 175 | 21 | 87 |
| 3-5 v. 7-8 | 1 | 0 | 0 | 0 | 0 |
| 3-5 v. 12-15 | 280 | 2 | 1 | 2 | 1 |
| 7-8 v. 12-15 | 26 | 1 | 1 | 0 | 1 |

**Table 5.1** Comparison of significant calls, obtained via a SAM analysis between biological replicate measurements from monoplex, multiplex, and synthetic datasets. The overlap of significant genes (at q-value=10%) common to monoplex and each of multiplex and synthetic datasets for each biological replicate pair tested are shown.

## 5.6 Conclusions

In this chapter, I investigated pooled experiments following a compressive sensing inspired design and decoder (poolMC) would be effective for gathering gene expression data. The approach described here is easy to implement with existing chips, as it only requires an intelligent mixing of samples. The analysis results indicate that gene expression measurements are sufficiently additive to be amenable to a linear decoder, and in some cases are sufficiently sparse to be compressed. When the experimental noise was sufficiently low and the expression profile was sparse, we have shown that poolMC can provide experimental compression.

However, the overall agreement between the lists of differentially expressed genes from the monoplex and multiplexed results did not show a strong overlap. This lack of overlap was due to three factors: (1) experiment size, (2) sparsity of gene expression, and (3) experimental noise. In this study, we carried out a small pilot study with only 12 multiplexed results compared to 15 monoplexed results. At this scale, multiplexing provides relatively few benefits in terms of both compression and robustness. As is shown in Figure 5.12, the compressive abilities of a poolMC type design increase significantly as the design is enlarged. The number of chips needed to identify $k$ spikes in a experiment of size $n$ is approximately $k\log(n/k)$. For example, a study with 100 samples would require approximately 23 chips, if only 1 in 10 samples showed differential expression. However, it should be noted that these calculations ignore measurement noise and should therefore be treated as a lower bound on the experiment size.

The second factor was the sparseness of the gene expression data. As is shown in Figure 5.12, the more spikes in a sample, the larger the required design. Gene expression data

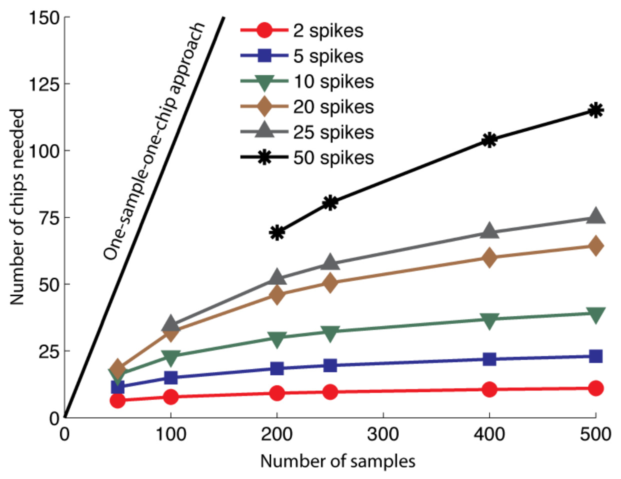

**Figure 5.12** The approximate number of microarray chips needed based on number of samples used for the pooling experiment and the number of spikes expected in the samples.

is not inherently sparse, but can be, depending on the particular samples chosen. In the experimental case used here, the data contain a large number of genes that do not change, corresponding to zero spike cases. These zero spike cases are in general accurately recovered from the multiplexing, but are of less interest than the cases that show differential expression. An ideal situation for multiplexing gene expression data would be a large dataset where there are only a small number of unusual samples, such as for screening for a rare disease or mutation. Alternately, the decoding methods could incorporate information about the correlation structure of the genes to better exploit sparsity of the data. Finally, it should be noted that negative spikes could be more difficult to decode than positive spikes because negative spikes have a bounded magnitude that cannot go below zero. In contrast, up-regulated genes are essentially unbounded in their magnitude, and as such easier to decode.

The third factor is experimental noise. Gene expression profiling is a well-standardized method with relatively little noise, however this study shows that even the technical noise present in gene expression profiling can cause artifacts in multiplexed results. The experimental design can be modified to increase the robustness of the predictions to noise, but at the cost of adding additional experiments.

This small experiment suggests that pooling can be carried out successfully within the theoretical guarantees provided by poolMC. In practice, however, there are a number of strong requirements of experiment size, sparseness, and assay noise that need to be considered to make pooling work well for gene expression profiling. Figure 5.12 implies that for smart pooling to be successful, the number of samples used should be increased and the type of experiment should be chosen carefully to avoid a large number of spikes. Finally, it would be interesting to study the connections between the linear decoding procedure used in this paper and other $\ell_1$ regression methods such as Lasso (Tibshirani, 1997).

# Chapter 6
# Conclusions and Future Directions

In this thesis I took up the challenge of making large-scale, high-throughput biological experiments more efficient and robust. Much of the challenge lay in identifying analogies between biological experiments and communication theory and importing ideas used in the latter to solve problems in the former. In this chapter I will discuss some of the key contributions of this thesis towards this challenge. I will also highlight ideas that represent useful avenues of inquiry for improving the methods described in this thesis and their application. Finally, I will speculate about other areas in which the pooling design framework could be fruitful.

## 6.1 Biological channel capacity

In communication theory there is a concept called channel capacity. It refers to the rate at which information, in the form of bits of data, can be transmitted over a communication channel. Depending on the medium of transmission and the type of channel, there are often limits on this rate – hence the term channel capacity. The engineering challenge is to come up with ways to transmit information – called the "message" – at maximum channel capacity. Additionally, the communication channel can be noisy, due to the physical act of transmitting an electrical signal over a medium, leading to the corruption of the transmitted information. These challenges led communication theorists to look for ways to efficiently and reliably transmit information. The fields of information and coding theory came out of efforts to achieve this goal. One approach was to use the fact that there is often redundancy in the information being transmitted. Therefore, the message can be transmitted by removing the redundant information – a process called "compression." The transmitted message can then be used to recreate all the information by expanding back the redundant compressed parts. Additionally, the transmitted message has to overcome the corruption caused by a noisy communication channel so that the message can be recreated accurately in its entirety. For over half a century, the fields of information and coding theory have studied this problem and produced a solid theoretical framework to help solve it. These advances have helped our digital age of cellular phones, the internet, GPS systems, etc. to become a reality.

Now think of the biological experiment as a communication channel across which we obtain information about the biological system. Just like the communication problem, the capacity of these channels is often limited and they too are subject to noisy transmission. Rapid technological advances have allowed us to increase the “channel capacity” of biological experiments to obtain large amounts of information fairly quickly. However, there is still the engineering problem of making maximum use of this capacity.

In the following sections I will discuss advances made in this thesis and potential avenues of inquiry that help maximize the use of biological channel capacity. Just like in communication theory I want to exploit redundancy in the biological information in order to achieve this goal. It is no coincidence that pooling design theory has such a strong overlap with coding theory.

## 6.2 Pooling designs and decoding

A key contribution of this thesis has been the identification of the pooling problem in the context of two high-throughput biological experiments and establishing the connection between these problems and the general theory of pooling designs. Drawing on a rich literature stream from statistics, mathematics, and computer science, the thesis brings a rigorous approach to solving the pooling problem and establishes a general framework that is easily transported to other experimental platforms.

In the context of pooling designs, the thesis helped develop the shifted transversal design (STD) pooling strategy into a practical solution to the pooling problem for biological experiments. Further, the STD strategy was made to fit the practical constraints of existing experimental set-ups. The software to create specific STD-based experimental designs and decode their results was made available.

### 6.2.1 Increasing complexity of pooling design strategies

Going forward, the goals of pooling design strategies will increase in complexity. Along with the quest for improving the efficiency and robustness of high-throughput biological experiments other practical needs will have to be met. The inability to mix a large number of items in a real experimental scenario seems to be the biggest obstacle to the success of pooling designs. In Chapter 3, I proposed the use of a block pooling design that solves the problem by cutting it up into manageable chunks – using the same pooling design repeatedly on subsets of items being tested. Although this approximation is a useful engineering solution, a more concrete mathematical formulation of the problem and a rigorous solution that emerges from this formulation is desirable.

### 6.2.2 Information equivalence metrics

To further make the case for pooling designs, it would be useful to develop a metric that can be help compare the information obtained by a pooling design relative to the "one item, one test" approach. The inherent redundancy of pooling designs allows it to not only identify the items of interest successfully but also overcome testing errors. A single run of the "one item, one test" (singleplex) strategy can identify all the items of interest provided there are not significant experimental errors. Multiple runs of the singleplex strategy may allow it to overcome testing error. But the effort, in terms of number of tests used, also increases. Therefore, it would be fruitful to have a metric of "equivalent information" that can be used to compare the pooling design to the singleplex. For example, this metric could tell us "for a biological experiment with a hit-rate of 1% and error-rate of 5% the pooling design is equivalent, in terms of sensitivity and specificity, to 2 runs of the singleplex method." Developing such a metric would require the use of simulation methods to calculate the average-case performance of the pooling design.

### 6.2.3 Computationally-intensive decoding strategies

Alongside developments in pooling design methods there need to be improvements in decoding approaches as well. Chapter 4 described two methods that represent initial steps towards a generalized, quantitative solution to the decoding problem. Inevitably these advanced methods would rely more heavily on the use of computation. An interesting new development along these lines was the introduction of Bayesian methods to solve the decoding problem (Uehara and Jimbo, 2009). The authors proposed a decoding strategy that can incorporate prior knowledge about items being tested into the solution. This knowledge is used along with the testing results to arrive at a posterior probability of each items "activity" using a Bayesian inference method. Instead of yes-no calls for items the Bayesian method produces a probabilistic estimate of each items activity. Such an approach can also be extended to ask interesting questions about second-order effects of pooling designs, such as synergitic or antagonistic effects of pooling compounds in high-throughput screens. Like most other computational approaches the Bayesian methods would shift work from experiments to analysis. However, the low cost of computing makes the investigation of these approaches worthwhile.

## 6.3 High-throughput drug screening

This thesis implemented a pooling design strategy to improve the efficiency and robustness of high-throughput drug screening. The strategy was tested with moderate success in a small proof-of-concept screen. The next step is to test the strategy on a large screen where the number of compounds and associated hit and error rates provide a more suitable proving ground for its success.

### 6.3.1 Cheminformatic approach to pooling design

Meanwhile, various kinks in the practical application of pooling designs need to be ironed out in order to increase its acceptance in the wider screening community. First, the concern about the deleterious effects of mixing chemical compounds in HTS needs to be addressed. Several approaches have been proposed to prevent interactions among compounds during pooling, such as: (i) reducing the number of compounds pooled; (ii) pooling dissimilar compounds; and (iii) systematically preventing chemical classes such as electrophiles and nucleophiles from mixing (Warrior et al., 2007; Hann et al., 1999). For adaptive pooling designs, computational methods have been designed that use chemical structure information to design optimal pools for maximizing coverage of the chemical space, while minimizing overlap (Remlinger et al., 2006). For orthogonal pooling, simulation techniques have been proposed that predict probabilities for the occurrence of synergy and blocking (antagonism) to help select the most suitable efficient pooling and retesting schemes (Xie et al., 2001). Similar methods can be developed for the pooling designs described in this thesis.

### 6.3.2 Exploiting compound interactions

Although reducing the effort required to screen a library of compounds for activity against a biological target while building robustness to testing errors are a fruitful goals pooling in high-throughput screen can offer much more. The search for new classes of multi-compound, multi-target therapeutics has become the focus of drug discovery (Borisy et al., 2003; Zimmermann et al., 2007). This implies that identifying the effect of a single compound on a target is insufficient. The cumulative effect of multiple compounds is now of interest.

Several research groups have systematically screened multi-compound mixtures to identify useful synergistic effects, with reasonable success (Feng and Shoichet, 2006). But even systematically testing pairs of compounds for thousand to million compound libraries becomes prohibitive. Pooling designs provide a way to test for multi-compound effects – whether they are synergistic, antagonistic, or additive effects – in a reliable manner. As described in the previous section, the increasing complexity of the goals of pooling designs would require advances in theory and practice. The STD strategy has an in-built parameter that controls the coverage of pairs of compounds being tested together. This can be exploited to test compounds together in a systematic yet efficient way. Next, the increased complexity of the decoding problem, when multi-compound effects are allowed, needs to be addressed. My guess is that the solution will come from the Bayesian pool decoding methods described in the pervious section.

A problem similar to the discovery of multi-compound effects is the problem of screening natural product libraries (Mishra et al., 2008; Koehn, 2008). These libraries consist of fractionated components of a naturally occurring therapeutic. The goal is to identify the components that possess the therapeutic properties and use them to develop more efficient and targeted cures. However, it is often seen in such screens that individual components do not possess the therapeutic properties by themselves. It is often a mixture of these components that actually possessed the therapeutic property in the natural state. As discussed above, the pooling design approach can address the challenge of screening natural product libraries.

### 6.3.3 Automating pooling design strategies

Finally, one of the practical challenges of pooled drug screening is the act of pooling compounds from a large library. With existing liquid-handling systems the pooling process may be time-consuming and inefficient. The tradeoff between the time taken to pool items and the corresponding savings achieved by doing so may cancel each other out. A solution to this problem could be creating a premixed library of compounds. The idea is to design a master pooling scheme that has well-tested properties guaranteeing the ability to handle a high hit and error-rate. Such a design can then be implemented once on a library of compounds and the resultant master mix can be stored for future use. This would eliminate the need to perform pooling for every time we screen this library. The master mix can be tested on every target of interest using common liquid-handling systems.

## 6.4 Gene expression microarrays

Chapter 5 described a novel smart pooling approach to run gene expression microarray experiments. The idea was to exploit the sparsity in a genes expression profile to reduce the number of microarray chips needed to obtain the expression profile. The methods hinged on choosing a set of samples that were more likely to exhibit a sparse gene expression profile. Further, the recovery of the expression profile from pooled measurements depended on an assumption of linearity in measurements. A small poorf-of-concept validated the linearity assumption but the sparsity assumption was not satisfied. The smart pooling strategy needs to be tested on a larger scale in a well-chosen system that assures the validity of the sparsity assumption. Possible candidates for such a system are biomarker discovery experiments that look for differentially expressed genes between normal and tumor samples, such as the Expression Project for Oncology (http://www.intgen.org/). Such experiments involve testing several tumor samples from a specific tissue on microarray chips and compare the results against the expression profiles of a similar number of normal samples from the same tissue in order to identify biomarker or genes that show significant differential expression. Typically, the number of samples in each category can range from a few tens to several hundreds (see Gene Expression Omnibus (http://www.ncbi.nlm.nih.gov/projects/geo/ ) for examples). The proposed smart pooling experiment would involve testing each set – normal and tumor – of samples using a pooling design strategy. The sparsity assumption is more likely to be satisfied within each group such that when the pooling design is applied to the normal sample set, say, then it is likely that there are few spikes across them thereby enabling efficient decoding. When a large number of samples are being tested for each group, a pooling design would provide valuable savings in microarray chips needed while guaranteeing success due to the satisfaction of the sparsity assumption.

## 6.5 Application of pooling designs to high-throughput biological experiments

As described in Chapter 1, there is a long history of applying group testing or pooling design strategies to solve a similar core problem – searching for a needle in a haystack, with errors – that cropped up in very diverse fields. In the past decade, the pooling approach has been applied to several new biological problems mainly due to the increasing availability of automated platforms that allow a large number of experiments to be performed simultaneously. Some of these attempts are described below.

### 6.5.1 Population genotyping for rare genetic diseases

A recent application of pooling designs that received a lot of attention was the problem of genotyping large populations to detect individuals carrying risk alleles, which indicate the presence of a rare genetic disorder (Prabhu and Pe'er, 2009; Erlich et al., 2009a,b, 2010). The central idea was to pool samples from individuals before sequencing the samples using expensive next-generation sequencing technologies to detect rare mutations. The results from the pooled samples can then be used to identify the individuals carrying these mutations. From a pooling design point of view, there are several advantages of this experimental platform. Firstly, pooling genetic material is usually much easier as the mixture is essentially homogeneous, unlike chemical compounds. This homogeneity ensures that artificial errors induced by pooling samples are limited. Second, next-generation sequencing (NGS) methods have much higher capacity that allows a large number of samples to be pooled (Mardis, 2008b,a). The equivalent of a well in HTS is a lane in NGS. Where a well in HTS could accommodate at a pool of about 10 compounds, a lane in NGS can accommodate about 1000 samples (Erlich et al., 2010). The expanded capacity makes pooling designs more attractive due to greater scope for savings in effort. Finally, rare mutations in populations are indeed rare. Good estimates of the "hit-rate" are available for rare mutations, making the task of designing pooling strategies much easier. In an application like HTS, every new target screened against a compound library has essentially an unknown hit-rate, whereas NGS does not have that problem. Further, NGS technology has much lower error-rates than HTS (Mardis, 2008b,a). All these factors put together make pooling designs for detecting rare mutations using NGS a very attractive field of study. There are several practical challenges in designing efficient decoding methods for the pooled measurements from NGS applications that still need to be addressed (Erlich et al., 2010).

### 6.5.2 Acute HIV infection testing

Another application where pooling designs have been tried is in the detection of very rare cases of HIV infection using an expensive nucleic acid amplification test (NAAT) (Kim et al., 2007; Westreich et al., 2008a). The problem is similar to the blood testing for syphilis described in Chapter 1. The methods currently in use resemble orthogonal pooling strategy described in Chapter 2. As described in Chapter 2, this pooling strategy does not allow the detection of multiple infections. Therefore, this application would benefit from the pooling design and decoding strategies described in this field.

### 6.5.3 Other applications

Several other applications display characteristics very similar to the pooling puzzle described in Chapter 1. A variety of pooling techniques have been used in these applications. But they could still benefit from the unified pooling design and decoding methods described in this thesis. Some of these applications are briefly described below:

1. Testing food for contamination – In the light of recent outbreaks of food contamination such as – salmonella, E. coli, melamine etc. there is a proposal to subject food supply chains to testing. The problem closely resembles group testing problems where false-positive detections are a major concern (Kennedy, 2008).

2. Multiplexing targets – This considers the flip side of the pooling problem in high-throughput screens, that of multiplexing targets rather than compounds (Grover et al., 2003). If both multiple targets and multiple compounds are screened via a pooling design, it opens up the possibility of discovering multi-target, multi-compound therapeutics.

3. Multiplexing siRNA screens – The goal is to compress RNA-based gene silencing screens to investigate gene-function relationships and to identify molecular targets (Martin et al., 2007; Silva et al., 2008).

4. Protein-protein interaction mapping – In order to uncover the interactome a lot of pooling design efforts similar to those used in this thesis (Chapters 3 and 4) have been applied (Jin et al., 2006, 2007; Vermeirssen et al., 2007; Xin et al., 2009). It would be beneficial if both applications borrow from each other to solidify a general framework for the pooling problem.

5. DNA-DNA hybridization experiments – Another instance of parallel development and application of pooling designs to detect target sequences, such as different virus subtypes, that are very similar and hence difficult to uncover using specific probes (Schliep et al., 2003).

6. Compressive sensing microarray – Analogous to the DNA-DNA hybridization experiments, the goal here is to develop a microarray chip that explicitly takes cross-hybridizataion into account for detecting microbes (Dai et al., 2009).

Finally, one of the more interesting challenges I faced during my thesis work was a communication problem very much in the spirit of the ones I tried to solve in my thesis. I spent a great deal of time during my thesis work identifying analogies between biological experiments and communication theory and importing ideas used in the latter to solve problems in the former. I spent an even greater deal of time on the reverse process – that of conveying the solutions to the target audience of experimentalists. This communication problem provided me the opportunity to develop my skills as a translator between two very different languages and intellectual cultures – mathematics and biology. This process of translation has given me immense pleasure as it allows me to explore different ways of conveying ideas to disparate audiences and to get them talking to one another. Many of the illustrations used in this thesis to convey the mathematics of pooling designs in the context of biological experiments are a result of this process. My hope is that this thesis will get establish an efficient and robust communication channel between these fields, so as to maximize information transfer between them, leading to the successful and widespread application of pooling designs in high-throughput biological experiments.

# Appendices

# Appendix A

# poolHiTS

## A.1 Shifted transversal design construction

Given the design parameters $q$ and $k$, the pooling matrix $M = \mathrm{STD}(n;q;k)$ can be constructed as follows. The STD has a layered construction consisting of $k$ layers of $q \times n$ boolean matrices. For all $j \in \{0,\ldots,k-1\}$, let $M_j$ be a $q \times n$ boolean matrix representing layer $L(j)$, with columns $C_{j,0},\ldots,C_{j,n-1}$. Let the circular shift operator, $\sigma_q$, be defined as, $\forall(x_1,\ldots,x_q) \in \{0,1\}^q$, $\sigma_q \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{bmatrix} = \begin{bmatrix} x_q \\ x_1 \\ \vdots \\ x_{q-1} \end{bmatrix}$ and $C_{0,0} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$. Note that $\sigma_q$ is a cyclic function and when applied $q$ times maps $\{0,1\}^q$ onto itself, $\sigma_q^s \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_q \end{bmatrix}$, $s = q$. To design a layer $L(j)$, for all $i \in \{0,\ldots,n-1\}$ construct $C_{j,i} = \sigma_q^{s(i,j)} C_{0,0}$, where,

- if $j < q : s(i,j) = \sum_{c=0}^{\Gamma} j^c \left\lfloor \frac{i}{q^c} \right\rfloor$
- if $j = q : s(i,q) = \lfloor \frac{i}{q^\Gamma} \rfloor$

The layers $L(j)$ are put together to form $M$ by, $\mathrm{STD}(n;q;k) = \bigcup_{j=0}^{k-1} L(j)$. A detailed description of the construction is available in the original STD paper (Thierry-Mieg, 2006a).

## A.2 Error-rate limit

Given $n,d$ and $E$, the STD algorithm uses a prime number $q$ $(< n)$ and a compression factor $\Gamma(= \left\lceil \frac{\log n}{\log q} \right\rceil - 1)$ which satisfy the inequality, $d\Gamma + 2\mathrm{E} \leq q$. The number of tests required for the design is, $t = q(d\Gamma + 2E + 1)$. The error-rate corrected by this design can be defined as, $e = \frac{E}{t} \times 100$. However, for a choice of $q$ that satisfies the design requirement there is a upper limit on the number of errors the corresponding design can correct, given by, $E_{\max} = \left\lfloor \frac{q-d\Gamma}{2} \right\rfloor$ (arrived at from the inequality above). This implies that the error-rate ($e$) that can be specified for a design is constrained in the following manner.

$$e \leq \frac{E_{\max}}{t} \times 100 \tag{A.1}$$

Using the expression for $t$,

$$e \leq \frac{E_{\max}}{q(d\Gamma + 2E_{\max} + 1)} \times 100 \tag{A.2}$$

Substituting for $E_{\max}$,

$$e \leq \frac{\lfloor \frac{q-d\Gamma}{2} \rfloor}{q(d\Gamma + 2\lfloor \frac{q-d\Gamma}{2} \rfloor + 1)} \times 100 \tag{A.3}$$

Using the above inequality it is possible to deduce the maximum error-rate that STD can handle for a particular $n$-compound library with at most $d$ active compounds and an assay with a limit on mixing not more than $m$ compounds in a test. However, in most cases, the numerical value of the maximum error rate can only be calculated by executing (Algorithm 3.3) completely. For the simplest case of $d = 0$, since $q$ is a prime number and hence odd (except for $q = 2$), the term $\lfloor \frac{q-d\Gamma}{2} \rfloor$ is reduced to $\lfloor \frac{q}{2} \rfloor$ and is always equal to $\frac{q-1}{2}$. Simplifying the expression for the error limit, we get, $e \leq \frac{q-1}{2q^2} \times 100$. Whereas, for $q = 2$ (the only even prime number), the value of the limit is, $e \leq \frac{1}{6} \times 100 \sim 16.67\%$. This is the maximum error-rate that STD-based designs can guarantee correction for.

## A.3 STD correction

In the case where the choice of the design parameters $q$ and $k$ ($= d\Gamma + 2E_{\min} + 1$) are such that, $k = q + 1$, then the construction of the last layer ($j = k - 1, k = q + 1$) will use $s(i,q) = \lfloor \frac{i}{q^\Gamma} \rfloor$ (see Appendix A.1). Since each layer $L(j)$ consists of $q$ rows and $n$ columns and these are constructed by applying the circular shift operator ($\sigma_q$), $s(i,q)$ times, on $C_{0,0}$ where $i \in \{0,\ldots,n-1\}$. By definition $q^\Gamma < n \leq q^{\Gamma+1}$, $s(i,q)$ takes the values $\{0,1,\ldots,q-1\}$ sequentially. As shown in the illustration below, for columns $i \in \{0,\ldots,q^\Gamma - 1\}$, $s(i,q) = 0$ and hence they all resemble $C_{0,0}$ exactly. The next $q^\Gamma$ columns have $s(i,q) = 1$ and resemble $C_{0,0}$ once shifted and so on.

| row/col. | 0 | … | $q^\Gamma - 1$ | $q^\Gamma$ | … | $2q^\Gamma - 1$ | … | … | … | $n-1$ | | … | $(q-1)q^\Gamma$ | … | $q^{\Gamma+1} - 1$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | … | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | … | 0 | 0 | 0 |
| $\vdots$ | | $\vdots$ | | | | $\vdots$ | | | | | | | | $\vdots$ | |
| $\lceil \frac{n-1}{q^\Gamma} \rceil$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | | … | 0 | 0 | 0 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | … | 0 | 0 | 0 |
| $\vdots$ | | $\vdots$ | | | | $\vdots$ | | | | | | | | $\vdots$ | |
| $q-1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $q \times n$ | … | 1 | 1 | 1 |

When the library size $n$ is such that $\left\lfloor \frac{n-1}{q^\Gamma} \right\rfloor < q-1$, only rows from 0 to $\left\lfloor \frac{n-1}{q^\Gamma} \right\rfloor$ have compounds in them and the rest of the rows in the layer are empty. It follows that in this layer the number of compounds mixed in each test is $q^\Gamma$. Usually, the number of compounds per test is $\lceil \frac{n}{q} \rceil$ and in this case we know that $\frac{n}{q} < q^\Gamma$. Hence, it is essential to make this correction in order to correctly implement the mixing constraint.

For example, In Table 2 of the original STD paper (Thierry-Mieg, 2006a), there is a case where $n = 10000$, $d = 3$ and $E = 2$ with the optimal choice of $q = 13$ and $k = 14$. In this case $k = q + 1$, so the actual number of tests is $t = 174$, and not 182 ($= 13 \times 14$) as stated in the paper, and the maximum number of compounds being mixed is actually 2197 rather than 769.

## A.4 Corrected STD

A modification is required for the special case where the choice of design parameters ($q$ and $k$) is such that $k = q + 1$ and $\left\lfloor \frac{n-1}{q^\Gamma} \right\rfloor < q - 1$. The construction remains essentially the same as Appendix A.1 with the only modification being the removal of the unnecessary tests with no compounds in them. As shown in Appendix A.3, the last $q - (\lfloor \frac{n-1}{q^\Gamma} \rfloor + 1)$ rows of $M$ need to be removed. It should be noted that the maximum number of compounds being mixed in a test is $q^\Gamma$. The MATLAB implementation of this construction is provided (see Appendix A.7).

## A.5 Choice of block size

The problem of block size selection is one of *sampling without replacement*. The compound library ($n$) with at most $d$ active compounds has to be divided into $B$ blocks of size $n_B$ and the probability of dividing these active compounds into different blocks needs to be determined. The hypergeometric distribution provides a way to estimate this probability.

Thus, the probability of finding $i$ active compounds in a block of size $n_B$ selected without replacement from the compound library of size $n$ that contains $d$ active compounds is given as,

$$P(i) = \frac{\begin{pmatrix} d \\ i \end{pmatrix} \begin{pmatrix} n-d \\ n_B - i \end{pmatrix}}{\begin{pmatrix} n \\ n_B \end{pmatrix}} \tag{A.4}$$

The identification of the experiment design parameter $d_B$ for the block requires the specification of a probability ($p_B$) representing a *confidence* in finding at most $d_B$ active compounds in a block of size $n_B$, which is the sum of all the individual probabilities of finding exactly $i = \{0, 1, \ldots, d_B\}$ active compounds out of $n_B$ compounds. Therefore the condition for calculating $d_B$ is, $\sum_{i=0}^{d_B} P(i) \geq p_B$, thus arriving at Equation 3.2. It is useful to note that other design parameters of error-rate ($e$) and mixing constraint ($m$) are so defined, as to be the same for individual blocks and the whole library. The MATLAB implementation of this equation is provided (see Appendix C.1).

## A.6 poolHiTS decoder

```
% poolHiTS_decode.m
%
% pooHiTS : A Shifted Transversal Design based pooling strategy for
% High-Throughput Drug Screening
%
% Decodes results (binary) of an expt. T using M from poolHiTS(n,d,e,m)
%
% Written by: Raghu Kainkaryam, University of Michigan
% Created: December 2007
% Modified : March 2008

function List=poolHiTS_decode(n,d,E,M,T)

[rm cm]=size(M);
Mneg=[];
Mpos=[];
Tneg=find(T==0);
```

```
Tpos=find(T==1);

% Search for Inactive Compounds
% All compounds present in (E+1) negative tests
% are tagged inactive

for j=1:cm
    if sum(M(Tneg,j)) ≥ E+1
        Mneg=[Mneg;j];
    end
end

% Search for Active Compounds
% All compounds present in (E+1) posiyive tests where
% all other compounds have been tagged inactive
% are tagged active

for x=1:cm
    tempM=M(Tpos,:);

    if isempty(find(Mneg==x))
        tempM(:,[Mneg;x])=[];
    else
        tempM(:,Mneg)=[];
    end

    check=find(M(Tpos,x)==1);
    sumM=sum(tempM,2);

    if length(find(sumM(check)==0)) ≥ E+1

        Mpos=[Mpos;x];
    end
end

Pos=zeros(cm,1);
Neg=ones(cm,1);
Pos(Mpos)=1;
Neg(Mneg)=0;
List=(Pos+Neg)/2;

% disp('Active Compound Ids:')
 Active=find(List==1);
```

```
%
% disp('Number of Inctive Compounds :')
 Inactive=find(List==0);
% length(Inactive);
%
% disp('Untagged Compound Ids :')
 UnTag=find(List==0.5);

if length(Active) > d, disp('False Positives detected'),end

% End of Function
```

## A.7 poolHiTS construction

This function is required for Appendix A.8.

```
% poolHiTS_constr.m
%
% pooHiTS : A Shifted Transversal Design based pooling strategy for
% High-Throughput Drug Screening
%
% Constructs poolHiTS(n,d,e,m) given [q Gamma k]
%
% Written by: Raghu Kainkaryam, University of Michigan
% Created: December 2007
% Modified : March 2008

function M=poolHiTS_const(n,d,e,m,q,Gamma,k)

flag=0;
% Check if STD Correction is needed
if k == q+1 && floor((n-1)/q^Gamma) < q-1 && q^Gamma≤m
    flag=1;
end

M=zeros(k*q,n);
s=zeros(k,n);

C00=zeros(q,1);
C00(1)=1;

for j=1:k
```

```
    for i=1:n
        if j-1 < q
            for c=0:Gamma
                s(j,i)=s(j,i)+ ((j-1)^c)*floor((i-1)/q^c);
            end
        elseif j-1==q
            s(j,i)=floor((i-1)/q^Gamma);
        end

        M((j-1)*q+1:j*q,i)=circshift(C00,s(j,i));
    end
end

if flag==1
    M((k-1)*q+(floor(n/q^Gamma)+1)+1:k*q,:)=[];
end

% End of Function
```

### A.7.1 poolHiTS algorithm

```
% poolHiTS.m
%
% pooHiTS : A Shifted Transversal Design based pooling strategy for
% High-Throughput Drug Screening

function poolHiTS_par=poolHiTS(n,d,e,m,pB)

List=[]; % List of all parameters

out=poolHiTS_design(n,d,e,m);
List=[List;horzcat(1,out,out(end))];

s=1; % counter

for i=1:d-1 % loop over d

    if pB ==1, % bSTD does not work for cf=1
        break;
    end

    for j=s:n   % loop over n
        if poolHiTS_hypgeo(n,d,j,i) < pB % Faster way to find blocks that work
            dB=i;
            for nB=s:j-1
                B=ceil(n/nB);
                out=poolHiTS_design(nB,dB,e,m);
                if isnan(out(end))≠1
                    List=[List;horzcat(B,out,B*out(end))];
                end
            end
            s=j;
            break;
        end
    end
end

% Find the design which uses least tests
Opt=find(List(:,end)==min(List(:,end)))
% disp('B nB dB e m q Gamma k Emin e_act e_max m_act tB t')
poolHiTS_par=List(Opt,:);

% If there is more than one design with min. tests, choose one with max.
% e_act
if length(Opt) >1
    Opt=find(poolHiTS_par(:,10)==max(poolHiTS_par(:,10)));
    % If there is more than one design with max. e_act, choose one closest to n
    if length(Opt) >1
        [clo Opt]=min(diag(poolHiTS_par(:,1)'*poolHiTS_par(:,2)));
    end
    poolHiTS_par=poolHiTS_par(Opt,:);
end

% End of Function
```

## A.8 poolHiTS example

Requires Appendices A.6, A.7, A.7.1, A.9, and A.10.

```
% poolHiTS-main.m
%
```

```
% pooHiTS : A Shifted Transversal Design based pooling strategy for
% High-Throughput Drug Screening
%
% (Error-rate, Mixing Constraint and Block Design included)
% (Example with False Negative Errors)

clear
clc

tic % timer
format short g

n=10000; % Total No. of Compounds
d=3;     % Max. No. of Active Compounds expected
e=0.01;  % Error-rate : fraction of False Results expected
m=10; % Mixing Constraint : Max. No. of Compounds per well
pB=0.99; % Confidence : probability of covering for actives in block

disp('Optimal STD Design Parameters chosen by poolHiTS');
disp('B nB dB e m q Gamma k Emin e_act e_max m_act tB t');
poolHiTS_par=poolHiTS(n,d,e,m,pB)

% poolHiTS Construction of 0-1 matrix M
M=poolHiTS_const(poolHiTS_par(2),poolHiTS_par(3),e,m,poolHiTS_par(6), ...
poolHiTS_par(7),poolHiTS_par(8));
B=poolHiTS_par(1);
disp('Use the binary matrix M'), B, disp('times over the entire library')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Run a test experiment
% Let compound nos. 1000, 4000 and 7000 be Active
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nat=zeros(n,1);
Nat([1000 4000 7000])=1;

% Taking care of blocks
B=poolHiTS_par(1);
nB=poolHiTS_par(2);
dB=poolHiTS_par(3);
Emin=poolHiTS_par(9);
right=floor(n/B);
extra=n-right*B;
```

```
count=1;
Status=[];
HouseKeep=[];

% Since the block size and number of compounds in a block might not add up
% to the actual library size, we need to distribute the blanks evenly
% across the all blocks. The blanks are used for housekeeping -- to check
% if the decoding is correct.
for block=1:B

    NatBlock=[];
    if block ≤ extra
        NatBlock=Nat(count:count+right);
        count=count+right+1;
    else
        NatBlock=Nat(count:count+right-1);
        count=count+right;
    end

    hkzeros=nB-length(NatBlock);
    NatBlock=[NatBlock;zeros(hkzeros,1)];

    % In-Silico Experiment
    T=M*NatBlock;
    T(find(T>0))=1; % Converting the results to binary -- positive/negative

    % Introduce Error (STD designed for Emin=iSTD(9))
    errT=randperm(length(T));

    % Since Emin=1, introduce Emin+1 random errors (EXtra errors)
    for bar=1:Emin
        if T(errT(bar))==1
            T(errT(bar))=0;
        else
            T(errT(bar))=1;
        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Decode & Store results of in-silico experiment with error)%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    Find=poolHiTS_decode(nB,dB,Emin,M,T);
    Status=[Status;Find(1:end-hkzeros)];
    HouseKeep=[HouseKeep;Find(end-hkzeros+1:end)];

end

Compare=horzcat(Nat,Status);

% Status of all compounds used as controls
Contrast=horzcat(zeros(length(HouseKeep),1),HouseKeep)

plot(1:n,Nat,'ob',1:n,Status,'.r')

% Time taken for whole process (in minutes)
time_mins=toc/60
% End of Main
```

## A.9 poolHiTS design

This function is required for Appendix A.8

```
% poolHiTS_design.m
%
% pooHiTS : A Shifted Transversal Design based pooling strategy for
% High-Throughput Drug Screening
%
% Designs poolHiTS(n,d,e,m)
% Returns [n d e m q G k Emin e_act e_max m_act t]

function Send=poolHiTS_design(n,d,e,m)

prime_set=primes(n); % List of all prime nos. upto n

Choice=[];

for foo=1:length(prime_set)

    q=prime_set(foo);

    if rem(log(n),log(q)) == 0
        Gamma=log(n)/log(q)-1;
    else
        Gamma=floor(log(n)/log(q));
    end

    if d*Gamma ≤q

        Emax=floor((q - d*Gamma)/2); % Max. No. Errors correctable

        e_max=Emax/(q*(d*Gamma+2*Emax+1)); % Max. Error-rate handled

        if e_max ≥ e

            for Emin=0:Emax

                if Emin/(q*(d*Gamma+2*Emin+1)) ≥ e
                    k=d*Gamma+2*Emin+1;

                    if k < q+1 && ceil(n/q) ≤m
                        Choice=[Choice;horzcat(n,d,e,m,q,Gamma,k,Emin, ...
                        Emin/(k*q),e_max,ceil(n/q),k*q)];
                        break

                    elseif k == q+1 && n == q^(Gamma+1) && q^Gamma≤m
                        Choice=[Choice;horzcat(n,d,e,m,q,Gamma,k,Emin, ...
                        Emin/(k*q),e_max,q^Gamma,k*q)];
                        break

                    elseif k == q+1 && floor((n-1)/q^Gamma) < q-1 ...
                    && q^Gamma≤m % STD Correction
                        Choice=[Choice;horzcat(n,d,e,m,q,Gamma,k,Emin, ...
                        Emin/(k*q),e_max,q^Gamma,(k-1)*q+...
                        floor((n-1)/q^Gamma)+1)];
                        break
                    end
                end
            end
        end
    end
end

if isempty(Choice)
    Send=NaN*ones(1,12);
    return;
end
```

```
[low loc]=min(Choice(:,end));

Send=Choice(loc,:);

% End of Function
```

## A.10 Block pooling success probability

This function is required for Appendix A.8

```
% poolHiTS_hypgeo.m
%
% pooHiTS : A Shifted Transversal Design based pooling strategy for
% High-Throughput Drug Screening
%
% Calculates the probability for given n, d, nB, dB

function P=poolHiTS_hypgeo(n,d,nB,dB)

F=[];

for i=0:dB
    F(i+1)=exp(gammaln(d+1)+gammaln(n-d+1)+gammaln(nB+1)+gammaln(n-nB+1) ...
    -gammaln(i+1)-gammaln(d-i+1)-gammaln(nB-i+1) ...
    -gammaln(n-d-nB+i+1)-gammaln(n+1));
end

P=sum(F);

% End of Function
```

# Appendix B
# poolMC

## B.1 poolMC design

```
    % pmc_design: poolMC design function takes in number of samples (n),
 %    number of chips (m), testing redundancy (d), and mixing constraint (r)
 % and produces the pooling design, which is a m x n binary matrix M,
 % with d ones in each column and at most r ones in each row.
%
% Usage: [M list] = pmc_design(n,m,d,r);
%
% M - m x n binary matrix with d 1's in each col. and at most r 1's in each
% row
%
% list - a sparse representation of M. list is a n x d matrix with the row
% positions for each sample.
%
% n - Number of samples being tested
%
% m - Number of rows to be used (Look at Supplementary matrials for a guide
% to choose this value)
%
% d - Number of times each sample is tested in the pooling design (Look at
% Supplementary matrials for a guide to choose this value)
%
% r - Maximum number of samples pooled on a single chip
% (To prevent an endless loop, choose a value ≥ n*d/m)

function [M list]= pmc_design(n,m,d,r)

list=zeros(n,d);

q=1;
```

```
% Unique row and column loop
while q≠0
    p=r+1;

    % Row weight loop
    while p>r
        z=0;

        % Empty row loop
        while z==0
            M=zeros(m,n);

            % Randomly assign 'd' ones to each columns of the matrix
            for i = 1:n
                while true
                    l = randint(1, d, [1 m]);
                    if length(unique(l)) == d
                        break;
                    end
                end
                list(i,:)=l;
            end

            % Check if any row is empty
            for j=1:n
                M(list(j,:),j)=1;
            end
            z=min(sum(M,1));
        end

        % Check for mixing constraint
        p=max(sum(M,2));
    end

    % Check if each column is unique
    for foo=1:n-1
        for bar=foo+1:n
            if nnz(M(:,foo)-M(:,bar))==0
                q=42;
            end
        end
    end
    % Check if each row is unique
    for foo=1:m-1
        for bar=foo+1:m
            if nnz(M(foo,:)-M(bar,:))==0
                q=42;
            end
        end
    end

    % Reset process if rows and cols are not unique
    if q≠42
        q=0;
    end
end

% End of Script
```

## B.2 poolMC decoder

```
    % pmc_decode: poolMC decoding script using l1-magic to
% solve quadratically constrained l1 minimization:
%
% min ||x||_1   s.t.  ||Phi*x - y||_2 ≤ \epsilon
%
% Phi - poolMC pooling design with m rows and n cols
% scaled such that each row sums to 1
%
% y - pooled measurements
%
% Dependencies: l1-magic
% Need to download l1-magic from http://www.acm.caltech.edu/l1magic

clear
clc
format short g
tic

% place l1magic in the same directory as this .m file
addpath(fullfile(pwd, '/l1magic/Optimization'));
```

```
% Load the binary matrix generated by pmc_design
% Below is the design used in the paper -- n=15, m=12, d=4, r=3
% M=[ 0     0     0     0     0     0     0     0     0     0     0     1
1     0     0
%     1     0     1     0     0     0     0     0     0     0     0     0
0     0     0
%     0     0     0     1     0     0     0     1     1     0     0     0
0     0     0
%     0     0     0     0     1     0     0     0     0     1     0     0
0     0     1
%     0     0     0     0     0     0     0     0     0     0     0     0
0     1     1
%     0     0     1     0     0     1     1     0     0     0     0     0
0     0     0
%     0     0     0     0     0     0     0     0     1     1     0     0
0     1     0
%     1     0     0     0     1     0     0     0     0     0     0     0
0     0     0
%     0     0     0     1     0     0     1     0     0     0     1     0
0     0     0
%     0     1     0     0     0     0     0     1     0     0     0     0
0     0     0
%     0     1     0     0     0     1     0     0     0     0     0     0
1     0     0
%     0     0     0     0     0     0     0     0     0     0     1     1
0     0     0];
M=load('pmc_design.txt');
[rm cm]=size(M);

% Scale down the matrix, since the experiment is carried out at fractional
% concentrations
Phi=M;
scale=sum(M,2);
for i=1:rm
    Phi(i,:)=M(i,:)./scale(i);
end

% Load the multiplex measurements.
Data=load('multiplex.txt');

[rdat cdat]=size(Data);

% Parameter to control the sensitivity of the decoded result.
% Increase its value to get a less sensitive result (more zeros in the
% solution)
epsilon = 1;

Sav=[];

% Repeate for each gene in the system
for i=1:rdat

    % Decoding on non-logged data (change if data is already non-logged)
    y=2.^Data(i,:)';

    % Estimate the median from the measurements
    sub=median(y);

    % Subtract the median from the results to induce sparsity
    ystar=y-sub;

    % Use the conjugate gradient method to initialize l1 minimization
    cgfun = @(z) Phi*Phi'*z;
    sol = cgsolve(cgfun,ystar, 1e-4, 1000, 0);
    % A' * sol is a solution to A * x = b
    x0 = Phi'*sol;

    % Run LP
    xstar = l1qc_logbarrier(x0, Phi, [], ystar, epsilon, 1e-4);
    xhat=xstar+sub;

    % Median correction for NaN or negative values
    foo=[];
    foo=[find(isnan(xhat)==1);find(xhat<0)];
    foo=unique(foo);

    if isempty(foo)==0
        % If the number of such values in xhat is less than half
        % set them to the median value of xhat
        if length(foo) < floor(length(xhat)/2)
            xhat(foo)=median(xhat);

        % If the number of such values in xhat is more than half
        % set them to a low value
        else
            xhat(foo)=2; % log2(2)=1
        end
    end

    Sav=vertcat(Sav,xhat');
end

save('Multiplex-decoded-15samples-21505genes.txt','Sav','-ASCII');

time_mins=toc/60

% End of Script
```

## B.3 poolMC example

```
% pmc_example: Example of poolMC pooling and decoding in action

clear
clc
format short g
tic

% place l1magic in the same directory as this .m file
addpath(fullfile(pwd, '/l1magic/Optimization'));

% Load the binary matrix generated by pmc_design
M=[ 0      0      0      0      0      0      0      0      0      0      0      1
1      0      0
    1      0      1      0      0      0      0      0      0      0      0      0
0      0      0
    0      0      0      1      0      0      0      1      1      0      0      0
0      0      0
    0      0      0      0      1      0      0      0      0      1      0      0
0      0      1
    0      0      0      0      0      0      0      0      0      0      0      0
0      1      1
    0      0      1      0      0      1      1      0      0      0      0      0
0      0      0
    0      0      0      0      0      0      0      0      1      1      0      0
0      1      0
    1      0      0      0      1      0      0      0      0      0      0      0
0      0      0
```

```
    0     0     0     1     0     0     1     0     0     0     1     0
0     0     0
    0     1     0     0     0     0     0     1     0     0     0     0
0     0     0
    0     1     0     0     0     1     0     0     0     0     0     0
1     0     0
    0     0     0     0     0     0     0     0     0     0     1     1
0     0     0];

[rm cm]=size(M);

% Scale down the matrix, since the experiment is carried out at fractional
% concentrations
Phi=M;
scale=sum(M,2);
for i=1:rm
    Phi(i,:)=M(i,:)./scale(i);
end

% Load monoplex data for gene AT5G54740 (#20282)
x=[ 10.178
    8.885
    10.935
    11.537
    9.717
    104.94
    10.382
    10.602
    8.1999
    10.588
    9.2549
    9.6879
    9.717
    8.7789
    9.3176];

% Parameter to control the sensitivity of the decoded result.
% Increase its value to get a less sensitive result (more zeros in the
% solution)
epsilon = 1;

% Simulates a noise-free multiplex experiment
y=Phi*x;
```

```
% Estimate the median from the measurements
sub=median(y);

% Subtract the median from the results to induce sparsity
ystar=y-sub;

% Use the conjugate gradient method to initialize l1 minimization
cgfun = @(z) Phi*Phi'*z;
sol = cgsolve(cgfun,ystar, 1e-4, 1000, 0);
% A' * sol is a solution to A * x = b
x0 = Phi'*sol;

% Run LP
xstar = l1qc_logbarrier(x0, Phi, [], ystar, epsilon, 1e-4);
xhat=xstar+sub;

% Median correction for NaN or negative values
foo=[];
foo=[find(isnan(xhat)==1);find(xhat<0)];
foo=unique(foo);

if isempty(foo)==0
    % If the number of such values in xhat is less than half
    % set them to the median value of xhat
    if length(foo) < floor(length(xhat)/2)
        xhat(foo)=median(xhat);

    % If the number of such values in xhat is more than half
    % set them to a low value
    else
        xhat(foo)=2; % log2=1
    end
end

% plot the results and compare monoplex to multiplex
figure('Color',[1 1 1]);
[s l]=sort(x,'descend');
plot(s,'sk');
hold on;plot(xhat(l),'.-b');
xlabel('Samples (sorted)','fontsize',12);
ylabel('Gene expression','fontsize',12);
legend('Actual','Decoded','Location','NorthEast');
```

```
% Time taken
time_mins=toc/60

% End of Script
```

# Appendix C
# QUAPO

## C.1 Restricted isometry property of Shifted Transversal Designs

Deterministic constructions of compressive sensing matrices satisfying the RIP property, described in Chapter 4 were introduced by DeVore (2007) and are identical to a construction called shifted transversal designs (STD), described by Thierry-Mieg (2006a). Since the STD construction is easier to describe, they are presented below.

For a given number of compounds $n$, maximum expected active $k$ and maximum that can be pooled $r$, pick a prime $q$ so that

$$k-1 \leq \frac{q}{\Gamma}$$

where $\Gamma = \min\{\gamma | q^{\gamma+1} \geq n\}$, and

$$\lceil \frac{n}{q} \rceil \leq r$$

Let,

$$p(x) = a_0 + a_1 x + \ldots a_\Gamma x^\Gamma$$

be a polynomial of degree $\Gamma$ over $GF(q)$.

Let us order the tuples $(j,k) \in GF(q) \times GF(q)$ in lexicographic order and let $Q$, a vector of length $q^2$, denote the graph of the polynomial $p$ over $GF(q)$. That is,

$$Q(j,k) = 1 \quad \text{if } p(j) = k \text{ and } 0 \text{ otherwise.}$$

The STD matrix is simply the graphs, represented as columns, of $n$ polynomials of degree $\Gamma$ over $GF(q)$. For the construction details of STD see Appendix A.1.

## C.2 QUAPO decoder

```
% quapo_decode.m
%
% Quantitative Analysis of Pooling for High-Throughput Screening (Main File)
% (using Sparse Random Matrices)
%
% Requires l1-magic package

path(path, './l1magic/Optimization');

% Load pooling design
M=load('pooling_design.txt');
[rm cm]=size(M);

% Load % Inhibition measurements (single vector) from pooling experiment
mplex=load('pooling_data.txt');

% Convert %Inhibition to activity
y=mplex./(100-mplex);

% Set sensitivity for decoding
epsilon=1;
tol=1e-4;
max_iter=1000;

% Use the conjugate gradient method to initialize l1 minimization
cgfun = @(z) M*M'*z;
sol = cgsolve(cgfun,y, tol, max_iter, 0);
% A' * sol is a solution to A * x = b
x0 = M'*sol;

% Run LP
xstar = l1qc_logbarrier(x0, M, [], y, epsilon, tol);

% Convert activity pack to %Inhibition value
perI=100*xstar./(1+xstar);

% End of script
```

# Bibliography

Balding, D. 1997. The design of pooling experiments for screening a clone map, *Fungal Genetics and Biology*, 21(3), 302–307, URL `http://dx.doi.org/10.1006/fgbi.1997.0985`.

Bar-Lev, Shaul K., Wolfgang Stadje, and Frank. 2006. Group testing procedures with incomplete identification and unreliable testing results, *Applied Stochastic Models in Business and Industry*, 22(3), 281–296, URL `http://dx.doi.org/10.1002/asmb.616`.

Berinde, R., A.C. Gilbert, P. Indyk, H. Karloff, and M.J. Strauss. 2008. Combining geometry and combinatorics: A unified approach to sparse signal recovery, in *46th Annual Allerton Conference on Communication, Control, and Computing*, 798–805.

Berinde, R. and P. Indyk. 2008. Sparse recovery using sparse random matrices, *MIT-CSAIL Technical Report*.

Berman, David M. and Alfred G. Gilman. 1998. Mammalian rgs proteins: Barbarians at the gate, *Journal of Biological Chemistry*, 273(3), 1269–1272, URL `http://dx.doi.org/10.1074/jbc.273.3.1269`.

Blazer, Levi L. and Richard R. Neubig. 2008. Small molecule protein-protein interaction inhibitors as cns therapeutic agents: Current progress and future hurdles, *Neuropsychopharmacology*, 34(1), 126–141, URL `http://dx.doi.org/10.1038/npp.2008.151`.

Blazer, Levi L., David L. Roman, Martha J. Larsen, Benjamin M. Greedy, Stephen M. Husbands, and Richard R. Neubig. 2010. Reversible inhibition of rgs proteins with small molecules as a mechanism for potentiating g-protein signaling, *submitted*.

Borisy, A. A., P. J. Elliott, N. W. Hurst, M. S. Lee, J. Lehar, E. R. Price, G. Serbedzija, G. R. Zimmermann, M. A. Foley, B. R. Stockwell, and C. T. Keith. 2003. Systematic discovery of multicomponent therapeutics., *Proc Natl Acad Sci U S A*, 100(13), 7977–7982, URL `http://dx.doi.org/10.1073/pnas.1337088100`.

Bruno, W., E. Knill, D. Balding, D. Bruce, N. Doggett, W. Sawhill, R. Stallings, C. Whittaker, and D. Torney. 1995. Efficient pooling designs for library screening, *Genomics*, 26(1), 21–30, URL `http://dx.doi.org/10.1016/0888-7543(95)80078-Z`.

Brusco, Michael and Stephanie Stahl. 2005. *Branch-and-Bound Applications in Combinatorial Data Analysis (Statistics and Computing)*, Springer, 1 edn., URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0387250379`.

Candès, E. J. and J. Romberg. 2005. *$\ell_1$-MAGIC: Recovery of Sparse Signals via Convex Programming*. Available at `http://www.acm.caltech.edu/l1magic`.

Candes, E. J. and M. B. Wakin. 2008. An introduction to compressive sampling, *Signal Processing Magazine, IEEE*, 25(2), 21–30, URL `http://dx.doi.org/10.1109/MSP.2007.914731`.

Candès, Emmanuel J., Justin K. Romberg, and Terence Tao. 2006. Stable signal recovery from incomplete and inaccurate measurements, *Communications on Pure and Applied Mathematics*, 59(8), 1207–1223, URL `http://dx.doi.org/10.1002/cpa.20124`.

Chung, Thomas D. Y. 1998. Screen compounds singly: Why muck it up?, *Journal of Biomolecular Screening*, 3(3), 171–173, URL `http://jbx.sagepub.com`.

Dai, Manhong, Pinglang Wang, Andrew D. Boyd, Georgi Kostov, Brian Athey, Edward G. Jones, William E. Bunney, Richard M. Myers, Terry P. Speed, Huda Akil, Stanley J. Watson, and Fan Meng. 2005. Evolving gene/transcript definitions significantly alter the interpretation of GeneChip data, *Nucl. Acids Res.*, 33(20), e175–, URL `http://nar.oxfordjournals.org/cgi/content/abstract/33/20/e175`.

Dai, W., M. A. Sheikh, O. Milenkovic, and R. G. Baraniuk. 2009. Compressive sensing dna microarrays., *EURASIP journal on bioinformatics & systems biology*, URL `http://view.ncbi.nlm.nih.gov/pubmed/19158952`.

De Bonis, Annalisa, Leszek Gasieniec, and Ugo Vaccaro. 2005. Optimal two-stage algorithms for group testing problems, *SIAM J. Comput.*, 34(5), 1253–1270, URL `http://dx.doi.org/10.1137/S0097539703428002`.

Devlin, James J., Amy Liang, Lan Trinh, Mark A. Polokoff, David Senator, Wei Zheng, Jason Kondracki, Peter J. Kretschmer, John Morser, Samuel E. Lipson, Richard Spann, John A. Loughlin, Kathryn V. Dunn, and Michael M. Morrissey. 1996. High capacity screening of pooled compounds: Identification of the active compound without re-assay of pool members, *Drug Development Research*, 37(2), 80–85, URL `http://dx.doi.org/10.1002/(SICI)1098-2299(199602)37:2%3C80::AID-DDR3%3E3.0.CO;2-H`.

DeVore, Ronald A. 2007. Deterministic constructions of compressed sensing matrices, *J. Complex.*, 23(4-6), 918–925, URL `http://dx.doi.org/10.1016/j.jco.2007.04.002`.

Dinneny, Jose R., Terri A. Long, Jean Y. Wang, Jee W. Jung, Daniel Mace, Solomon Pointer, Christa Barron, Siobhan M. Brady, John Schiefelbein, and Philip N. Benfey. 2008. Cell identity mediates the response of arabidopsis roots to abiotic stress, *Science*, 320(5878), 1153795–945, URL `http://dx.doi.org/10.1126/science.1153795`.

Donoho, David. 2006. Compressed sensing, *IEEE Transactions on Information Theory*, 52(4), 1289 – 1306.

Dorfman, Robert. 1943. The detection of defective members of large populations, *The Annals of Mathematical Statistics*, 14(4), 436–440.

Dove, Alan. 1999. Drug screening[mdash]beyond the bottleneck, *Nat Biotech*, 17(9), 859–863, URL `http://dx.doi.org/10.1038/12845`.

Du, Ding-Zhu and Frank K. Hwang. 2000. *Combinatorial Group Testing and Its Applications (Applied Mathematics)*, World Scientific Publishing Company.

Du, Dingzhu and Frank Hwang. 2006. *Pooling Designs And Nonadaptive Group Testing: Important Tools for DNA Sequencing (Series on Applied Mathematics)*, World Scientific Publishing Company, URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20%&amp;path=ASIN/9812568220`.

Eppstein, David, Michael T. Goodrich, and Daniel S. Hirschberg. 2005. Improved combinatorial group testing for real-world problem sizes, in *Algorithms and Data Structures*, Springer Berlin / Heidelberg, vol. 3608/2005, 86–98, URL `http://dx.doi.org/10.1007/11534273_9`.

Erlich, Yaniv, Kenneth Chang, Assaf Gordon, Roy Ronen, Oron Navon, Michelle Rooks, and Gregory J. Hannon. 2009a. Dna sudoku–harnessing high-throughput sequencing for multiplexed specimen analysis., *Genome research*, 19(7), 1243–1253, URL `http://dx.doi.org/10.1101/gr.092957.109`.

Erlich, Yaniv, Assaf Gordon, Michael Brand, Gregory J. Hannon, and Partha P. Mitra. 2010. Compressed genotyping, *IEEE Transactions on Information Theory*, 56(2), 706–723, URL `http://dx.doi.org/10.1109/TIT.2009.2037043`.

Erlich, Yaniv, Noam Shental, Amnon Amir, and Or Zuk. 2009b. Compressed sensing approach for high throughput carrier screen, in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 539–544, URL `http://dx.doi.org/10.1109/ALLERTON.2009.5394904`.

Feng, BY and BK Shoichet. 2006. Synergy and antagonism of promiscuous inhibition in multiple-compound mixtures., *Journal of medicinal chemistry*, 49(7), 2151–2154, URL `http://dx.doi.org/10.1021/jm060029z`.

Ferrand, Sandrine, Andres Schmid, Caroline Engeloch, and Fraser J. Glickman. 2005. Statistical evaluation of a self-deconvoluting matrix strategy for high-throughput screening of the cxcr3 receptor, *ASSAY and Drug Development Technologies*, 3(4), 413–424, URL `http://dx.doi.org/10.1089/adt.2005.3.413`.

Garnier, Martine, Paola F. Zaratin, Giovanna Ficalora, Maurizio Valente, Laura Fontanella, Man-Hee Rhee, Kendall J. Blumer, and Mark A. Scheideler. 2003. Up-regulation of regulator of g protein signaling 4 expression in a model of neuropathic pain and insensitivity to morphine, *Journal of Pharmacology and Experimental Therapeutics*, 304(3), 1299–1306, URL `http://dx.doi.org/10.1124/jpet.102.043471`.

Grover, G. Scott, Benjamin A. Turner, Christian N. Parker, Jannika Meier, Deepak S. Lala, and Paul H. Lee. 2003. Multiplexing nuclear receptors for agonist identification in a cell-based reporter gene high-throughput screen, *J Biomol Screen*, 8(3), 239–246, URL `http://dx.doi.org/10.1177/1087057103008003001`.

Hackstadt, Amber and Ann Hess. 2009. Filtering for increased power for microarray data analysis, *BMC Bioinformatics*, 10(1), 11+, URL `http://dx.doi.org/10.1186/1471-2105-10-11`.

Hann, M, B Hudson, X Lewell, R Lifely, L Miller, and N Ramsden. 1999. Strategic pooling of compounds for high-throughput screening, *J. Chem. Inf. Comput. Sci.*, 39(5), 897–902, URL `http://dx.doi.org/10.1021/ci990423o`.

Houghten, R. 2000. Drug discovery and vaccine development using mixture-based synthetic combinatorial libraries, *Drug Discovery Today*, 5(7), 276–285, URL `http://dx.doi.org/10.1016/S1359-6446(00)01513-0`.

Huang, T. 2004. Pooling spaces and non-adaptive pooling designs, *Discrete Mathematics*, 282(1-3), 163–169, URL `http://dx.doi.org/10.1016/j.disc.2003.11.004`.

Hughes-Oliver, Jacqueline. 2006. Pooling experiments for blood screening and drug discovery, in *Screening*, Springer, New York, 48–68, URL `http://dx.doi.org/10.1007/0-387-28014-6_3`.

Hwang, F. K. and Y. C. Liu. 2003. Random pooling designs under various structures, *Journal of Combinatorial Optimization*, 7(4), 339–352, URL `http://dx.doi.org/10.1023/B:JOCO.0000017382.83399.0b`.

Irizarry, Rafael A., Bridget Hobbs, Francois Collin, Yasmin D. Beazer-Barclay, Kristen J. Antonellis, Uwe Scherf, and Terence P. Speed. 2003. Exploration, normalization, and summaries of high density oligonucleotide array probe level data, *Biostat*, 4(2), 249–264, URL `http://dx.doi.org/10.1093/biostatistics/4.2.249`.

Janzen, William P. 2002. *High Throughput Screening: Methods and Protocols (Methods in Molecular Biology, 190)*, Humana Press.

Jin, Fulai, Larisa Avramova, Jing Huang, and Tony Hazbun. 2007. A yeast two-hybrid smart-pool-array system for protein-interaction mapping, *Nat Meth*, 4(5), 405–407, URL `http://dx.doi.org/10.1038/nmeth1042`.

Jin, Fulai, Tony Hazbun, Gregory A. Michaud, Michael Salcius, Paul F. Predki, Stanley Fields, and Jing Huang. 2006. A pooling-deconvolution strategy for biological network elucidation., *Nature methods*, 3(3), 183–189, URL `http://dx.doi.org/10.1038/nmeth859`.

Jones, Matthew C. and Anatoly A. Zhigljavsky. 2001. Comparison of costs for multi-stage group testing methods in the pharmaceutical industry, *Communications in Statistics - Theory and Methods*, 30(10), 2189–2209, URL `http://dx.doi.org/10.1081/STA-100106070`.

Kainkaryam, Raghunandan, Angela Bruex, Anna Gilbert, John Schiefelbein, and Peter Woolf. 2010. poolmc: Smart pooling of mrna samples in microarray experiments, *BMC Bioinformatics*, 11(1), 299, URL `http://www.biomedcentral.com/1471-2105/11/299`.

Kainkaryam, Raghunandan M. and Peter J. Woolf. 2008. poolhits: A shifted transversal design based pooling strategy for high-throughput drug screening, *BMC Bioinformatics*, 9(1), URL `http://dx.doi.org/10.1186/1471-2105-9-256`.

Kainkaryam, Raghunandan M. and Peter J. Woolf. 2009. Pooling in high-throughput drug screening, *Current Opinion in Drug Discovery & Development*, 12(3), 339–350, URL `http://www.biomedcentral.com/content/pdf/cd-1002721.pdf`.

Kautz, W. and R. Singleton. 1964. Nonrandom binary superimposed codes, *IEEE Transactions on Information Theory*, 10(4), 363–377.

Kendziorski, C, R A A. Irizarry, K-S S. Chen, J D D. Haag, and M N N. Gould. 2005. On the utility of pooling biological samples in microarray experiments., *Proc Natl Acad Sci U S A*, URL `http://dx.doi.org/10.1073/pnas.0500607102`.

Kennedy, Shaun. 2008. Epidemiology: Why can't we test our way to absolute food safety?, *Science*, 322(5908), 1641–1643, URL `http://dx.doi.org/10.1126/science.1163867`.

Kim, Hae-Young Y., Michael G G. Hudgens, Jonathan M M. Dreyfuss, Daniel J J. Westreich, and Christopher D D. Pilcher. 2007. Comparison of group testing algorithms for case identification in the presence of test error., *Biometrics*, URL `http://dx.doi.org/10.1111/j.1541-0420.2007.00817.x`.

Knill, E., A. Schliep, and Torney. 1996. Interpretation of pooling experiments using the markov chain monte carlo method, *Journal of Computational Biology*, 3(3), 395–406, URL `http://dx.doi.org/10.1089/cmb.1996.3.395`.

Koehn, Frank E. 2008. High impact technologies for natural products screening, in Frank Petersen and René Amstutz, (Eds.) *Natural Compounds as Drugs Volume I*, Basel: Birkhäuser Basel, vol. 65 of *Progress in Drug Research*, chap. 5, 175–210, URL `http://dx.doi.org/10.1007/978-3-7643-8117-2_5`.

Leifert, Wayne R., Kelly Bailey, Tamara H. Cooper, Amanda L. Aloia, Richard V. Glatz, and Edward J. McMurchie. 2006. Measurement of heterotrimeric g-protein and regulators of g-protein signaling interactions by time-resolved fluorescence resonance energy transfer, *Analytical Biochemistry*, 355(2), 201 – 212, URL `http://www.sciencedirect.com/science/article/B6W9V-4JXRHWX-6/2/d9158b28d96094c027a95fe04937f529`.

Malo, Nathalie, James A. Hanley, Sonia Cerquozzi, Jerry Pelletier, and Robert Nadon. 2006. Statistical practice in high-throughput screening data analysis, *Nature Biotechnology*, 24(2), 167–175, URL `http://dx.doi.org/10.1038/nbt1186`.

Mardis, Elaine R. 2008a. The impact of next-generation sequencing technology on genetics, *Trends in Genetics*, 24(3), 133 – 141, URL http://www.sciencedirect.com/science/article/B6TCY-4RTCPK7-2/2/eeab9ba2236d4783467cd05d61b1fa9d.

Mardis, Elaine R. 2008b. Next-generation dna sequencing methods, *Annual Review of Genomics and Human Genetics*, 9(1), 387–402.

Martin, Scott E., Tamara L. Jones, Cheryl L. Thomas, Philip L. Lorenzi, Dac A. Nguyen, Timothy Runfola, Michele Gunsior, John N. Weinstein, Paul K. Goldsmith, Eric Lader, Konrad Huppi, and Natasha J. Caplen. 2007. Multiplexing sirnas to compress rnai-based screen size in human cells., *Nucleic acids research*, 35(8), URL http://dx.doi.org/10.1093/nar/gkm141.

Mary-Huard, Tristan, Jean-Jacques J. Daudin, Michela Baccini, Annibale Biggeri, and Avner Bar-Hen. 2007. Biases induced by pooling samples in microarray experiments., *Bioinformatics*, 23(13), URL http://dx.doi.org/10.1093/bioinformatics/btm182.

Mercier, Kelly A. and Robert Powers. 2005. Determining the optimal size of small molecule mixtures for high throughput nmr screening, *Journal of Biomolecular NMR*, 31(3), 243–258.

Mishra, K.P., L. Ganju, M. Sairam, P.K. Banerjee, and R.C. Sawhney. 2008. A review of high throughput technology for the screening of natural products, *Biomedicine and Pharmacotherapy*, 62(2), 94 – 98, URL http://www.sciencedirect.com/science/article/B6VKN-4P7FW3P-1/2/a644883ac33b250d7a6832acce4cf90e.

Motlekar, Nuzhat, Scott L. Diamond, and Andrew D. Napper. 2008. Evaluation of an orthogonal pooling strategy for rapid high-throughput screening of proteases, *ASSAY and Drug Development Technologies*, 6(3), 395–405, URL http://dx.doi.org/10.1089/adt.2007.110.

Neubig, R. R. 2002. Regulators of g&nbsp;protein signaling (rgs proteins): Novel central nervous system drug targets, *The Journal of Peptide Research*, 60(6), 312–316, URL http://dx.doi.org/10.1034/j.1399-3011.2002.21064.x.

Patterson, Nick and Stacey Gabriel. 2009. Combinatorics and next-generation sequencing, *Nature Biotechnology*, 27(9), 826–827, URL http://dx.doi.org/10.1038/nbt0909-826.

Peng, Xuejun, Constance Wood, Eric Blalock, Kuey Chen, Philip Landfield, and Arnold Stromberg. 2003. Statistical implications of pooling rna samples for microarray experiments, *BMC Bioinformatics*, 4(1), 26, URL http://www.biomedcentral.com/1471-2105/4/26.

Porat, Ely and Amir Rothschild. 2008. Explicit non-adaptive combinatorial group testing schemes, in *Automata, Languages and Programming*, Springer Berlin / Heidelberg, 748–759, URL http://dx.doi.org/10.1007/978-3-540-70575-8_61.

Prabhu, Snehit and Itsik Pe'er. 2009. Overlapping pools for high-throughput targeted re-sequencing, *Genome Research*, 19(7), 1254–1261, URL `http://dx.doi.org/10.1101/gr.088559.108`.

Remlinger, Katja S., Jacqueline M. Hughes-Oliver, Stanley S. Young, and Raymond L. Lam. 2006. Statistical design of pools using optimal coverage and minimal collision, *Technometrics*, 48(1), 133–143, URL `http://dx.doi.org/10.1198/004017005000000481`.

Schliep, Alexander, David C. Torney, and Sven Rahmann. 2003. Group testing with dna chips: generating designs and decoding experiments., *Proceedings / IEEE Computer Society Bioinformatics Conference. IEEE Computer Society Bioinformatics Conference*, 2, 84–91, URL `http://view.ncbi.nlm.nih.gov/pubmed/16452782`.

Selvin, Paul R. 2000. The renaissance of fluorescence resonance energy transfer, *Nat Struct Mol Biol*, 7(9), 730–734, URL `http://dx.doi.org/10.1038/78948`.

Shih, Joanna H., Aleksandra M. Michalowska, Kevin Dobbin, Yumei Ye, Ting Hu Qiu, and Jeffrey E. Green. 2004. Effects of pooling mrna in microarray class comparisons, *Bioinformatics*, bth391–, URL `http://bioinformatics.oxfordjournals.org/cgi/content/abstract/bth391v1`.

Silva, Jose M., Krista Marran, Joel S. Parker, Javier Silva, Michael Golding, Michael R. Schlabach, Stephen J. Elledge, Gregory J. Hannon, and Kenneth Chang. 2008. Profiling essential genes in human mammary cells by multiplex rnai screening, *Science*, 319(5863), 617–620, URL `http://dx.doi.org/10.1126/science.1149185`.

Snider, Michael. 1998. Screening of compound libraries... consomme or gumbo?, *Journal of Biomolecular Screening*, 3(3), 169–170, URL `http://jbx.sagepub.com`.

Sobel, M. and P. A. Groll. 1959. Group testing to eliminate efficiently all defectives in a binomial sample, *The Bell System Journal*, 38(1179-1252).

Thierry-Mieg, N. 2006a. A new pooling strategy for high-throughput screening: the shifted transversal design., *BMC Bioinformatics*, 7, URL `http://dx.doi.org/10.1186/1471-2105-7-28`.

Thierry-Mieg, Nicolas. 2006b. Pooling in systems biology becomes smart., *Nature methods*, 3(3), 161–162, URL `http://dx.doi.org/10.1038/nmeth0306-161`.

Thierry-Mieg, Nicolas and Gilles Bailly. 2008. Interpool: interpreting smart-pooling results, *Bioinformatics*, 24(5), 696–703, URL `http://dx.doi.org/10.1093/bioinformatics/btn001`.

Tibshirani, Robert. 1997. The lasso method for variable selection in the cox model, *Statistics in Medicine*, 16(4), 385–395, URL `http://dx.doi.org/10.1002/(SICI)1097-0258(19970228)16:4%3C385::AID-SIM380%3E3.0.CO;2-3`.

Traynor, John R. and Richard R. Neubig. 2005. Regulators of g protein signaling & drugs of abuse, *Molecular Interventions*, 5(1), 30–41, URL http://dx.doi.org/10.1124/mi.5.1.7.

Tusher, Virginia G., Robert Tibshirani, and Gilbert Chu. 2001. Significance analysis of microarrays applied to the ionizing radiation response, *Proceedings of the National Academy of Sciences of the United States of America*, 98(9), 5116–5121, URL http://dx.doi.org/10.1073/pnas.091062498.

Uehara, Hiroaki and Masakazu Jimbo. 2009. A positive detecting code and its decoding algorithm for dna library screening, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 99(1), URL http://dx.doi.org/10.1109/TCBB.2007.70266.

Vermeirssen, Vanessa, Bart Deplancke, M. Inmaculada Barrasa, John S. Reece-Hoyes, H. Efsun Arda, Christian A. Grove, Natalia J. Martinez, Reynaldo Sequerra, Lynn Doucette-Stamm, Michael R. Brent, and Albertha J. Walhout. 2007. Matrix and steiner-triple-system smart pooling assays for high-performance transcription regulatory network mapping., *Nature methods*, 4(8), 659–664, URL http://dx.doi.org/10.1038/nmeth1063.

Walters, Patrick and Mark Namchuk. 2003. Designing screens: How to make your hits a hit, *Nat Rev Drug Discov*, 2(4), 259–266, URL http://dx.doi.org/10.1038/nrd1063%0D%0A.

Warrior, Usha, Gopalakrishnan, M. Sujatha, Traphagen, Linda, Freiberg, Gail, Towne, Danli, Humphrey, Patrick, Kofron, James, Burns, and J. David. 2007. Maximizing the identification of leads from compound mixtures, *Letters in Drug Design &#38; Discovery*, 4(3), 215–223, URL http://dx.doi.org/10.2174/157018007780077408.

Westreich, Daniel J., Michael G. Hudgens, Susan A. Fiscus, and Christopher D. Pilcher. 2008a. Optimizing screening for acute human immunodeficiency virus infection with pooled nucleic acid amplification tests, *J. Clin. Microbiol.*, 46(5), 1785–1792, URL http://dx.doi.org/10.1128/JCM.00787-07.

Westreich, Daniel J., Michael G. Hudgens, Susan A. Fiscus, and Christopher D. Pilcher. 2008b. Optimizing screening for acute human immunodeficiency virus infection with pooled nucleic acid amplification tests, *J. Clin. Microbiol.*, 46(5), 1785–1792, URL http://dx.doi.org/10.1128/JCM.00787-07.

Wilson-Lingardo, Laura, Peter Davis, David Ecker, Normand Hebert, Oscar Acevedo, Kelly Sprankle, Thomas Brennan, Leslie Schwarcz, Susan Freier, and Jacqueline Wyatt. 1996. Deconvolution of combinatorial libraries for drug discovery: Experimental comparison of pooling strategies, *Journal of Medicinal Chemistry*, 39(14), 2720–2726, URL http://dx.doi.org/10.1021/jm960169g.

Wu, Shaogui and Bailing Liu. 2005. Application of scintillation proximity assay in drug discovery, *BioDrugs*, 19, 383–392(10).

Xie, Minge, Kay Tatsuoka, Jerome Sacks, and S. Stanley Young. 2001. Group testing with blockers and synergism, *Journal of the American Statistical Association*, 96(453), 92–102, URL `http://www.jstor.org/stable/2670343`.

Xin, Xiaofeng, Jean-François Rual, Tomoko Hirozane-Kishikawa, David E. Hill, Marc Vidal, Charles Boone, and Nicolas Thierry-Mieg. 2009. Shifted transversal design smart-pooling for high coverage interactome mapping, *Genome Research*, 19(7), 1262–1269, URL `http://dx.doi.org/10.1101/gr.090019.108`.

Zhang, Shu-Dong and Timothy W. Gant. 2005. Effect of pooling samples on the efficiency of comparative studies using microarrays, *Bioinformatics*, 21(24), 4378–4383, URL `http://bioinformatics.oxfordjournals.org/cgi/content/abstract/21/24/4378`.

Zhang, Wuyan, Alicia Carriquiry, Dan Nettleton, and Jack C. Dekkers. 2007. Pooling mrna in microarray experiments and its effect on power, *Bioinformatics*, 23(10), 1217–1224, URL `http://dx.doi.org/10.1093/bioinformatics/btm081`.

Zimmermann, G. R., J. Lehár, and C. T. Keith. 2007. Multi-target therapeutics: when the whole is greater than the sum of the parts., *Drug discovery today*, 12(1-2), 34–42, URL `http://dx.doi.org/10.1016/j.drudis.2006.11.008`.