ON SOLVING SELECTED NONLINEAR INTEGER PROGRAMMING PROBLEMS IN DATA MINING, COMPUTATIONAL BIOLOGY, AND SUSTAINABILITY

by

Andrew Christopher Trapp

BS, Rochester Institute of Technology, 2000

MS, Bowling Green State University, 2006

Submitted to the Graduate Faculty of

the Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2011

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Andrew Christopher Trapp

It was defended on

March 3, 2011

and approved by

Oleg A. Prokopyev, Assistant Professor, Department of Industrial Engineering

Andrew J. Schaefer, Associate Professor, Department of Industrial Engineering

Carlos J. Camacho, Associate Professor, Department of Computational Biology

Juan Pablo Vielma, Assistant Professor, Department of Industrial Engineering

Jayant Rajgopal, Associate Professor, Department of Industrial Engineering

Dissertation Director: Oleg A. Prokopyev, Assistant Professor, Department of Industrial Engineering

ON SOLVING SELECTED NONLINEAR INTEGER PROGRAMMING PROBLEMS IN DATA MINING, COMPUTATIONAL BIOLOGY, AND SUSTAINABILITY

Andrew Christopher Trapp, PhD

University of Pittsburgh, 2011

This thesis consists of three essays concerning the use of optimization techniques to solve four problems in the fields of data mining, computational biology, and sustainable energy devices. To the best of our knowledge, the particular problems we discuss have not been previously addressed using optimization, which is a specific contribution of this dissertation. In particular, we analyze each of the problems to capture their underlying essence, subsequently demonstrating that each problem can be modeled as a nonlinear (mixed) integer program. We then discuss the design and implementation of solution techniques to locate optimal solutions to the aforementioned problems. Running throughout this dissertation is the theme of using mixed-integer programming techniques in conjunction with context-dependent algorithms to identify optimal and previously undiscovered underlying structure.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE: SUSTAINABLE ENERGY DEVICES

A	Area (m^2)
c	Speed of sound ($m \cdot s^{-1}$)
C	Capacitance (m^{-1})
c_p	Heat capacity ($J \cdot kg^{-1} \cdot K^{-1}$)
d	Diameter
D	Dimension
f	Frequency (s^{-1})
g	Gravitational acceleration
h	Heat transfer coefficient ($W \cdot m^{-2} \cdot K^{-1}$)
H	Height (Cylindrical Radius) (m)
k_b	Boltzmann constant
k	Thermal conductivity ($W \cdot m^{-1} \cdot K^{-1}$)
l	Plate thickness (m)
L	Inertance ($kg \cdot m^{-4}$), length (m)
p	Pressure ($N \cdot m^{-2}$)
$\tilde{p}$	Constant for quadratic pressure estimate
Q	Heat flow (W)
r	Variable radius height (m) along radial direction
$\tilde{r}$	Constant for viscous resistance formulation
R	Resistance ($kg \cdot m^{-2} \cdot s^{-1}$)
T	Temperature (K, $^\circ C$)
u	Velocity ($m \cdot s^{-1}$)
$\tilde{u}$	Constant for quadratic velocity estimate
w	Objective function component weight
W	Acoustic work (W) per channel
y	Plate spacing (m)
z	Distance along stack; 0 at "hot side" of stack
Z	Stack Placement (along z axis), 0 at closed end

GREEK SYMBOLS: SUSTAINABLE ENERGY DEVICES

α	Thermal diffusion rate ($m^2 \cdot s^{-1}$)
β	Thermal expansion coefficient (taken as $1/T_\infty$)
δ	Penetration depth (m)
ϵ	Plate heat capacity ratio
ε	Surface emissivity
γ	Isentropic coefficient
Γ	Temperature gradient ratio
λ	Wavelength
μ	Dynamic viscosity ($kg \cdot m^{-1} \cdot s^{-1}$)
ν	Viscous diffusion rate ($m^2 \cdot s^{-1}$)
ρ	Density ($kg \cdot m^{-3}$)
ω	Angular frequency (s^{-1})
Π	Perimeter (m)
∇T	Temperature gradient ($K \cdot m^{-1}$)

DIMENSIONLESS GROUPS: SUSTAINABLE ENERGY DEVICES

$\mathcal{A}$	Packing Number ($\approx 1.25 \pm 0.25$)
$\mathcal{F}$	Fixed Upper Bounding Constant (4)
Gr	Grasshoff Number
Nu	Nusselt Number
Pr	Prandtl Number
Ra	Rayleigh Number

SUBSCRIPTS AND SUPERSCRIPTS: SUSTAINABLE ENERGY DEVICES

o	Naught
∞	Ambient, free stream
c	Channel, cold
$char$	Characteristic
$crit$	Critical
$cond$	Conductive
$conv$	Convective
D	Diameter
h	Hot side
κ	Thermal
m	Time averaged
obj	Objective
rad	Radiative
s	Solid, surface, stable
rr,rz,zr,zz	Tensor directions
ν	Viscous
w	Wall

PREFACE

The contents of this dissertation could only come through the sacrificial help and service of so many. First and foremost, I would like to thank Jesus, my constant companion and support throughout this process. Without Your sustaining power I would not be where I am today. To my family, I would like to thank my parents for noticing my God-given talent, encouraging me to pursue mathematics, and believing in me throughout this whole process. To my wife Natasha, your unfailing love and constant companionship through these long years have made this journey so much easier. Thank you. To my son Jonathan, I hope that I can be the father that you need. And to the little one that is on the way, I am looking forward to getting to know you and spending life with you.

Academically, I owe so much to my advisor Dr. Oleg Prokopyev, whose excellent guidance helped me to navigate the sometimes rough and choppy waters of the doctorate. In addition to your incredible talent and insight, I also appreciate your patience with me as I have matured as a researcher. Thank you for the countless hours you invested in me. I also want to acknowledge Dr. Andrew Schaefer, who while not my advisor, is on my dissertation committee, and spent much time giving me helpful advice anyway. I consider you a mentor as well as a friend. I also want to thank my committee members Dr. Jayant Rajgopal, Dr. Juan Pablo Vielma, and Dr. Carlos Camacho for all of your helpful support.

In addition, I am grateful for the fruitful collaboration shared with coauthors Dr. Andrew Schaefer, Dr. Carlos Camacho, Dr. Laura Schaefer, Dr. Florian Zink, Dr. Alpay Temiz, Dr. Sergiy Butenko, and Dr. Stanislav Busygin. Dr. Jeffrey Kharoufeh is another mentor and friend. I appreciate your advice and consideration, especially during the many inherent challenges involved in the job search. Dr. Bopaya Bidanda has handled everything that I've brought to him with grace and clarity – I appreciate your kindness more than words can say.

I also would like to thank Dr. Paul Wilson in the Mathematics Department at the Rochester Institute of Technology, who first encouraged me to pursue graduate studies.

I am sincerely grateful for the gracious assistance from Dr. Eric Beckman, Dr. Laura Schaefer, Gena Kovalcik, Kim Wisniewski and other associates at the Mascaro Center for Sustainable Innovation, who played a pivotal role in my doctoral studies through the GAANN Fellowship (#P200A060149) and related training that supported my studies for multiple years. Speaking of support, the IE staff has always been there for me during my stay at Pitt – thank you Minerva, Richard, Nora, George, Jackie, Jim and Frank!

I am grateful for the friendship of many current and former Pitt students, including Sakine Batun, Zeynep Erkin, Amin Khademei, John Flory, Dave Sanchez, Anahita Khojandi, Guvenc Degirmenci, Gorkem Saka, Gozde Icten, Murat Kurt, Isil Ondes, Behdad Beheshti, Florian Zink, Veronica Miller, Mustafa Baz, Osman Ozaltin, Mehmet Gokhan, Pinar Yildirim, Serdar Karademir, Mehmet Can Demirci, Michelle Najera, Lu Whaley, Burhaneddin Sandikci, and Natalie Scala. Also a shout-out to my friends Todd Fytczyk, Vivek Sandela, Rich Nocon, Erik Bardy, and Dave Voelker. For those I have forgotten to mention by name, know that I appreciate your support as well.

1.0 INTRODUCTION

In this thesis we use optimization and context-relevant algorithmic design to identify optimal and previously undiscovered underlying structure in three distinct application areas. To the best of our knowledge, there does not appear to be any applications of exact optimization techniques in the literature for the particular problems we address. We analyze each problem to understand its structure, and subsequently develop one or more mathematical programming formulations that capture its essence. More precisely, we demonstrate that each of these problems can be modeled using nonlinear integer programming. After their formulation, we then take advantage of the particular intricacies of each problem to develop approaches to obtain an optimal solution using integer programming (IP) techniques. The remainder of this chapter will provide a brief introduction to each problem domain we consider, as well as a survey of related literature involving optimization-based solution approaches.

1.1 DATA MINING

Data mining is the art and science of identifying meaningful information according to specified criteria of interest in typically large records of data using advanced mathematical, statistical and/or algorithmic methods [109]. The technology proliferation of the last twenty to thirty years has led to an explosion of data sets that to a large extent have supported, and continue to promote, the development of data mining techniques. Data mining has found applications in such diverse areas as fraud detection [17, 81], internet traffic studies [73], energy demand analysis [39, 90], evaluations for credit ratings [53, 54], customer preference prediction [25, 91], and biomedical data analysis [6, 19, 69, 94, 101, 102].

One of the challenges to successfully mining data is the intensive computation involved in manipulating large data sets to identify the desired hidden information. It is very common in the literature to address data mining tasks using heuristics and approximation algorithms, which while often being relatively efficient at identifying reasonable solutions, they unfortunately lack optimality guarantees. In contrast, our approach is to use optimization-based techniques to identify exact solutions, a growing practice in the literature [6, 19, 69, 94].

It will be useful to distinguish between *supervised* and *unsupervised* learning. Conducting data mining in the context of *supervised* learning implies that preclassified training data are available to assist in performing the desired analysis. Such knowledge is able to guide the data mining task, though care must be taken to ensure that these training data do not lead to *overfitting*, a phenomenon where predictions tend to be based on random error or noise within the training data, rather than on the data themselves. In the case of *unsupervised* learning, training data are not made available, so alternative techniques must be used that allow the data to self-interpret. Data mining embodies several classes of (semi-)automated procedures that include prediction and clustering, which we next review briefly and also discuss optimization-based approaches to handling such tasks.

For large data sets, processing all of the data simultaneously may be beyond current computational limitations. *Dimensionality reduction* is the process of distilling from a large data set only the influential data to explain a relevant phenomenon. When operating on the set of features, this is called *feature selection*, and can drastically reduce the size of the data set by removing that which is determined to have little meaning. Successful applications of optimization techniques in this context include [13, 15].

Prediction, a type of supervised learning, attempts to make inferences about the future through the use of prior and current data. Several types of data mining can fall under the category of prediction, depending upon the desired task to be accomplished. *Association* is the task of mining data for potential cause and effect relationships. It involves scouring the data in search of predictive rules to identify representative data elements that likely imply one or more different data elements. For example, consumers that purchase wine may also have a greater likelihood of purchasing salmon [23]. Loyalty cards that track purchases are one means by which such association rules can be constructed in the retail context.

Some examples of optimization in association involve *optimal rule discovery* [64, 107]. In general, this approach attempts to find the K best rules that optimize a value measure subject to a set of constraints, where K is a user-defined integer. It extends previous efforts that search for a single optimal rule that satisfies all constraints, though has the drawback of potentially lacking in diversity by drawing rules from only one subsection of the data. Chen [23] considers the use of data envelopment analysis (DEA) to search among a candidate set of possible rules to identify a subset that is efficient with respect to several discriminating criteria.

Other types of prediction include *classification*, where training data are used to form discrete sets to which uncategorized data are assigned according to some measure of proximity. Optimization is a central component in many classification tasks, appearing for example in the contexts of supervised biclustering [19], k-means algorithms [12, 28], and support vector machines [30, 89]. *Regression* is another type of prediction that uses the training data. Rather than a discrete classification, it yields a continuous function. It is a classical statistical technique that attempts to fit data in an optimal way through a function that explains a dependent variable in terms of one or more independent variables [72].

Clustering is a data mining task involving unsupervised learning. It is similar to classification in that it attempts to group data into coherent sets based upon some quality criterion. In traditional clustering these sets are not predefined, but rather are formed dynamically as the data are analyzed. Another challenge in clustering is that the number of sets is typically not known a priori. Applying optimization to the context of clustering dates back to the late 1960's and early 1970's [85, 106]. More recently, Bradley et al. [14] use concave minimization to perform the task of clustering.

In contrast to traditional clustering, which is performed on either the sample sets or the feature sets individually, *biclustering* is jointly applied to the feature and sample sets. Biclustering is the specific data mining task we consider in this thesis, addressing the detection of two important unsupervised biclustering patterns that have significance to the field of biomedicine. For additional details on optimization in the context of data mining, we refer the reader to the excellent surveys by Bradley et al. [16] and Olafsson et al. [78].

1.2 COMPUTATIONAL BIOLOGY

The biological sciences have experienced significant advancements over the past twenty to thirty years with the advent of the personal computer. Together with the rise of DNA technology and the successful sequencing of the human genome, computational biology has become a burgeoning interdisciplinary field with many subdomains. We next explore some of the more prominent areas in which optimization-based techniques have been applied.

Sequence analysis involves comparing and contrasting genomic sequences. It includes *sequence alignment*, which is concerned with mapping elements of one sequence to one (pairwise alignment) or more (multiple alignment) other sequences. A measure of proximity evaluating the quality of the mapping is typically involved. This could be beneficial, for example, when trying to infer attributes of an unknown DNA sequence from those of a known sequence. Another challenge in the area of sequence analysis is *protein structure recognition*, which is necessary to properly understand a protein's function.

Optimization has been widely and successfully applied to sequence analysis problems in the literature. Kececioglu et al. [60] as well as Althaus et al. [3] study the multiple sequence alignment problem from the viewpoint of combinatorial optimization, developing multiple formulations and demonstrating strong valid inequalities. One of the many examples of optimization in protein recognition is Xie and Sahinidis [111], who use contact map overlap (CMO) optimization to align protein sequences so that the number of common residue contacts is maximized. Their approach improved upon the performance of then current technologies, yielding results in strong agreement with standard classifications.

Protein structure prediction is another challenge in computational biology. Accurate prediction can help to understand yet undocumented three-dimensional protein structures at the molecular level. Xu et al. [112] conduct 3D protein structure prediction via threading. They introduce the RAPTOR software package that formulates the protein threading problem as a large scale integer programming problem, which when relaxed to a linear programming problem has the advantageous tendency of yielding integral solutions.

While our interests lie in this last domain, still other areas of computational biology exist to which optimization techniques have been successfully applied, among them genome

rearrangements [22] and haplotyping [47]. Several excellent surveys covering a broader selection of optimization-based applications to the field of computational biology can be found in [4, 45, 62].

1.3 SUSTAINABLE ENERGY DEVICES

Sustainability is loosely defined as meeting today's needs without compromising the needs of tomorrow [76]. The vast field of energy has received a great deal of attention in the literature on sustainability, and rightfully so, given the rising global energy demands combined with the steadily decreasing supply of fossil fuels. We choose to focus on an application in the field of sustainable energy that touches the environmental, societal and economic aspects of the ability to sustain life.

In particular we focus on optimizing the design of the thermoacoustic Stirling heat engine (TAE), a device that takes hot heat input and creates loud acoustic work. It is a main driver for the thermoacoustic refrigerator (TAR), which receives the acoustic work and, through a reversal of the process, provides cooling. These devices offer sustainable advances over standard refrigeration means, in that they contain *no toxic gases* such as common CFC- and HCFC-based refrigerants. Additionally, they contain no moving parts, and so maintenance needs are significantly decreased. While TARs are a reality today, they are not yet competitive with incumbent technology in terms of efficiency. Our goal is the use of optimization techniques to identify an optimal structural design for the TAE with can then improve the overall efficiency of the TAR to better compete with incumbent refrigeration technology.

In comparison with the internal combustion engine, the thermoacoustic engine is not nearly as developed; its relatively poor cycle performance is likely due to the lack of understanding regarding the thermal and acoustic parameter tradeoffs [50]. Such complex interactions can be better understood through mathematical analysis as well as optimization, though this approach appears under-utilized in the thermoacoustic literature.

Some existing efforts include Zink et al. [115], who use an optimization-based approach in conjunction with a finite element solver to identify (locally) optimal solutions to their model

featuring two variables. Another study is Minner et al. [71], who consider the optimization of a thermoacoustic refrigeration system. Through extensive model development they seek to optimize the coefficient of performance, considering geometric parameters and fluid properties of the system and the Nelder-Mead simplex algorithm to search for a (locally) optimal solution. However, in order to account for the thermoacoustic operating conditions, they rely extensively upon DeltaE, a blackbox simulation tool based on linear acoustic theory developed by Swift et al. [93]. DeltaE considers a thermoacoustic device as a combination of individual sections, and analyzes each section with regard to its acoustic properties as well as the velocity, pressure, and temperature behavior.

Both Wetzel [108] and Besnoin [11] discuss thermoacoustic device optimization in their studies. While Wetzel focuses on the optimal performance of a thermoacoustic refrigerator, Besnoin targets heat exchangers. Zoontjens [116] demonstrate the optimization of inertance sections of thermoacoustic devices, using DeltaE to vary individual parameters to determine optimal designs. Ueda [103] evaluates how varying certain engine parameters affects pressure amplitudes. Another work that makes use of DeltaE is Tijani et al. [100], who attempt to optimize the spacing of the stack.

The inherent nonlinear dynamics in such thermoacoustic systems result in serious difficulties for solution approaches. With the exception of the Zink et al. [115] and Minner et al. [71] studies, the previous works vary no more than a single parameter, holding all others constant. Moreover, in every case the solution approaches that are used guarantee only a locally optimal solution, which may potentially be greatly inferior to a globally optimal solution.

1.4 PROBLEM STATEMENTS AND CONTRIBUTIONS

The focus of this dissertation is the mathematical analysis, modeling, design and implementation of solution techniques to locate optimal solutions (with respect to an appropriate objective) to problems in the application areas of data mining, computational biology, and sustainable energy devices. The common thread through these seemingly disparate areas is the discovery of underlying structure using nonlinear integer programming techniques.

In data mining, we seek to identify hidden patterns in real biological data using two proposed biclustering criteria. In so doing, our contributions are twofold: i) the development of novel optimization-based approaches that will detect such patterns, and ii) the identification of patterns of optimal size with respect to predefined criteria. In computational biology, our aim is to use optimization to better understand the underlying interaction code structure; describing these relationships using a minimum set of parameters is an additional contribution. Finally, in the area of sustainable energy devices, we identify the structural variable levels that optimize device design with respect to multiple objectives, with the goal of making such devices competitive with incumbent refrigeration technology. Our contribution is the optimization framework that models the thermoacoustic Stirling heat engine and its subsequent optimization to identify globally optimal design parameters by simultaneously varying five structural parameters, improving on existing studies that varied only a single parameter at a time.

1.5 OVERVIEW OF THE DISSERTATION

The remainder of this dissertation is organized in the following manner. In Chapter 2 we present the application of optimization to data mining in solving two biclustering problems, while an application of optimization to the field of computational biology is discussed in Chapter 3. Chapter 4 addresses the design optimization of a sustainable energy device, and conclusions for these applications are drawn in Chapter 5.

2.0 OPTIMIZATION IN DATA MINING

In this chapter we investigate whether a specific class of data mining problems known as *biclustering* can be efficiently addressed using mathematical programming techniques. There are several advantages of using optimization within the context of data mining. Mathematical programming approaches present an adaptable framework that allows for the modeling of logical implications and other coherent pattern-specific structure. For example, if an additional restriction is necessary it can be implemented by simply adding a corresponding constraint to the optimization model. This flexibility lends itself naturally to the formulation of data mining tasks.

Standard methods for solving different classes of mathematical programs include exact methods (e.g., branch-and-bound, branch-and-cut, branch-and-price, cutting plane [77]), metaheuristic techniques [44, 105], as well as approximation algorithms [105]. If optimal solutions are desired, then mathematical programming approaches have the additional advantage of the powerful and robust solvers such as CPLEX that are already available. For data sets that are not particularly large, the corresponding optimization formulations can typically be solved outright without the need for additional configurations. For larger data sets, obtaining optimal solutions may require the application of advanced optimization-based techniques that exploit specific problem structure to enhance stand-alone solver's capabilities.

While numerous data mining approaches exist in both the literature and practice, often no guarantee is made on the quality of the output; there are many that obtain decent results in reasonable amounts of time. In contrast to such heuristic approaches, we leverage mathematical programming approaches in conjunction with state-of-the-art solver technology for the identification of optimal-sized patterns with respect to specified criteria of interest.

2.1 BICLUSTERING

Input data may be given as a rectangular matrix $A = (a_{ij})_{m \times n}$, where m corresponds to the number of features (i.e., rows of the data matrix), and n corresponds to the number of samples (i.e., columns). Thus entry a_{ij} contains the value of the i^{th} feature in the j^{th} sample. Samples are typically observations sampled from a larger population of data, whereas features are attributes that describe the characteristics of each sample. Figure 2.1 presents an initial data set with six samples and nine features. These data are subsequently grouped into six biclusters, albeit in a rather ideal manner. Consider partitioning the samples of a data set into r sample biclusters:

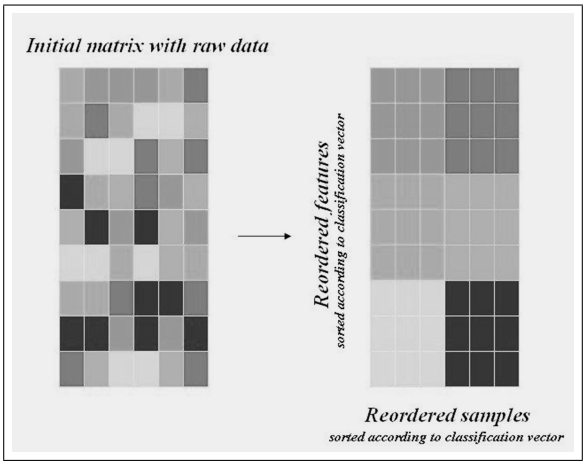


Figure 2.1: Generic scheme of biclustering

$$\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r, \ \mathcal{S}_k \subseteq \{1 \ldots n\}, \ k = 1 \ldots r,$$

$$\mathcal{S}_1 \cup \mathcal{S}_2 \cup \ldots \cup \mathcal{S}_r = \{1 \ldots n\}, \quad \mathcal{S}_k \cap \mathcal{S}_\ell = \emptyset, \ k, \ell = 1 \ldots r, \ k \neq \ell.$$

Here, samples that are grouped together have certain common properties. At the same time, consider partitioning the features into r feature clusters:

$$\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r, \ \mathcal{F}_k \subseteq \{1 \ldots m\}, \ k = 1 \ldots r,$$

$$\mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_r = \{1 \ldots m\}, \quad \mathcal{F}_k \cap \mathcal{F}_\ell = \emptyset, \ k, \ell = 1 \ldots r, \ k \neq \ell.$$

Clustered features also share certain properties in common, with features of cluster k "responsible" for creating sample cluster k, and vice versa. This simultaneous clustering of samples and features according to a specific pattern of interest is known as biclustering, and contrasts with traditional clustering methods that classify only features or samples independently. In biclustering each sample (feature) cluster is induced by a specific subset of samples (features), thereby allowing for the discovery of local patterns of interest that

are relevant to a specific subset of features only for a specific subset of samples (and vice versa). This implies that "clustering derives a global model while biclustering produces a local model" [66]. Biclustering can be formally defined as follows:

Definition 1. *[20] A biclustering of a data set is a collection of pairs of sample and feature subsets* $\mathcal{B} = ((\mathcal{S}_1, \mathcal{F}_1), (\mathcal{S}_2, \mathcal{F}_2), \ldots, (\mathcal{S}_r, \mathcal{F}_r))$ *such that the collection* $(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r)$ *forms a partition of the set of samples, and the collection* $(\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r)$ *forms a partition of the set of features.*

We note that it is not necessary to require an exact partitioning of sample and feature clusters $(\mathcal{S}_k, \mathcal{F}_k)$, so that other pairs $(\mathcal{S}_k, \mathcal{F}_\ell), k \neq \ell$ are also permitted. Overlapping co-clusters is another potential relaxation to the strict partitioning requirement of Definition 1.

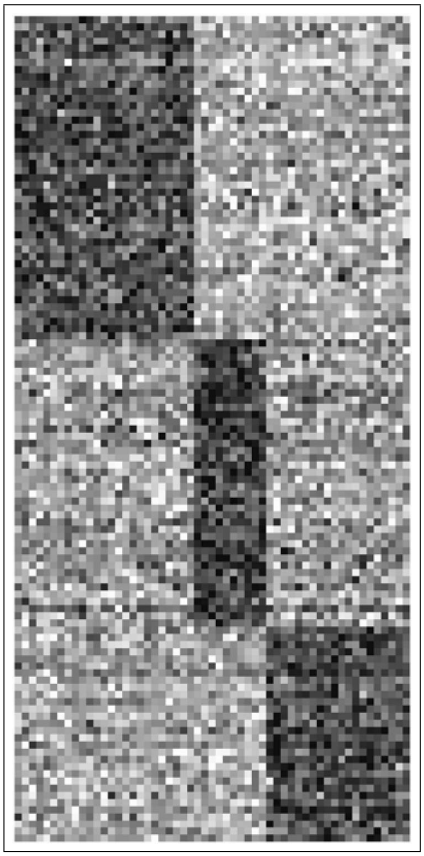


Figure 2.2: $r = 3$ biclusters

Cheng and Church [24] first introduced biclustering in the context of finding co-regulation patterns in gene expression data; they provide a measure to evaluate bicluster quality as well as several efficient algorithms to locate them. Other names for biclustering in the literature include *co-clustering*, *bidimensional clustering*, and *subspace clustering* [20, 66]. Cho et al. [27] introduce two further criteria for evaluating biclusters and propose two efficient heuristic algorithms for their identification. Dhillon [32] presents a novel use of biclustering (as opposed to standard clustering techniques) to analyze common words across documents; a bipartite spectral graph partitioning algorithm is proposed to locate significant textual biclusters. Busygin et al. [18] introduce the double conjugated clustering algorithm that enables any standard clustering algorithm to find biclusters. Kluger et al. [61] establishes the spectral biclustering approach, which uses a condition based upon singular value decomposition to create checkerboard biclusters from genetic microarray data.

The correspondence between clusters of samples and features becomes evident once they have been sorted according to their classifications. This correspondence can be depicted graphically through the use of a *heatmap*, which represents the magnitude of two-dimensional data through the intensity of the corresponding pixel color. Figure 2.2 displays a heatmap of a biclustered data set, revealing three significant biclusters. These biclusters of samples and features can be readily observed from the dark areas containing predominantly dark pixels, which correspond to higher values of the elements a_{ij} in data matrix A.

2.1.1 Applications of Biclustering

Biclustering is of critical importance in biomedical applications, particularly in the analysis of DNA microarray data sets. DNA microarrays measure gene expression levels of thousands of genes simultaneously, allowing researchers to observe their actions across various types of cells. A typical microarray data set includes multiple sample classes representing medical conditions or perhaps certain cell types. When conducted in a reliable manner, biclustering is able to not only diagnose disease conditions represented by sample clusters, but also to identify the genes (features) that serve as their markers [58].

While the primary application domain for biclustering is biomedical data analysis, other application areas include text mining, which is crucial for such techniques as text indexing, web search, text filtering, among others [87]. In marketing contexts, *collaborative filtering* is a biclustering technique that groups together customers having similar preferences on a subset of products. Given data on consumption history, biclustering techniques can then serve to build robust online and in-store recommendation systems for customers.

Additional uses of biclustering include dimensionality reduction of databases via automatic subspace clustering of high dimensional data [1], electoral data analysis to identify groups of constituencies in political districts [48], and analyzing foreign exchange data to locate subsets of currencies whose exchange rates generate related behavior during certain subsets of time periods [63]. Another potential application of biclustering is to detect special structure within the constraint matrices of a given optimization problem, which could

then be used in the implementation of advanced decomposition techniques such as Benders' Partitioning [10], Dantzig-Wolfe Decomposition [31], and others [96].

For more information on a wide variety of biclustering patterns, algorithms and related applications, we refer to the surveys by Busygin et al. [20] and Madeira and Oliveira [66].

2.1.2 Similarity Measures in Biclustering

The quality of a biclustering can be determined by the proximity of samples and features within biclusters as well as their distinctness across other biclusters. There are many ways to establish an acceptable biclustering similarity measure, that is, the process of forming pairs $(\mathcal{S}_k, \mathcal{F}_k)$ of sample and feature subsets. Our key idea is to formulate the patterns of interest as constraints of an optimization problem with an appropriate objective function that measures the desired biclustering properties. To motivate the two formal biclustering pattern definitions we consider, we next discuss some exemplary similarity measures.

One of the most established biclustering metrics is the *mean squared residue score* proposed by Cheng and Church [24]. To formulate this measure, let us introduce:

$$\mu_{ik}^{(r)} = \frac{1}{|\mathcal{S}_k|} \sum_{j \in \mathcal{S}_k} a_{ij} \tag{2.1}$$

as the mean of the i^{th} row in the sample cluster $\mathcal{S}_k$,

$$\mu_{jk}^{(c)} = \frac{1}{|\mathcal{F}_k|} \sum_{i \in \mathcal{F}_k} a_{ij} \tag{2.2}$$

as the mean of the j^{th} column in the feature cluster $\mathcal{F}_k$, and

$$\mu_k = \frac{\sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} a_{ij}}{|\mathcal{F}_k||\mathcal{S}_k|}$$

as the mean value in the bicluster $(\mathcal{S}_k, \mathcal{F}_k)$. The *residue* of element a_{ij} is defined as:

$$r_{ij} = a_{ij} - \mu_{ik}^{(r)} - \mu_{jk}^{(c)} + \mu_k, \ i \in \mathcal{F}_k, \ j \in \mathcal{S}_k. \tag{2.3}$$

Finally, the *mean squared residue* score of the bicluster $(\mathcal{S}_k, \mathcal{F}_k)$ is defined as:

$$H_k = \sum_{i \in \mathcal{F}_k} \sum_{j \in \mathcal{S}_k} (r_{ij})^2 .$$

This value is equal to zero if and only if the entries of all columns (rows) in the bicluster are equal to one another. A bicluster $(\mathcal{S}_k, \mathcal{F}_k)$ is called a δ-bicluster if $H_k \leq \delta$. After proving that finding the largest square δ-bicluster is NP-hard, Cheng and Church used a simple greedy procedure to find biclusters. This approach begins with the entire data matrix and successively removes columns or rows that contribute most to the mean squared residue score [24]. The residue in (2.3) is also utilized by Yang et al. [113] in the FLOC algorithm, as well as by Cho et al. [27] in a manner that simultaneously calculated all biclusters, optimizing the total squared residue as:

$$\min \sum_{k=1}^{K} H_k, \tag{2.4}$$

where input parameter K is the total number of desired biclusters. Cho et al. [27] also introduced the residue measure:

$$r_{ij} = a_{ij} - \mu_k, \tag{2.5}$$

which is the same metric used in "direct clustering" (also known as block clustering) by Hartigan [48]. Cho et al. also demonstrated that their algorithms cause the objective to monotonically decrease expression (2.4) and converge to a local minimum [27]. Another class of optimization-based approaches is based on information theory [34, 35], casting the task of biclustering as an optimization problem that attempts to minimize the resulting loss in mutual information.

We next demonstrate that two distinct biclustering tasks from the literature can be cast as mathematical programs, addressing *unsupervised biclustering under the biclustering consistency conditions* in Section 2.2, and finding *order-preserving submatrices* in Section 2.3.

2.2 UNSUPERVISED BICLUSTERING UNDER THE BICLUSTERING CONSISTENCY CONDITIONS

2.2.1 Acknowledgment

The following content is reproduced with kind permission from Springer Science+Business Media: A. Trapp, O.A. Prokopyev, and S. Busygin, "Finding Checkerboard Patterns via Fractional 0-1 Programming," *Journal of Combinatorial Optimization*, 20 (1), pp. 1-26, 2010.

2.2.2 Introduction

As discussed in Section 2.1.2, biclustering approaches use diverse criteria to associate clusters of samples to clusters of features [20, 66]. The *consistent biclustering* criteria was introduced in [19] to locate checkerboard patterns in data, and an algorithm was presented to handle *supervised* biclustering under the biclustering consistency conditions. The key advantage of the proposed criteria is that in contrast to other biclustering schemes, consistent biclustering is theoretically justified by the conic separation property [19]. We extend this work on consistent biclustering for the case of *unsupervised* learning, developing new optimization-based algorithms for handling the unsupervised biclustering problem under the biclustering consistency conditions. In addition, we also present some new computational complexity results.

We organize the remainder of our discussion in the following manner. In Section 2.2.3 we review the notion of consistent biclustering, while computational complexity issues of the biclustering problem are discussed in Section 2.2.4. We introduce a mathematical programming formulation to handle unsupervised biclustering under biclustering consistency conditions in Section 2.2.5. In Section 2.2.6 two heuristic algorithms for unsupervised biclustering are presented, while in Section 2.2.7 we discuss our computational experiments. Concluding remarks are presented in Section 2.4.1.

2.2.3 Consistent Biclustering

We next introduce the criteria we will use to conduct biclustering in the context of unsupervised learning, namely, *biclustering consistency* [19]. Let each of the n samples be assigned to one of the clusters $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r$. Define a 0–1 matrix $S = (s_{jk})_{n \times r}$ such that $s_{jk} = 1$ if $j \in \mathcal{S}_k$, and $s_{jk} = 0$ otherwise. The sample class *centroids* can be computed as the matrix $C = (c_{ik})_{m \times r}$:

$$C = AS(S^T S)^{-1}, \tag{2.6}$$

where the k^{th} column represents the centroid of the cluster $\mathcal{S}_k$.

Consider row i of the matrix C. Each of the values c_{ij} reveals the average expression of the i^{th} feature for sample cluster k. Assigning the feature to the cluster where it is most expressed generates the checkerboard pattern. So, let us assign the i^{th} feature to the cluster $\hat{k}$ with the largest $c_{i\hat{k}}$ value:

$$i \in \mathcal{F}_{\hat{k}} \;\Rightarrow\; \forall\, k = 1 \ldots r,\; k \neq \hat{k} :\; c_{i\hat{k}} > c_{ik}. \tag{2.7}$$

In the same manner, let us assume that all features have been partitioned into clusters $\mathcal{F}_1,\ \mathcal{F}_2,\ \ldots,\ \mathcal{F}_r$. Then individual samples can be clustered using the same principle of maximal average expression. We want to determine whether the assignment of samples to feature clusters coincides with the prior assignment of features to sample clusters. To do so, construct a 0–1 matrix $F = (f_{ik})_{m \times r}$ such that $f_{ik} = 1$ if $i \in \mathcal{F}_k$ and $f_{ik} = 0$ otherwise. Then, the feature cluster centroids can be computed using matrix $D = (d_{jk})_{n \times r}$:

$$D = A^T F(F^T F)^{-1}, \tag{2.8}$$

whose k^{th} column represents the centroid of the cluster $\mathcal{F}_k$. We want to verify the sample clustering condition:

$$j \in \mathcal{S}_{\hat{k}} \;\Rightarrow\; \forall\, k = 1 \ldots r,\; k \neq \hat{k} :\; d_{j\hat{k}} > d_{jk}. \tag{2.9}$$

Biclustering consistency is defined as:

Definition 2. *[19] A biclustering $\mathcal{B}$ will be called consistent if both relations (2.7) and (2.9) hold, where the matrices C and D are defined as in (2.6) and (2.8).*

Figure 2.2 of Section 2.1 illustrates an exemplary data set with three consistent biclusters of over-expressed values. The average expression of any feature from bicluster $\mathcal{B}_1$ on samples from bicluster $\mathcal{B}_1$ is greater than the average expression of the same feature on samples from biclusters $\mathcal{B}_2$ and $\mathcal{B}_3$. Correspondingly, the average expression of any sample from bicluster $\mathcal{B}_1$ on features from bicluster $\mathcal{B}_1$ is greater than the average expression of the same sample on features from biclusters $\mathcal{B}_2$ and $\mathcal{B}_3$. Similar observations can be made for biclusters $\mathcal{B}_2$ and $\mathcal{B}_3$. In DNA microarray data, this checkerboard pattern may mean strong up-regulation of certain genes under a cancer condition of a particular type (whose samples constitute one class of the data set).

One of the key advantages of consistent biclustering is that it provides a formal setup for the desired separability of clusters. In particular, consistent biclustering implies separability of the clusters by convex cones:

Theorem 1. *[19] If $\mathcal{B}$ is a consistent biclustering, then convex cones $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_r \subseteq \mathbb{R}^m$ exist such that all samples from $\mathcal{S}_k$ belong to the cone $\mathcal{P}_k$ and no other sample belongs to it, $k = 1 \ldots r$. Similarly, there exist convex cones $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_r \subseteq \mathbb{R}^n$ such that all features from $\mathcal{F}_k$ belong to the cone $\mathcal{Q}_k$ and no other feature belongs to it, $k = 1 \ldots r$.*

It also follows from the conic separability that the convex hulls of clusters are separated, i.e, they do not intersect. In general, however, the consistent biclustering criteria is rather strict and data sets may not necessarily contain biclusters satisfying conditions (2.7) and (2.9). Hence, we say that:

Definition 3. *A data set is called biclustering-admitting if there exists some consistent biclustering for it.*

In supervised clustering a training set of samples with known classifications is provided. Having advanced access to this information permits its use in classifying the test set of samples. We also present a related notion:

Definition 4. *The data set is called conditionally biclustering-admitting with respect to a given (partial) classification of some samples and/or features if there exists a consistent biclustering preserving the given (partial) classification.*

2.2.4 Computational Complexity Issues

The computational complexity of consistent biclustering was left as an open question in [19]. We provide some insight by demonstrating the NP-hardness of finding, for a given data matrix, the largest conditionally biclustering-admitting submatrix (CBASM), even for two classes (i.e., $r = 2$). Consider the following decision version of this problem:

Instance: A real-valued data matrix $A = (a_{ij})_{m\times n}$, $\tilde{\mathcal{S}}_1, \tilde{\mathcal{S}}_2 \subseteq \{1, \ldots, n\}$, $\tilde{\mathcal{F}}_1, \tilde{\mathcal{F}}_2 \subseteq \{1, \ldots, m\}$ such that $\tilde{\mathcal{S}}_1 \bigcap \tilde{\mathcal{S}}_2 = \tilde{\mathcal{F}}_1 \bigcap \tilde{\mathcal{F}}_2 = \emptyset$, and six integers k, k_1, k_2, and ℓ, ℓ_1 and ℓ_2 such that $k \leq n$, $\ell \leq m$, $k_1 \leq |\tilde{\mathcal{S}}_1|$, $k_2 \leq |\tilde{\mathcal{S}}_2|$, $\ell_1 \leq |\tilde{\mathcal{F}}_1|$ and $\ell_2 \leq |\tilde{\mathcal{F}}_2|$.
Question: Are there sets $\mathcal{S}_1$, $\mathcal{S}_2 \subseteq \{1, \ldots, n\}$ and $\mathcal{F}_1$, $\mathcal{F}_2 \subseteq \{1, \ldots, m\}$ such that $\mathcal{S}_1 \bigcap \mathcal{S}_2 = \mathcal{F}_1 \bigcap \mathcal{F}_2 = \tilde{\mathcal{S}}_1 \bigcap \mathcal{S}_2 = \mathcal{S}_1 \bigcap \tilde{\mathcal{S}}_2 = \tilde{\mathcal{F}}_1 \bigcap \mathcal{F}_2 = \mathcal{F}_1 \bigcap \tilde{\mathcal{F}}_2 = \emptyset$, and $|S_1| + |S_2| \geq k$, $|F_1| + |F_2| \geq \ell$, $|\tilde{\mathcal{S}}_1 \bigcap \mathcal{S}_1| \geq k_1$, $|\tilde{\mathcal{S}}_2 \bigcap \mathcal{S}_2| \geq k_2$, $|\tilde{\mathcal{F}}_1 \bigcap \mathcal{F}_1| \geq \ell_1$, $|\tilde{\mathcal{F}}_2 \bigcap \mathcal{F}_2| \geq \ell_2$ and biclustering $\mathcal{B} = ((\mathcal{S}_1, \mathcal{F}_1), (\mathcal{S}_2, \mathcal{F}_2))$ is consistent?

The idea behind the above described decision problem is to check if there exists a submatrix of size $\ell \times k$ that is biclustering-admitting and partially preserves a given classification of samples (at least for k_1 and k_2 samples from $\tilde{\mathcal{S}}_1, \tilde{\mathcal{S}}_2$, respectively) and features (at least for ℓ_1 and ℓ_2 features from $\tilde{\mathcal{F}}_1, \tilde{\mathcal{F}}_2$, respectively). All other samples and features from $\tilde{\mathcal{S}}_i$, $\tilde{\mathcal{F}}_i$, respectively ($i = 1, 2$) that do not belong to their respective $\mathcal{S}_i$ and $\mathcal{F}_i$ in the resulting consistent biclustering are considered to be *outliers*.

Theorem 2. *CBASM problem is NP-complete.*

Proof. In the following reduction we use the **Balanced Complete Bipartite Subgraph** problem, which is known to be NP-complete [43]:

Instance: Bipartite graph $G = (V, E)$, positive integer $K \leq |V|$.
Question: Are there two disjoint sets V_1, $V_2 \subseteq V$ such that $|V_1|{=}|V_2|{=}K$ and such that $u \in V_1$, $v \in V_2$ implies that $\{u, v\} \in E$?

Given a bipartite graph $G = (V, U, E)$ define that matrix $A = (a_{ij})_{m\times n}$ as follows. Let $m = |V| + 1$ and $n = |U| + 1$. Define the values of a_{ij} as follows:

- $a_{ij} = 1$ if $i \leq |V|$, $j \leq |U|$ and $(i, j) \in E$,
- $a_{i,(n+1)} = a_{(m+1),j} = 1 - \epsilon$ for $i = 1 \dots m$, $j = 1 \dots n$, where $\epsilon < \min\{\frac{1}{m}, \frac{1}{n}\}$,
- $a_{(m+1),(n+1)} = 1$,
- $a_{ij} = 0$, otherwise.

Let $\tilde{\mathcal{S}}_1 = \{1, 2, \dots, n\}$, $\tilde{\mathcal{F}}_1 = \{1, 2, \dots, m\}$, $\tilde{\mathcal{S}}_2 = \{n + 1\}$, $\tilde{\mathcal{F}}_2 = \{m + 1\}$, $k = K + 1$, $\ell = K + 1$, $k_1 = K$, $k_2 = 1$, $\ell_1 = K$, $\ell_2 = 1$.

Next we show that G contains a balanced complete bipartite graph of size K if and only if the matrix A contains a submatrix of size $(K + 1) \times (K + 1)$ conditionally biclustering-admitting with respect to a given (partial) classification of samples $\tilde{\mathcal{S}}_1$, $\tilde{\mathcal{S}}_2$ and features $\tilde{\mathcal{F}}_1$, $\tilde{\mathcal{F}}_2$.

The first direction can be proven as follows. Suppose G contains a balanced complete bipartite subgraph with nodes V_1 and V_2 of size K. Let $\mathcal{F}_1$ and $\mathcal{S}_1$ consists of all indices that correspond to nodes from V_1 and V_2, respectively, while $\mathcal{S}_2 = \{n + 1\}$ and $\mathcal{F}_2 = \{m + 1\}$ by construction. Then for any $i \in \mathcal{F}_1$ and $j \in \mathcal{S}_1$:

$$c_{i1} = \frac{\sum_{p \in V_2} a_{ip}}{K} = 1 > 1 - \epsilon = a_{i,(n+1)} = c_{i2}, \quad \text{and} \tag{2.10}$$

$$d_{j1} = \frac{\sum_{q \in V_1} a_{qj}}{K} = 1 > 1 - \epsilon = a_{(m+1),j} = d_{j2}. \tag{2.11}$$

For $(\mathcal{S}_2, \mathcal{F}_2)$ we have that:

$$c_{(m+1),2} = a_{(m+1),(n+1)} = 1 > \frac{\sum_{p \in V_2} a_{(m+1),p}}{K} = 1 - \epsilon = c_{(m+1),1}, \quad \text{and} \tag{2.12}$$

$$d_{(n+1),2} = a_{(m+1),(n+1)} = 1 > \frac{\sum_{q \in V_1} a_{q,(n+1)}}{K} = 1 - \epsilon = d_{(n+1),1}. \tag{2.13}$$

Inequalities (2.10) – (2.13) imply that (2.7) and (2.9) are satisfied and the constructed submatrix is biclustering-admitting.

In order to show the other direction, assume that we have a submatrix of size $(K + 1) \times (K + 1)$ that is conditionally biclustering-admitting with respect to a given (partial) classification of samples and features described above. Let V_1 and V_2 correspond to indices from $\mathcal{F}_1$ and $\mathcal{S}_1$, respectively. Next we show that the subgraph induced by V_1 and V_2 is

complete, i.e., if $i \in \mathcal{F}_1$ and $j \in \mathcal{S}_1$ then $a_{ij} = 1$. Suppose this is not true and there exist $i \in V_1$ and $j \in V_2$ such that $a_{ij} = 0$. Then:

$$c_{i1} = \frac{\sum_{p \in V_2} a_{ip}}{K} \leq \frac{K-1}{K} = 1 - \frac{1}{K} \leq 1 - \epsilon = a_{i,(n+1)} = c_{i2}, \tag{2.14}$$

and (2.7) is not satisfied. This contradicts the initial assumption that the biclustering is consistent. Therefore, $a_{ij} = 1$. ∎

2.2.5 Mathematical Modeling of Consistent Biclustering

We next review in Section 2.2.5.1 a fractional 0–1 program from [19] that can handle the supervised biclustering problem. Afterwards, we discuss how to extend such an approach to model the case of unsupervised biclustering, which does not use training data to provide an initial sample classification.

2.2.5.1 Supervised Consistent Biclustering Supervised consistent biclustering consists of two routines. After evaluating training samples to derive classification criteria, these criteria are subsequently applied to the test samples. When only a small subset of features is expected to be relevant, such as in biological data analysis, the classification criteria should involve dimensionality reduction and feature selection. Consistent biclustering can address such concerns, as it selects a subset of features of the original data set in such a way that the obtained subset of data becomes conditionally biclustering-admitting with respect to the given classification of training samples.

We introduce the vector of 0–1 variables $x = (x_i)_{i=1\ldots m}$, and consider the i^{th} feature selected if $x_i = 1$, and $x_i = 0$, otherwise. The condition of biclustering consistency when considering only the selected features becomes:

$$\frac{\sum_{i=1}^{m} a_{ij} f_{i\hat{k}} x_i}{\sum_{i=1}^{m} f_{i\hat{k}} x_i} > \frac{\sum_{i=1}^{m} a_{ij} f_{ik} x_i}{\sum_{i=1}^{m} f_{ik} x_i}, \ \forall\ j \in \mathcal{S}_{\hat{k}},\ \ \hat{k}, k = 1 \ldots r,\ \ \hat{k} \neq k. \tag{2.15}$$

The fractional relations (2.15) can then be used as constraints in a mathematical program. A potential objective function may involve selecting the largest number of features so as to

lose the least amount of information provided by the training set. Thus one possible fractional 0-1 programming formulation that enforces the biclustering criterion is:

$$\max_{x \in \mathbb{B}^n} \sum_{i=1}^{m} x_i \tag{2.16}$$

subject to

$$\frac{\sum_{i=1}^{m} a_{ij} f_{i\hat{k}} x_i}{\sum_{i=1}^{m} f_{i\hat{k}} x_i} \geq (1+t) \frac{\sum_{i=1}^{m} a_{ij} f_{ik} x_i}{\sum_{i=1}^{m} f_{ik} x_i}, \ \forall \ j \in \mathcal{S}_{\hat{k}}, \ \hat{k}, k = 1 \ldots r, \ \hat{k} \neq k, \tag{2.17}$$

where (2.17) is a modified version of (2.15) involving a positive constant t called the *parameter of separation*. Larger values of t may strengthen the class separation, thereby providing a reduced set selected features with higher quality. This also closes the feasible domain by eliminating the strict inequalities of (2.15).

Test samples are subsequently classified according to (2.9). That is, if $b = (b_i)_{i=1\ldots m}$ is a test sample, then we assign it to the class $\mathcal{F}_{\hat{k}}$ satisfying:

$$\frac{\sum_{i=1}^{m} b_i f_{i\hat{k}} x_i^*}{\sum_{i=1}^{m} f_{i\hat{k}} x_i^*} > \frac{\sum_{i=1}^{m} b_i f_{ik} x_i^*}{\sum_{i=1}^{m} f_{ik} x_i^*}, \ k = 1 \ldots r, \ \hat{k} \neq k, \tag{2.18}$$

where x^* is the output of (2.16) – (2.17) indicating the optimal feature selection.

Optimization problem (2.16) – (2.17) is a specific type of *fractional 0–1 programming* problem [83, 84, 95, 110]. By applying an approach to linearize problems with fractional 0-1 objective functions [110], optimization problem (2.16) – (2.17) can be reformulated as a linear mixed 0-1 programming problem. This technique as well as a simple heuristic algorithm for solving (2.16) – (2.17) were discussed in [19]. The obtained features were used for classification of test data according to (2.18), providing excellent results for two biomedical data sets (HuGE and ALL vs. AML).

2.2.5.2 Unsupervised Consistent Biclustering In contrast to supervised biclustering, where preclassified training data are available, unsupervised biclustering does not use training data to develop appropriate classification criteria, instead allowing the data to self-interpret. Consider assigning each sample to one of the clusters:

$$\mathcal{S}_1,\ \mathcal{S}_2,\ \ldots,\ \mathcal{S}_r.$$

To facilitate this, let $S = (s_{jk})_{n\times r}$ now represent a set of 0–1 variables such that $s_{jk} = 1$ if $j \in \mathcal{S}_k$, and $s_{jk} = 0$, otherwise. Similarly, consider clustering all features into clusters:

$$\mathcal{F}_1,\ \mathcal{F}_2,\ \ldots,\ \mathcal{F}_r.$$

Also, we now let $F = (f_{ik})_{m\times r}$ represent a set of 0–1 variables such that $f_{ik} = 1$ if $i \in \mathcal{F}_k$ and $f_{ik} = 0$, otherwise.

The following sets of constraints:

$$s_{j\hat{k}}\left(\frac{\sum_{i=1}^m a_{ij}f_{i\hat{k}}}{\sum_{i=1}^m f_{i\hat{k}}} - (1+t)\frac{\sum_{i=1}^m a_{ij}f_{ik}}{\sum_{i=1}^m f_{ik}}\right) \geq 0\ \forall\ j,\ \hat{k},\ k = 1\ldots r,\ \hat{k}\neq k, \tag{2.19}$$

$$f_{i\hat{k}}\left(\frac{\sum_{j=1}^n a_{ij}s_{j\hat{k}}}{\sum_{j=1}^n s_{j\hat{k}}} - (1+t)\frac{\sum_{j=1}^n a_{ij}s_{jk}}{\sum_{j=1}^n s_{jk}}\right) \geq 0\ \forall\ i,\ \hat{k},\ k = 1\ldots r,\ \hat{k}\neq k, \tag{2.20}$$

enforce the biclustering consistency conditions of (2.7) and (2.9). Furthermore, the unsupervised biclustering formulation requires additional constraints related to the clustering of samples and features, so that each feature and sample can be assigned to at most one cluster:

$$\sum_{k=1}^r f_{ik} \leq 1\ \forall\ i\ \text{ and }\ \sum_{k=1}^r s_{jk} \leq 1\ \forall\ j, \tag{2.21}$$

and each cluster must contain at least one feature and sample:

$$\sum_{i=1}^m f_{ik} \geq 1\ \forall\ k\ \text{ and }\ \sum_{j=1}^n s_{jk} \geq 1\ \forall\ k. \tag{2.22}$$

Constraints (2.21) – (2.22) could also be modified in various ways. For example, we may allow for certain features to belong to several biclusters, or require each sample to be clustered so that there are no outliers.

Combining constraints (2.19) – (2.20) and (2.21) – (2.22) with a suitable objective function such as:

$$n \cdot \sum_{i=1}^{m} \sum_{k=1}^{r} f_{ik} + m \cdot \sum_{j=1}^{n}, \sum_{k=1}^{r} s_{jk} \tag{2.23}$$

yields a fractional 0–1 programming problem. With objective (2.23), this formulation attempts to select as many features and samples as possible while simultaneously satisfying biclustering consistency conditions. Samples and features that do not appear in any clusters in the solution of this mathematical program are simply taken to be outliers.

2.2.5.3 Linear Mixed 0–1 Reformulation We next demonstrate a linearization that transforms formulation (2.23), (2.19) – (2.20) and (2.21) – (2.22), a type of fractional 0–1 programming problem, into a mixed integer program (MIP) that can then be solved using standard linear mixed integer programming solvers such as CPLEX.

First, observe that conditions (2.19) and (2.20) are equivalent to:

$$\frac{\sum_{i=1}^{m} a_{ij} f_{i\hat{k}}}{\sum_{i=1}^{m} f_{i\hat{k}}} - (1+t)\frac{\sum_{i=1}^{m} a_{ij} f_{ik}}{\sum_{i=1}^{m} f_{ik}} \geq -L_j^s(1 - s_{j\hat{k}}) \quad \forall\ j,\ \hat{k},\ k,\ \hat{k} \neq k, \tag{2.24}$$

$$\frac{\sum_{j=1}^{n} a_{ij} s_{j\hat{k}}}{\sum_{j=1}^{n} s_{j\hat{k}}} - (1+t)\frac{\sum_{j=1}^{n} a_{ij} s_{jk}}{\sum_{j=1}^{n} s_{jk}} \geq -L_i^f(1 - f_{i\hat{k}}) \quad \forall\ i,\ \hat{k},\ k,\ \hat{k} \neq k, \tag{2.25}$$

for large enough constants L_i^f and L_j^s. For instance, these can be chosen as:

$$L_j^s = \max_i a_{ij} - (1+t)\min_i a_{ij}, \quad \text{and} \quad L_i^f = \max_j a_{ij} - (1+t)\min_j a_{ij}. \tag{2.26}$$

The following proposition can then be utilized to linearize our formulation:

Proposition 1. *[110] A polynomial mixed 0–1 term $z = xy$, where x is a 0–1 variable, and y is a nonnegative variable with upper bound M, can be represented by the following linear inequalities: (1) $y - z \leq M - Mx$, (2) $z \leq y$, (3) $z \leq Mx$, and (4) $z \geq 0$.*

Proposition 1 states that a nonlinear variable product containing a binary variable and another nonnegative variable with upper bound M can be implicitly represented by the introduction of a new variable together with four auxiliary linear constraint sets. Making use of Proposition 1, let us introduce new variables:

$$u_k = \frac{1}{\sum_{i=1}^{m} f_{ik}},\ k = 1 \dots r; \quad v_k = \frac{1}{\sum_{j=1}^{n} s_{jk}},\ k = 1 \dots r, \tag{2.27}$$

$$z_{ik} = \frac{f_{ik}}{\sum_{\ell=1}^{m} f_{\ell k}},\ i = 1 \dots m,\ k = 1 \dots r; \quad y_{jk} = \frac{s_{jk}}{\sum_{\ell=1}^{n} s_{\ell k}},\ j = 1 \dots n,\ k = 1 \dots r. \tag{2.28}$$

Then by substituting variables, we can replace nonlinear 0–1 inequalities (2.24) and (2.25) with the following linear-mixed 0–1 constraint sets:

$$\sum_{i=1}^{m} a_{ij} z_{i\hat{k}} - (1+t) \sum_{i=1}^{m} a_{ij} z_{ik} \geq -L_j^s (1 - s_{j\hat{k}}) \ \ \forall\ j,\ \hat{k},\ k,\ \hat{k} \neq k, \tag{2.29}$$

$$\sum_{j=1}^{n} a_{ij} y_{j\hat{k}} - (1+t) \sum_{j=1}^{n} a_{ij} y_{jk} \geq -L_i^f (1 - f_{i\hat{k}}) \ \ \forall\ i,\ \hat{k},\ k,\ \hat{k} \neq k, \tag{2.30}$$

$$\sum_{i=1}^{m} z_{ik} = 1,\ k = 1 \dots r, \quad \sum_{j=1}^{n} y_{jk} = 1,\ k = 1 \dots r, \tag{2.31}$$

$$u_k - z_{ik} \leq 1 - f_{ik},\ v_k - y_{jk} \leq 1 - s_{jk},\ z_{ik} \leq u_k,\ y_{jk} \leq v_k,\ \forall\ i, j, k, \tag{2.32}$$

$$z_{ik} \leq f_{ik},\ y_{jk} \leq s_{jk},\ z_{ik} \geq 0,\ y_{jk} \geq 0,\ \forall\ i, j, k. \tag{2.33}$$

Objective function (2.23) along with conditions (2.21) – (2.22) and (2.29) – (2.33), which we subsequently refer to as the main MIP, represents the original nonlinear 0–1 programming problem with a linear mixed 0–1 program having $2r(m+n+1)$ variables.

2.2.6 Heuristic Approaches

Our linear mixed 0–1 reformulation is not suitable for solving large-scale instances of the biclustering problem even when coupled with the best techniques implemented in modern integer programming solvers. As a consequence, we next present two alternative heuristic approaches for solving this problem. The first algorithm iteratively solves a relaxation of the main MIP, while the second is a heuristic based upon local search that maintains biclustering consistency conditions.

2.2.6.1 Heuristic 1 (H1): MIP-based Heuristic The first heuristic is an extension of the algorithm described in [19]. It solves relaxations of the main MIP in an iterative manner until specified criteria are met.

To shed light on our motivations for this heuristic, consider the meaning of variables z_{ik} and y_{jk}. We have introduced them so that:

$$z_{ik} = \frac{f_{ik}}{\sum_{\ell=1}^{m} f_{\ell k}}, \ i \in \mathcal{F}_k, \quad \text{and} \tag{2.34}$$

$$y_{jk} = \frac{s_{jk}}{\sum_{\ell=1}^{n} s_{\ell k}}, \ j \in \mathcal{S}_k. \tag{2.35}$$

Thus, for $i \in \mathcal{F}_k$, z_{ik} is the reciprocal of the cardinality of cluster $\mathcal{F}_k$ after feature selection, if the i^{th} feature is selected, and 0 otherwise; likewise, for $j \in \mathcal{S}_k$, y_{jk} is the reciprocal of the cardinality of cluster $\mathcal{S}_k$ if the j^{th} sample is selected, and 0 otherwise. This reveals that z_{ik} and y_{jk} are also binary variables just as f_{ik} and s_{jk} are, however, their nonzero values are simply not set to 1. Though these nonzero values are not known until the optimal sizes of feature and sample clusters are obtained, knowing the values of z_{ik} and y_{jk} suffices to define the values of f_{ik} and s_{jk}, and the system of constraints with respect only to the continuous variables $0 \leq z_{ik} \leq 1$ and $0 \leq y_{jk} \leq 1$ (that is, dropping constraints (2.32) involving u_k and v_k) constitutes a linear relaxation of the main MIP. Furthermore, it can be strengthened by the system of inequalities connecting z_{ik} to f_{ik} and y_{jk} to s_{jk}. Indeed, knowing that no more

than m_k features can be selected for cluster $\mathcal{F}_k$ and that no more than n_k samples can be selected for cluster $\mathcal{S}_k$, it is valid to impose the following inequalities:

$$m_k z_{ik} \geq f_{ik} \; \forall \; i,k, \quad \text{and} \quad n_k y_{jk} \geq s_{jk} \; \forall \; j,k. \tag{2.36}$$

Inequalities (2.36) strengthen the relaxation of the main MIP. Furthermore, we can show:

Theorem 3. *If f^*, s^* is an optimal solution to (2.23), (2.21) – (2.22) and (2.29) – (2.33), and if $\forall\; k\; m_k = \sum_{i=1}^{m} f_{ik}^*$, and if $\forall\; k\; n_k = \sum_{j=1}^{n} s_{jk}^*$, then f^*, s^* is also an optimal solution to (2.23), (2.21) – (2.22), (2.29) – (2.31), (2.33), and (2.36).*

Proof. Because the new program is a relaxation of the original problem (noting that the two additional constraint sets are satisfied by any feasible solution to the original program), f^* and s^* must be feasible for the new program as well. Thus, we need only demonstrate that formulation (2.23), (2.21) – (2.22), (2.29) – (2.31), (2.33), and (2.36) does not have a better solution in order to prove the result.

Suppose there exists some f^{**}, s^{**} such that:

$$m\sum_{j=1}^{n}\sum_{k=1}^{r} s_{jk}^{**} + n\sum_{i=1}^{m}\sum_{k=1}^{r} f_{ik}^{**} > m\sum_{j=1}^{n}\sum_{k=1}^{r} s_{jk}^{*} + n\sum_{i=1}^{m}\sum_{k=1}^{r} f_{ik}^{*}.$$

Because constraint sets (2.36) must hold, then summing over i and k gives:

$$n\sum_{i=1}^{m}\sum_{k=1}^{r} f_{ik}^{**} \leq n\sum_{i=1}^{m}\sum_{k=1}^{r} m_k z_{ik},$$

and

$$m\sum_{j=1}^{n}\sum_{k=1}^{r} s_{jk}^{**} \leq m\sum_{j=1}^{n}\sum_{k=1}^{r} n_k y_{jk}.$$

Now summing these equations, and in conjunction with (2.31) along with our assumptions about m_k and n_k, we have that:

$$m\sum_{j=1}^{n}\sum_{k=1}^{r} s_{jk}^{**} + n\sum_{i=1}^{m}\sum_{k=1}^{r} f_{ik}^{**} \leq m\sum_{j=1}^{n}\sum_{k=1}^{r} n_k y_{jk} + n\sum_{i=1}^{m}\sum_{k=1}^{r} m_k z_{ik} =$$

$$m\sum_{k=1}^{r}\sum_{j=1}^{n} n_k y_{jk} + n\sum_{k=1}^{r}\sum_{i=1}^{m} m_k z_{ik} = m\sum_{k=1}^{r} n_k \sum_{j=1}^{n} y_{jk} + n\sum_{k=1}^{r} m_k \sum_{i=1}^{m} z_{ik} =$$

$$= m \sum_{k=1}^{r} n_k + n \sum_{k=1}^{r} m_k = m \sum_{j=1}^{n} \sum_{k=1}^{r} s^*_{jk} + n \sum_{i=1}^{m} \sum_{k=1}^{r} f^*_{ik},$$

which leads us to a contradiction. ∎

Algorithm 1. *[Heuristic 1 (H1): MIP-based Heuristic]*

1. *Assign* $m_k := m$ *and* $n_k := n$, $k = 1 \dots r$.

2. *Solve the mixed 0–1 programming formulation consisting of* (2.23) *together with constraints* (2.21) – (2.22), (2.29) – (2.31), (2.33), *and* (2.36).

3. ***If*** $m_k = \sum_{i=1}^{m} f_{ik}$ $\forall$ k ***And*** $n_k = \sum_{j=1}^{m} s_{jk}$ $\forall$ k, ***Go To*** *6.*

4. *Assign* $m_k := \sum_{i=1}^{m} f_{ik}$ *and* $n_k := \sum_{j=1}^{n} s_{jk}$ $\forall$ k.

5. ***Go To*** *2.*

6. ***Stop***.

Our computational experiments indicate that, though this heuristic performs much faster than solving the main MIP directly, it is still not efficient enough for tackling reasonably large real-life microarray data sets.

2.2.6.2 Heuristic 2 (H2): Multi-start Iterative Heuristic As an alternative to Algorithm 1, we also present another iterative-based heuristic that maintains the biclustering consistency conditions. The heuristic is first primed by generating a random assignment of the n samples to the r clusters. Using this initial clustering of samples, the m features are then clustered according to (2.7). Given the obtained partitioning of features, we update the sample clustering according to (2.9). The process continues in this manner, iteratively refining both row and column clusters, until two stopping conditions have been met:

1) r clusters have been generated, and
2) no samples or features have switched clusters for one full iteration.

It is possible that we may reach an iteration where one or more clusters may become empty. If fewer than r clusters have been formed, the algorithm resets by generating another

random assignment of samples to clusters and proceeds as before. The pseudocode of the above described routine (together with some additional enhancements that are subsequently discussed) is outlined in Procedure 3.

The final result of Procedure 3 can substantially depend on the initial random assignment of samples to clusters. Therefore, we choose to run multiple trials of the procedure (hence the term *multi-start*), in order to determine to which cluster each sample predominantly belonged based on a simple majority vote, with ties broken arbitrarily. Let us formalize our terminology by defining a *solution* as consisting of a particular assignment of samples to clusters, and let us define a *trial* as consisting of one pass of Procedure 3, typically consisting of multiple iterations, until stopping conditions 1) and 2) above have been met.

An initial sample clustering $(\mathcal{S}_1^0, \mathcal{S}_2^0, \ldots, \mathcal{S}_r^0)$ is constructed in Step **1.** of Procedure 1. We obtain our initial solution using multiple trials of Procedure 3. Among generated sample clusterings, we choose the one with minimum value of a specific metric (see Step **3.** of Procedure 1), so as to ensure that the starting solution is in some sense more diverse than the others. After generating the initial sample clustering $(\mathcal{S}_1^0, \ldots, \mathcal{S}_r^0)$, the multi-start Procedure 2 begins. This procedure applies *MSLim* trials of Procedure 3 to generate a starting solution for use in the final biclustering of samples and features (Step **3.** of Algorithm 2). For each trial the corresponding sample clusterings are recorded, and upon completion of the multi-start procedure, there will be a cluster to which each sample was assigned a majority of times (ties broken arbitrarily). We assign each sample to its "majority" cluster and use this final sample clustering in the final trial of the algorithm.

A potential issue with this multi-start Procedure 2 is that the actual clusters $k = 1 \ldots r$ are not necessarily unique across trials, that is, it's possible to have identical sample clusterings with differing cluster numbers. For example, in a given trial samples 1 and 2 could be assigned to cluster 1, while samples 3 and 4 could be assigned to cluster 2. This clustering should be considered identical to the clustering where samples 1 and 2 are assigned to cluster 2 while samples 3 and 4 are assigned to cluster 1. In order to eliminate such undesirable symmetry we make use of the initial sample clustering $(\mathcal{S}_1^0, \mathcal{S}_2^0, \ldots, \mathcal{S}_r^0)$ (see Step **3.** of Procedure 2 for more information).

Algorithm 2. *[Heuristic 2 (H2): Multi-start Iterative Heuristic]*

1. *Call Procedure 1.*
2. *Call Procedure 2.*
3. *Using results from Procedure 2 as initial sample clustering, call Procedure 3 to obtain final biclustering.*

Procedure 1. */* Construction of initial sample clustering */*

1. ***Do*** *generate random assignment of samples to clusters.*
2. *Call Procedure 3.*
3. *Evaluate "diversity" of the resulting solution by computing*

$$d = \sum_{k=1}^{r} \left(\frac{n_k}{n} - \frac{1}{r} \right)^2,$$

where n_k is the number of samples in cluster k.

4. ***If*** *the obtained solution has a lower value of d, store it as the current initial sample clustering $(\mathcal{S}_1^0, \mathcal{S}_2^0, \ldots, \mathcal{S}_r^0)$.*
5. ***While*** *initTrialCount $<$ initLim;*
6. ***Stop***, *and* ***Return*** *the obtained solution.*

Procedure 2. */* Improve initial sample clustering via voting */*

1. ***Do*** *generate random assignment of samples to clusters.*
2. *Call Procedure 3 to obtain sample clustering $(\mathcal{S}_{1'}, \mathcal{S}_{2'}, \ldots, \mathcal{S}_{r'})$.*
3. *Permute $i' \rightarrow j$ minimizing Euclidean distance of $(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_r)$ from $(\mathcal{S}_1^0, \mathcal{S}_2^0, \ldots, \mathcal{S}_r^0)$.*
4. *Record assignment of samples to clusters for current trial.*
5. ***While*** *MSTrialCount $<$ MSLim;*
6. *Categorize each sample according to the cluster to which it was assigned a majority of times; break ties arbitrarily.*

Procedure 3. */* Iteration-based biclustering */*

1. *Assign features to clusters according to sample clustering. For a given feature, this is accomplished as follows:*

2. ***For*** $k = 1 \dots r$*, determine the number of entries* n_k *and the sum of entries* s_k *in this feature.*

3. *Across all* $k : n_k \geq 1$*, calculate the average expression* $\frac{s_k}{n_k}$*. Let cluster* k^* *be the cluster with the largest average expression.*

4. ***For all*** $\hat{k} : \hat{k} \neq k^*, n_{\hat{k}} \geq 1$ *in this feature, verify that* $\frac{s_{k^*}^*}{n_{k^*}^*} > (1+t)\frac{s_{\hat{k}}}{n_{\hat{k}}}$*, where* t *is the parameter of separation.*

5. ***If*** *the inequality in Step* ***4.*** *does not hold for at least one* $\hat{k}$*, this feature is labeled as unclustered; otherwise, the feature is assigned to cluster* k^**.*

6. *Once all features have been assigned to a particular cluster according to sample clustering, cluster samples according to feature clustering by repeating similar steps.*

7. ***If*** *an entire pass occurs with no feature nor sample clustering changes, find the number of non-empty clusters* $\hat{r}$*.*

8. ***If*** $\hat{r} = r$*,* ***Stop****. Otherwise,* ***Go To*** *Step* ***1.*** *of the calling procedure and restart the current trial.*

Note that in Step **5.** of Procedure 3 features (and samples) can be labeled as *unclustered*, so that in Steps **1.** through **3.**, only *clustered* samples are considered in feature clustering, and likewise in sample clustering.

Algorithm 2 may continue indefinitely due to some features consistently alternating between a small number of clusters; indeed, in computational testing there was a tendency for some features to make alternating moves each iteration. Two parameters are introduced to eliminate this possibility: *PercentCutOff* and *BeginFlopCheck*. For each feature, the actual ratio of the number of alternating moves per total number of iterations is calculated for a given trial. *PercentCutOff* specifies the permissible ratio of alternating feature moves per total number of iterations for a given trial. The *BeginFlopCheck* parameter works in conjunction with the *PercentCutOff* parameter by specifying the number of iterations after

which we begin checking for the *PercentCutOff* ratio; this allows for some early shuffling of features between clusters. After reaching *BeginFlopCheck* number of iterations, if the actual ratio is greater than *PercentCutOff*, the feature is discarded from future consideration for the remainder of this trial. These modifications cause rapid convergence for a given trial.

2.2.7 Computational Experiments and Results

2.2.7.1 Test Data We use both *synthetic* and *real biological* data to test our algorithms. We next detail the types of data used to conduct our computational experiments.

Synthetic Data Sets. A set of synthetic test instances was generated in order to test the accuracy of the main MIP formulation. To create these instances, we varied the values of m and n, using $r = 2, 3, 4$. The test instances were constructed with m rows and n columns, with entries formed by repeating the values $\{0, 1, \ldots, r\}$. An example with $m = 6$, $n = 6$, and $r = 3$ is displayed in Table 2.1. Table 2.1a presents unclassified data, while Table 2.1b presents the data after classification. Three biclusters appear in Table 2.1b, each consisting of two samples and two features.

Table 2.1: Synthetic test data: $m = 6$, $n = 6$, $r = 3$

(a) Unclassified data

0	1	2	0	1	2
2	0	1	2	0	1
1	2	0	1	2	0
0	1	2	0	1	2
2	0	1	2	0	1
1	2	0	1	2	0

(b) Classified data

2	**2**	0	0	1	1
2	**2**	0	0	1	1
1	1	**2**	**2**	0	0
1	1	**2**	**2**	0	0
0	0	1	1	**2**	**2**
0	0	1	1	**2**	**2**

HuGE Data Set. Our biclustering algorithms were tested on the Human Gene Expression (HuGE) Index data set [55]. The purpose of the HuGE project is to provide a comprehensive database of gene expressions in normal tissues of different parts of the human body as well as to highlight similarities and differences among the organ systems. We refer the reader to [52] for the detailed description of these studies. The HuGE data set consists of 59 samples from 19 distinct tissue types. It was obtained using oligonucleotide microarrays capturing 7,070 genes (features). The samples were obtained from 49 human individuals: 24 males with median age of 63 and 25 females with median age of 50. Each sample came from a different

Table 2.2: Summary of tissues contained in HuGE data set

Tissue type	**Abbreviation**	**# Samples**
blood	BD	1
brain	BRA	11
breast	BRE	2
colon	CO	1
cervix	CX	1
endometrium	END	2
esophagus	ES	1
kidney	KI	6
liver	LI	6
lung	LU	6
muscle	MU	6
myometrium	MYO	2
ovary	OV	2
placenta	PL	2
prostate	PR	4
spleen	SP	1
stomach	ST	1
testes	TE	1
vulva	VU	3

individual except for the first seven BRA samples that were from different brain regions of the same individual and the 5^{th} LI sample, which came from that individual as well. The HuGE data set is summarized in Table 2.2.

We made use of the HuGE data set in testing both Algorithms 1 and 2. We generated test data for our first algorithm by creating subsets of two of the 12 total tissue types having

multiple samples. For these tissue types, all samples were included in the data set, and all genes were included so long as they contained no negative expressions. Subsets of the HuGE data set were also used to test the performance of Algorithm 2. We constructed ten test data sets of tissue groupings; four sets having three tissues each, three sets having four tissues each, and three sets having five tissues each. Only those tissues having multiple samples were considered in choosing these groupings. Furthermore, each of these ten tissue groupings were used to make two sets: one set containing all 7,070 genes, and another set containing only nonnegative expressions across all tissues in that set.

2.2.7.2 Other Algorithms We compared our unsupervised biclustering algorithms to three other publicly available [33] biclustering (co-clustering) algorithms:

- Euclidian co-clustering algorithm (further referred to as **CC-e**) [27],
- Minimum squared residue co-clustering algorithm (**CC-r**) [27], and
- Information theoretic co-clustering algorithm (**CC-i**)[34].

These three algorithms represents recent algorithmic implementations of two classes of biclustering methodologies: (i) algorithms based on minimization of some residue measure (e.g., (2.3), (2.5)), also utilized in [24, 27, 48, 113], and (ii) information theoretic-based methods, see [34, 35]. Similar to the algorithms proposed in this paper, these methods are also optimization-based, but emphasize a different objective function metric in their respective optimization problems.

2.2.7.3 Environments and Parameter Values Computational testing was performed on multiple platforms. Both heuristic algorithms were compiled using Microsoft Visual Studio .NET 2003 and were run on Windows XP with a Pentium 4, 2.4GHz processor and 2GB of RAM. The source code for the three biclustering algorithms (**CC-e**, **CC-i**, and **CC-r**) was compiled with the GNU GCC compiler (version 4.1.2) and run on a Unix platform with an AMD Opteron 240, 1.394GHz speed processor using 4GB of RAM.

In testing heuristic **H1** we set the parameter of separation $t = 3$ in order to strongly differentiate the resulting clusters. The callable library of CPLEX 9.0 [56] was used to both

formulate the problem instances as well as to perform the optimization. In order to speed up the running time we adjusted some of the CPLEX MIP default settings, relying largely upon the *STOP* tool described in [7] to aid our knowledge of what adjustments to make. *STOP* is an open-source tool that can suggest solver settings to tune based upon one or more problem instances. With this knowledge, we set the MIP Emphasis parameter to emphasize *feasibility over optimality*, which encourages CPLEX to focus on finding a feasible integer solution quickly. We also adjusted two cutting plane parameters, setting the Mixed Integer Rounding Cuts parameter to *aggressive* and the Fractional Cuts to *off* (so that no fractional cuts would be generated by CPLEX).

Concerning branching rules, we chose to have CPLEX implement the *alternative best estimate* option for deciding subsequent nodes on which to branch, and we set the Dive parameter to *traditional dive*. The Variableselect parameter, which decides on which variable we branch, was set to *strong branching*. Additionally, we made use of the RINS and node heuristics of CPLEX to help quickly locate other feasible solutions. Both the RINS and node heuristic parameters were set to activate *every iteration*, and we restricted the number of nodes on which the RINS heuristic could operate to *5,000*. Finally, because of our belief that sample variables have a higher priority than feature variables in terms of branching strategies, we configured CPLEX to give greater priority to branching on sample variables over feature variables.

In our testing of the **H2** heuristic, we set *BeginFlopCheck* to 10 iterations and *PercentCutOff* to 0.8. Thus, after 10 iterations, if a feature had alternated more than 80% of the time, that feature was considered an outlier and no longer available for that trial. These parameters were chosen so as to refrain from considering a feature as an outlier unless multiple iterations passed and the feature moved around frequently. Moreover, limited testing revealed that the algorithm's convergence was not overly sensitive to these parameters. Ranges of *BeginFlopCheck* parameter from 5 to 15, and from approximately 60% to 95% for *PercentCutOff*, resulted in convergence of the algorithm. Finally, in an effort to provide a more direct comparison with the three co-clustering algorithms (**CC-e**, **CC-i**, and **CC-r**), we set the parameter of separation t to 0 for the **H2** heuristic, so that, in effect, t had no influence on our algorithm.

Table 2.3: Performance of main MIP (Section 2.2.5.3) on synthetic data

m	**n**	**r**	**seconds**
4	4	2	0.016
8	8	2	0.016
16	16	2	0.047
32	32	2	0.172
64	64	2	0.75
128	64	2	6.735
128	128	2	2.2
256	256	2	20.208
512	256	2	97.545
1,024	256	2	555.395
2,048	256	2	842.359
4,096	256	2	3,534.412
6	6	3	0.093
12	12	3	1.437
24	24	3	9.578
288	24	3	513.794
576	24	3	751.184
1,152	24	3	1,242.25
2,304	24	3	4,791.955
8	8	4	1.485
16	16	4	37.391
24	24	4	104.313
48	24	4	139.891
96	24	4	1,133.384

2.2.7.4 Results We next present the computational results we found using the aforementioned algorithms and parameters.

MIP Results on Synthetic Data Sets. In this section we describe our application of the main MIP (2.23), (2.21) – (2.22) and (2.29) – (2.33) to some synthetic instances we generated according to Section 2.2.7.1 (**Synthetic Data Sets**). The size of the synthetic instances, as well as the time required to solve them to optimality, are reported in Table 2.3. Every generated synthetic instance appearing in Table 2.3 was successfully solved to optimality by CPLEX 9.0 [56]. Unfortunately we were not able to experience the same level of success on subsets of the HuGE data set using the main MIP in conjunction with CPLEX 9.0, and thus we employed Algorithms 1 and 2 were necessary to perform the unsupervised biclustering.

It is important to note that the results on synthetic instances indicate the potential for utilizing the main MIP for biclustering rather large instances of data with 0–1 values. This situation may appear, for example, in the constraint matrix of mathematical programs. Biclustering could then correspond to finding an arrow-head structure within these matrices, which could prove very useful in the application of decomposition approaches (e.g., Benders', Dantzig-Wolfe) [67].

MIP-based Heuristic Results on Subsets of HuGE Data Set. The MIP-based Heuristic **H1** recovered significant biclusters with the subsets of HuGE data chosen as detailed in Section 2.2.7.1(**Huge Data Sets**). In order to perform this biclustering, it was helpful to implement a three hour time limit for the first iteration (in conjunction with an optimality gap setting of 1%), followed by two-hour time limits for each subsequent iteration (and no optimality gap). As mentioned previously, the parameter of separation was chosen as $t = 3$; this high setting of t yielded tight clusters.

The developed MIP-based heuristic performs fairly well for $r = 2$ and $n = 20$ to 30 samples. Two heatmaps identifying exemplary clusterings with no errors and $r = 2$ are included in Figures 2.3a and 2.3b. Figure 2.3a depicts the results of **H1** on the HuGE data subset including all of the (6) liver and (11) brain samples, using only those features with nonnegative values (so that the instance had $m = 1,534$ features and $n = 17$ samples). The resulting biclusters are characterized by liver tissue in the first 6 samples and 12 features, while the

remaining 11 samples and 46 features are characterized by brain tissue. **H1** and **CC-i** have no errors on this data set, while **CC-e** and **CC-r** have seven and three errors, respectively.

Figure 2.3b displays the results of **H1** on the HuGE data subset including all liver and muscle samples with nonnegative feature entries, giving $m = 2,097$ features and $n = 12$ samples. **H1** identified 6 samples and 24 features representing liver tissue, and 6 samples and 35 features representing muscle tissue. As in the previous data set, **H1** performed perfectly, while **CC-e**, **CC-i CC-r** have six, three and three errors, respectively.

Unfortunately, the performance of **H1** substantially declined when $r > 2$. We also note that certain tissue subsets of the HuGE data simply did not lend themselves well to unsupervised biclustering. On such subsets, our clusterings would typically misclassify one or more samples; we believe this outcome was likely due to similarity of samples across tissue types.

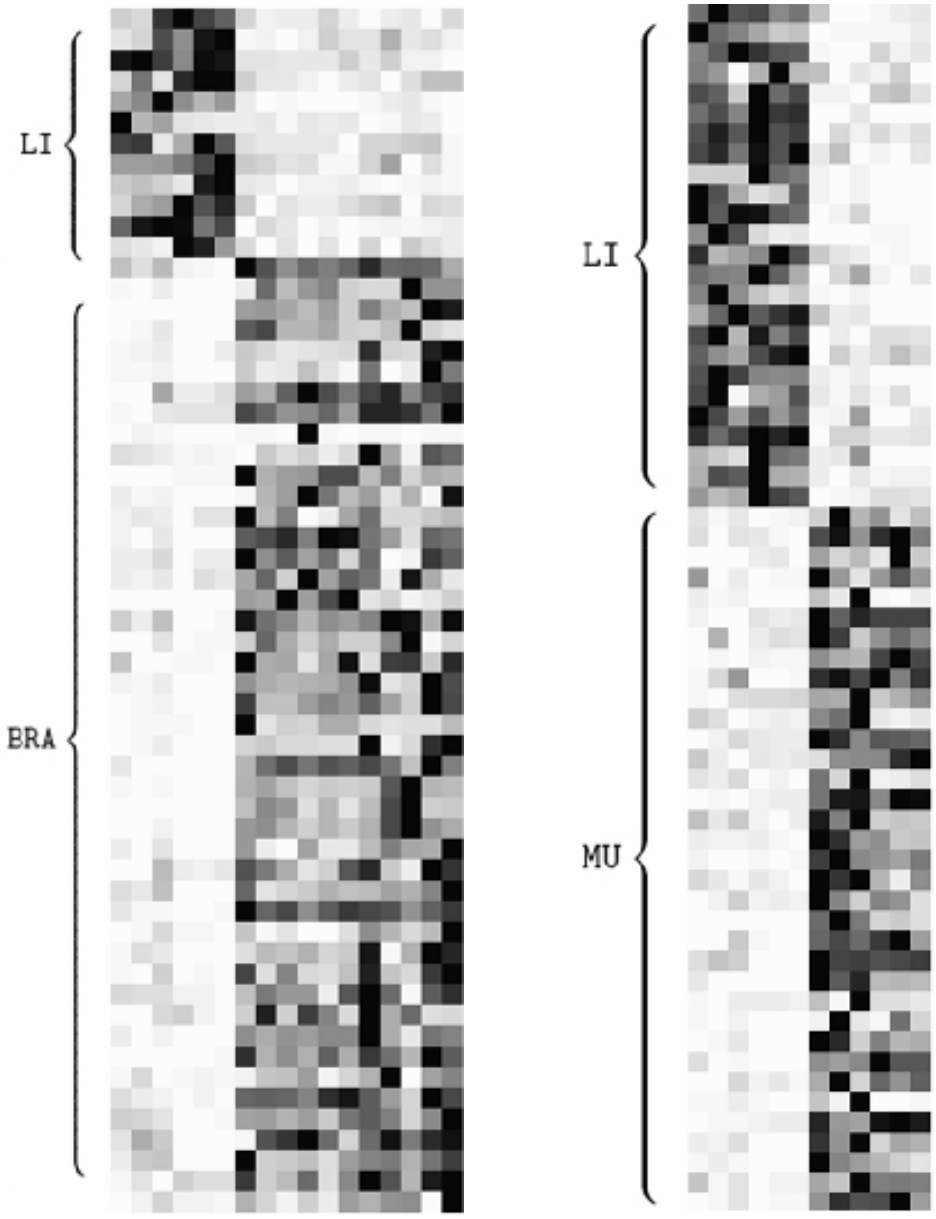
Table 2.4: Computational results of three biclustering algorithms vs. **H2** on HuGE data

Organs	**S**	**F**	**Cce**	**Cci**	**CCr**	**H2**
BRA-LI-MU$^+$	23	1,309	13 (43.5)	**0 (100)**	8 (65.2)	**0 (100)**
BRA-LI-MU	23	7,070	12 (47.8)	—	3 (87.0)	**0 (100)**
KI-MYO-VU$^+$	11	3,245	4 (63.6)	3 (72.7)	**0 (100)**	6 (45.5)
KI-MYO-VU	11	7,070	5 (55.6)	—	**3 (72.7)**	4 (63.6)
BRA-BRE-LU$^+$	19	1,602	8 (57.9)	3 (84.2)	4 (79.0)	**2 (89.5)**
BRA-BRE-LU	19	7,070	7 (63.2)	—	8 (57.9)	**4 (79.0)**
KI-MU-OV$^+$	14	2,440	7 (50.0)	**3 (78.6)**	7 (50.0)	**3 (78.6)**
KI-MU-OV	14	7,070	6 (57.1)	—	5 (64.3)	**0 (100)**
% Correct $r = 3$			52.2/53.7	**86.6**/—	71.6/71.6	83.6/**85.8**
BRA-BRE-KI-MU$^+$	25	1,389	10 (60.0)	2 (92.0)	8 (68.0)	**0 (100)**
BRA-BRE-KI-MU	25	7,070	9 (64.0)	—	11 (56.0)	**4 (84.0)**
LI-LU-PR-VU$^+$	19	2,301	11 (42.1)	**4 (79.0)**	7 (63.2)	6 (68.4)
LI-LU-PR-VU	19	7,070	11 (42.1)	—	8 (57.9)	**6 (68.4)**
BRE-KI-PL-PR$^+$	14	2,564	8 (42.9)	6 (57.1)	6 (57.1)	**2 (85.7)**
BRE-KI-PL-PR	14	7,070	7 (50.0)	—	6 (57.1)	**4 (71.4)**
% Correct $r = 4$			50.0/51.7	79.3/—	63.8/60.3	**86.2/81.0**
BRA-BRE-END-LI-LU$^+$	27	1,377	16 (40.7)	13 (51.9)	12 (55.6)	**9 (66.7)**
BRA-BRE-END-LI-LU	27	7,070	12 (55.6)	—	9 (66.7)	**8 (70.4)**
BRA-KI-MU-MYO-VU$^+$	28	1,429	16 (42.9)	**4 (85.7)**	9 (67.9)	12 (57.1)
BRA-KI-MU-MYO-VU	28	7,070	13 (53.6)	—	**7 (75.0)**	8 (71.4)
KI-LU-MU-PL-PR$^+$	24	2,048	13 (45.8)	12 (50.0)	9 (62.5)	**7 (70.8)**
KI-LU-MU-PL-PR	24	7,070	12 (50.0)	—	9 (62.5)	**2 (91.7)**
% Correct $r = 5$			43.0/48.1	63.3/—	62.0/65.2	**64.6/70.9**
Overall % Correct			48.0/51.0	75.5/—	65.7/65.9	**77.0/78.7**

Iterative Heuristic Results on Subsets of HuGE Data Set. As described in Section 2.2.7.1 (**Huge Data Sets**), we created subsets of tissue groupings from the HuGE data set to test the performance of **H2**. After obtaining these results, we then compared the performance of **H2** with those of publicly available biclustering algorithms (see Section 2.2.7.2). The computational results have been summarized in Table 2.4. The rightmost four columns contain the number of *misclassifications* per algorithm, followed by the percentage of *correct classifications*. A "+" superscript on the data set name indicates that only positive genes from the HuGE data set were used, whereas its absence indicates that all genes were included. The "% correct r =" heading contain two percentages for the specified level of r: the first is the percent correct for only nonnegative data, while the second is for all data.

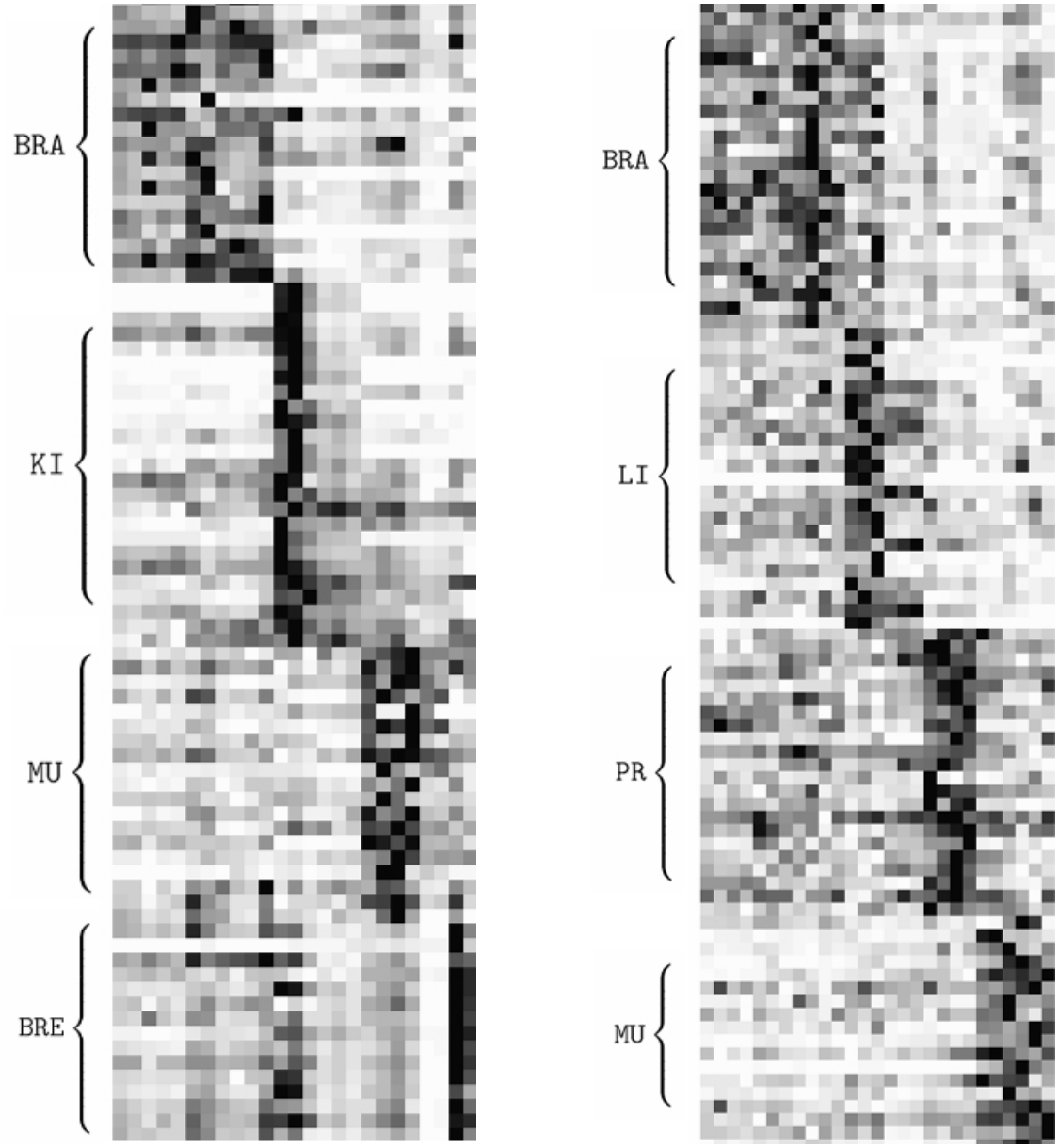
From Table 2.4 we can see that in 13 of the 20 data sets, **H2** had the highest percentage of correct classifications, while in another two data sets, **H2** tied for the highest percentage. Only in five of the 20 data sets was **H2** outperformed, and of these, only once (KI-MYO-VU without negatives) was it outperformed by all of the three other algorithms. As noted previously, the **CC-i** algorithm only operates on nonnegative data, and so its performance could not be measured on data sets containing negative entries.

Further performance measures are evidenced in the rows with the "%correct r =" heading. For each value of r ($r = 3$, $r = 4$, and $r = 5$, as well as for the overall case), the percentage of correct classifications for each heuristic was calculated. The first percentage reported is the performance on nonnegative data, while the second percentage is the performance on all data. Among the four algorithms, **H2** had the highest percentage of correct classifications on seven of the eight measures, including both of the overall measures. Figures 2.4a and 2.4b are heatmaps of exemplary biclusterings located in the HuGE test data. Some concluding remarks concerning the checkerboard pattern are presented in Section 2.4.1.



(a) 11 BRA and 6 LI samples (b) 6 LI and 6 MU samples

Figure 2.3: Heatmaps illustrating biclusters found using **H1** on subsets of HuGE data



(a) 11 BRA, 2 BRE, 6 KI, and 6 MU samples

(b) 11 BRA, 6 LI, 6 MU, and 4 PR samples

Figure 2.4: Heatmaps illustrating biclusters found using **H2** on subsets of HuGE data

2.3 ORDER-PRESERVING SUBMATRIX PATTERNS

2.3.1 Acknowledgment

The following content is reproduced with kind permission from The Institute for Operations Research and the Management Sciences: A.C. Trapp and O.A. Prokopyev, "Solving the Order-Preserving Submatrix Problem via Integer Programming," *INFORMS Journal on Computing*, 22 (3), pp. 387-400, 2010.

2.3.2 Introduction

Another important concept in data mining is tracking trends, for example in disease progression or stock behavior over time. Continuing with the theme of generalizing traditional clustering approaches to consider local patterns, we now turn our attention to another type of biclustering pattern. Namely, we consider the problem of finding an embedded matrix (submatrix) within a given data set that exhibits coherent increasing and decreasing trends across the samples (columns) of each feature. We are particularly interested in locating the largest such submatrix.

Instead of simultaneously identifying biclusters as with the checkerboard pattern in Section 2.2, biclusters can also be identified on a one-by-one basis [20]. If it is desirable to locate additional biclusters in the data, the biclustering procedure can then be repeated. If this is the case, the corresponding features and samples of the newly identified bicluster are either amputated from the data or their values masked with random numbers [24]. Otherwise, the remaining data are not further processed and are simply considered as outliers. The *order-preserving* submatrix is a biclustering pattern that may be considered in this manner, introduced by Ben-Dor et al. [8, 9] in the context of detecting coherent trends in gene expression data. They both define the OPSM pattern and introduce an efficient greedy algorithm that quickly locates statistically significant OPSMs.

Given the data set $A = (a_{ij})_{m \times n}$, the OPSM problem consists of identifying a submatrix of k rows and ℓ columns from the original data matrix in which there exists a permutation of the selected columns such that in every selected row the values corresponding to selected

columns are strictly increasing. More formally, let $\mathcal{F}_0$ be a set of row indices $\{f_1, f_2, \ldots, f_k\}$. Then there exists a permutation of a subset of column indices $\mathcal{S}_0 = \{s_1, s_2, \ldots, s_\ell\}$ such that for all $i = 1, \ldots, k$ and $j = 1, \ldots, \ell - 1$ we have that:

$$a_{f_i,s_j} < a_{f_i,s_{j+1}}. \tag{2.37}$$

The corresponding submatrix $(\mathcal{F}_0, \mathcal{S}_0) \in \mathbb{N}^{k \times \ell}$ is called *order-preserving*. A data set and a corresponding column permutation that induces a 3×4 OPSM is depicted in Figure 2.5.

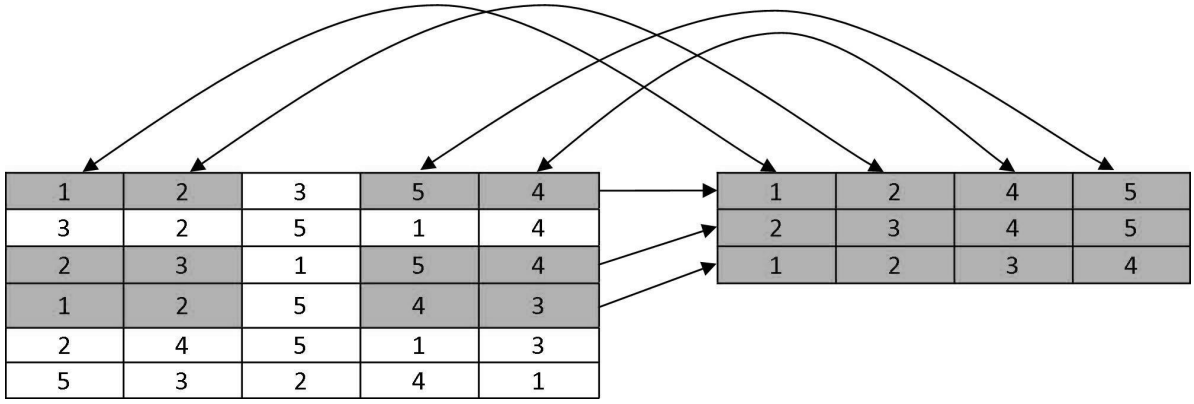


Figure 2.5: Columns of data matrix permuted to induce 3×4 OPSM

The appearance of these types of patterns in real-life data sets has a natural interpretation. Suppose we have a patient's DNA microarray data set where each sample corresponds to a particular stage of a disease. Then there is likely a subset of features that are co-expressed with the disease progression. Considering the relative orderings of the expression levels gives an indication of the coherent tendencies, or trends, across the sample set. A similar situation occurs whenever we consider data representing some temporal progression: data from drug treatment, data from nominally identical exposure to environmental effects, data with some genetic abnormalities, etc. [9, 20].

We are aware of several attempts in the recent literature to establish efficient solution approaches for the OPSM problem. Cheung et al. [26] study the OPSM problem and propose some additional extensions, employing a specific data structure together with an algorithm that is able to discover all OPSMs. They further discuss some pruning methods

that aid in eliminating portions of the search space, and report on a number of computational experiments. Hochbaum and Levin [51] develop a 5-approximation algorithm and a 3-approximation algorithm for the OPSM. They approach the OPSM problem from its complementary viewpoint that deletes the least number of non-promising entries in the original matrix to obtain an OPSM. Unfortunately, no computational tests are provided.

Gao et al. [42] study the problem of locating OPSMs in massive sets of gene expression data. They argue that, while OPSMs in such data sets are of clear interest to biologists, most traditional clustering methods would completely overlook such submatrices with small row support (i.e., small number of rows in the OPSM in relation to the overall row count of the data set). They introduce the *KiWi* mining framework that relies on two parameters (k and w). In short, their algorithm uses a statistical metric to evaluate candidate patterns, keeping the k most promising that appear in the next w positions of the supporting sequences. While a heuristic, their approach substantially reduces the search space and problem scale, finding very large OPSMs embedded in massive real data sets.

The above approaches are all able to identify sizable OPSMs in data, however, the only performance guarantees provided by any of these studies are the approximation algorithms of Hochbaum and Levin [51]. This serves to underscore our goal of establishing an approach that is guaranteed to identify order-preserving biclusters of maximum size (according to specified performance criterion). In Section 2.3.4 we provide a general linear mixed 0–1 programming formulation that can be solved using standard solvers such as CPLEX, while in Section 2.3.5 we demonstrate an alternative approach that iteratively solves a series of smaller linear 0–1 programs in conjunction with valid inequalities and other improvements. Section 2.3.6 details our computational experiments on both synthetic and real biological data together with some additional algorithmic enhancements, while Section 2.4.2 contains some concluding remarks. We next discuss computational complexity issues related to finding order-preserving submatrices.

2.3.3 Computational Complexity Issues

The decision version of the OPSM problem consists of checking whether there exists a $k \times \ell$ order-preserving submatrix $(\mathcal{F}_0, \mathcal{S}_0)$ for given integers k, ℓ and input data matrix $A \in \mathbb{R}^{m \times n}$. More formally, the decision version of the OPSM problem is defined as follows:

Instance: A real-valued data matrix $A = (a_{ij})_{m \times n}$ and two positive integers $k \leq m$ and $\ell \leq n$.

Question: In A, is there an order-preserving submatrix of size k-by-ℓ? That is, we need to check whether there is a set of row indices $\mathcal{F} = \{f_1, \ldots, f_k\}$ and a sequence of columns indices $\mathcal{S} = \{s_1, \ldots, s_\ell\}$ such that for all $1 \leq i \leq k$ and $1 \leq j \leq \ell - 1$:

$$a_{f_i, s_j} < a_{f_i, s_{j+1}}. \tag{2.38}$$

Theorem 4. *[9] The decision version of the OPSM problem is NP-complete.*

In the optimization version of the problem we consider finding the largest OPSM according to the number of elements $|\mathcal{F}_0| \cdot |\mathcal{S}_0|$. Because of the NP-completeness of the decision version, the optimization version of the OPSM problem is clearly NP-hard. We refer the reader to [43] for background on computational complexity theory.

We can also look at the computational complexity of the OPSM problem from the point of view of *parameterized complexity theory* [37]. In this theory a problem is *fixed parameter tractable* (FPT) with respect to parameter k if there exists a solution running in $f(k) \times \ell^{O(1)}$ time, where ℓ is the input size of the problem and f is a function of k that is independent of ℓ. In other words, the problem is in FPT with respect to parameter k if it is polynomially solvable for the fixed value of k. Next we show that OPSM is FPT with respect to the number of columns n and FPT with respect to the number of rows m. Though the proofs below are constructive, the enumerative algorithms described there can be utilized for solving the OPSM problem only in the case of *extremely small values* of n or m, respectively.

Proposition 2. *The OPSM problem is polynomially solvable if the number of columns n in input data matrix $A \in \mathbb{R}^{m \times n}$ is fixed.*

Proof. Because n is fixed, we can enumerate all 2^n subsets of the original set of columns indices. For each subset of size r ($r = 1, \ldots, n$) we can consider all $r!$ permutations of indices. Then for each row we can check whether the selected permutation of a particular subset of column indices forms an increasing sequence of values. As observed in [9], this can be done in $O(mr)$ time, resulting in a $\sum_{r=1}^{n} \frac{n!}{(n-r)!} O(mr)$ algorithm for solving the OPSM problem, which is polynomial for each fixed value of n. ∎

Proposition 3. *The OPSM problem is polynomially solvable if the number of rows m in input data matrix $A \in \mathbb{R}^{m \times n}$ is fixed.*

Proof. Because m is fixed, we can enumerate all 2^m subsets of the original set of row indices. Then for each subset, assuming that all considered rows are in the resulting submatrix, the objective is to maximize the number of selected columns. Construct a directed graph $G = (\mathcal{N}, \mathcal{A})$ as follows. For each column j introduce a node $j \in \mathcal{N}$. Introduce an arc $(j_1, j_2) \in \mathcal{A}$ if and only if $a_{ij_1} < a_{ij_2}$ for every row i in the subset of the considered rows. The resulting graph G is acyclic. The longest directed path in G then corresponds to the maximum number of columns included in the submatrix. The problem of finding the longest path in an acyclic graph is polynomially solvable (see, e.g., [2]), which implies the necessary result. ∎

The definition of OPSM can be generalized to finding *any* fixed pattern. For a fixed vector $w = \{w_1, \ldots, w_{n-1}\}$, where $w_j \in \{-1, +1\}$ for all $j = 1, \ldots, n-1$, consider the decision version of the following problem, which is further referred to as the w-OPSM problem:
Instance: A real-valued data matrix $A = (a_{ij})_{m \times n}$ and two positive integers $k \leq m$ and $\ell \leq n$.
Question: In A, is there an order-preserving submatrix of size k-by-ℓ satisfying pattern w? That is, we need to check whether there is a set of row indices $\mathcal{F} = \{f_1, \ldots, f_k\}$ and a sequence of columns indices $\mathcal{S} = \{s_1, \ldots, s_\ell\}$ such that for all $1 \leq i \leq k$ and $1 \leq j \leq \ell - 1$:

$$w_j \cdot a_{f_i, s_j} < w_j \cdot a_{f_i, s_{j+1}}. \tag{2.39}$$

We can observe from (2.39) that $w_j = 1$ corresponds to up regulation between columns j and $j + 1$, (the "up" pattern), whereas $w_j = -1$ corresponds to down regulation between

the same columns (the "down" pattern). If $w_j = 1$ for all $j = 1, \ldots, \ell - 1$ then we obtain the original OPSM problem that searches for a permutation of columns obeying a strictly increasing order. In general, however, the w-OPSM and OPSM problems are not the same. For example, consider the matrix:

$$A = \begin{pmatrix} 4 & 5 & 2 \\ 3 & 7 & 6 \end{pmatrix}. \tag{2.40}$$

If we are looking for the strictly increasing pattern then the largest submatrix consists only of two columns (a single "up" relationship). However, in the case of an {"up","down"} pattern, the final answer is the whole matrix. The question we now ask is whether the w-OPSM problem is difficult for all possible patterns. Is there any pattern w that can be discovered in polynomial time? Unfortunately, it can be shown that for any fixed pattern the problem of finding the largest submatrix satisfying this pattern is NP-hard.

Theorem 5. *w-OPSM is NP-complete for any fixed pattern w.*

Proof. Our proof is similar to the related result for the OPSM problem presented in [9]; we also use the reduction from the *Balanced Complete Bipartite Subgraph* problem, which is known to be NP-complete (see [43]):

Instance: Bipartite graph $G = (V, U, E)$, positive integer $K \leq |V|$.

Question: Are there two sets $\bar{V} \subseteq V$, $\bar{U} \subseteq U$ such that $|\bar{V}|=|\bar{U}|=K$ and such that $u \in \bar{U}$, $v \in \bar{V}$ implies that $\{u, v\} \in E$?

For a given pattern w let H be a set of indices such that:

$$H = \{j : \ w_{j-1} = -1, w_j = 1, 1 < j < \ell\} \cup \{1 : \text{ if } w_1 = 1\} \cup \{\ell : \text{ if } w_{\ell-1} = -1\}. \tag{2.41}$$

Given a bipartite graph $G = (V, U, E)$ let $m = |V|$ and $n = |U| + |H|$. Define the matrix $A = (a_{ij})_{m \times n}$ as follows: *(i)* $a_{ij} = -1$, if $(i, j) \notin E$, *(ii)* $a_{ij} = j$, if $(i, j) \in E$, and *(iii)* $a_{ij} = -1$, if $n - |H| + 1 \leq j \leq n$, where $i = 1, \ldots, |V|$.

Next we show that G contains a balanced complete bipartite subgraph of size $\ell - |H|$ if and only if the matrix A contains an w-order-preserving submatrix Q of size $(\ell - |H|)$-by-ℓ.

Assume that there exists an order-preserving submatrix of size $(\ell - |H|)$-by-ℓ that follows pattern w. It can be verified from (2.39) and (2.41) that the number of columns that has one

or more elements equal to -1 is at most $|H|$. In other words, only columns in positions $j \in H$ in the final submatrix may have elements equal to -1. All other columns can not contain "-1" elements, which implies that we have $\ell - |H|$ columns with only positive elements. By construction, these $\ell - |H|$ rows and $\ell - |H|$ columns with all positive entries will correspond to a complete bipartite subgraph in G.

To show the other direction, assume that there exists a complete bipartite subgraph in G of size $\ell - |H|$. Next we show that the constructed matrix A contains a submatrix Q corresponding to pattern w of size $(\ell - |H|)$-by-ℓ.

Let I, J be the index sets corresponding to nodes from $\bar{V}, \bar{U}$ in the complete bipartite subgraph in G, respectively, i.e. $(i, j) \in E$. We also assume that J is sorted in increasing order. Let $R = \{|U|+1, \ldots, |U|+|H|\}$, i.e., R is the set of indices corresponding to columns with all elements equal to -1. Thus in Q we keep only rows from A that correspond to nodes from I. The following procedure constructs the respective w-order-preserving submatrix Q.

a) If $w_1 = 1$, then add column $|U| + 1$ to Q. Remove index $|U| + 1$ from R.

b) If $w_1 = -1$, then let $j^* \in \arg\max_{j \in J} j$. Add column j^* to Q and remove j^* from J.

c) For every h such that $1 < h < \ell$:

   (1) if $w_{h-1} = 1$ and $w_h = 1$, add $j^* \in \arg\min_{j \in J}\{j\}$ to Q and remove j^* from J.

   (2) if $w_{h-1} = -1$ and $w_h = -1$, add $j^* \in \arg\max_{j \in J}\{j\}$ to Q and remove j^* from J.

   (3) if $w_{h-1} = 1$ and $w_h = -1$, add $j^* \in \arg\max_{j \in J}\{j\}$ to Q and remove j^* from J.

   (4) if $w_{h-1} = -1$ and $w_h = 1$, add any r^* from R to Q and remove r^* from R.

d) If $w_{\ell-1} = 1$, add column $j^* \in \arg\min_{j \in J}\{j\}$ to Q and remove j^* from J.

e) If $w_{\ell-1} = -1$, add any column r^* from R to Q and remove r^* from R.

The key intuition is that at every step of the construction we look "ahead" one step and add columns to Q from J or R in a such a way that we will have a column with a smaller or larger value (depending on the given pattern w) at the next step of the construction. If we observe an "up"-"up" pattern (see item "(c)-(1)") in two consecutive columns in given w, then we add to Q a column with the smallest available value from J. This implies that we can continue the construction of Q adding columns from J with larger values at the next step of the procedure. Likewise, if we observe "down"-"down" or "up"-"down" patterns (see

items "(c)-(2)" and "(c)-(3)"), we add to Q a column with the largest available value from J, allowing the next step of the construction. A similar explanation can be provided for every step in items "(a)", "(b)", "(c)-(4)", "(d)" and "(e)". It is rather easy to verify that the construction is valid and that the obtained submatrix satisfies (2.39). ∎

We next discuss two exact approaches for solving the OPSM problem that use mathematical programming. Section 2.3.4 covers a general linear 0–1 programming formulation, while our main emphasis is on an alternative approach that iteratively solves a series of smaller linear 0–1 programs for a restricted version of the initial OPSM problem. It is detailed in Section 2.3.5, where we also discuss valid inequalities and other improvements to further enhance this approach. In both cases, we follow the approach of [9] in that we consider only the relative ordering (i.e., the ranks) of the expression levels for each gene over permutations of the samples, thereby eliminating any potential scaling issues.

2.3.4 Mathematical Modeling of OPSM: General IP Formulation

Our initial approach to solving the OPSM problem is with a general linear 0–1 programming formulation. By filling ordered positions with columns from input matrix A, we attempt to find a permutation of the original columns so that the order induced across a subset of rows is strictly increasing; of all such column and row subsets, we wish to find the largest such submatrix. That is, we assume there are $1, \ldots, n$ positions that may be filled by any of the n columns. We then attempt to fill the first $K \leq n$ positions with columns such that all included rows maintain this increasing order.

To facilitate this concept, we introduce a binary variable s_{jk} for each column j and possible position k, with $s_{jk} = 1$ implying that column j has been selected for position k. Figure 2.6 illustrates the relationship between s_{jk} variables and column-position interactions (for the sake of illustration, we assume that all rows are included in Figure 2.6, and that columns two and five are not involved in this OPSM).

For each row we introduce a binary variable x_i, with $x_i = 1$ indicating that row i is selected for the OPSM. Likewise, for each column we introduce a binary variable y_j, with

$y_j = 1$ implying that column j is selected in the OPSM. Finally, we define binary variables z_{ij} as the product of each row and column combination, that is, $z_{ij} = x_i y_j$. Thus, a single binary z_{ij} variable corresponds to each entry in the input matrix A. These variables are used in the following GF-OPSM formulation for the OPSM problem.

$$\text{(GF-OPSM)}: \ \max \sum_{i}^{m} \sum_{j}^{n} z_{ij} \tag{2.42}$$

subject to

$$x_i + \sum_{u=1}^{k} s_{ju} + \sum_{v=k+1}^{n} s_{\ell v} \leq 2 \quad \forall\ j, \ell \tag{2.43}$$

$$\text{such that } a_{ij} \geq a_{i\ell},\ i = 1, \ldots, m,\ k = 1, \ldots, n-1,$$

$$\sum_{j=1}^{n} s_{jk} \geq \sum_{j=1}^{n} s_{j,k+1}, \quad k = 1, \ldots, n-1, \tag{2.44}$$

$$\sum_{k=1}^{n} s_{jk} = y_j, \quad j = 1, \ldots, n, \tag{2.45}$$

$$\sum_{j=1}^{n} s_{jk} \leq 1, \quad k = 1, \ldots, n, \tag{2.46}$$

$$z_{ij} \leq x_i,\ z_{ij} \leq y_j, \quad \forall\ i, j, \tag{2.47}$$

$$x_i \in \{0,1\},\ y_j \in \{0,1\},\ z_{ij} \in \{0,1\} \ \ \forall\ i, j. \tag{2.48}$$

The objective function of this formulation maximizes the number of entries included in the OPSM, ensuring that we find the largest OPSM by area. Constraint set (2.43) enforces the condition of strictly increasing order by requiring that, if row i is in the final solution, and $a_{ij} \geq a_{i\ell}$ holds, then column j cannot appear to the left of column ℓ in the final permutation of columns. Constraints (2.44) ensure that, if position k is not filled in the final permutation (i.e., there are less than k columns in the final submatrix), then positions $k+1, \ldots, n$ are also not filled in the final permutation. These constraints enforce a decision hierarchy that

removes symmetry (and thus duplicate solutions) from the problem. Constraints (2.45) ensure that, if column j is chosen, it fills exactly one position in the final permutation. On the other hand, constraints (2.46) ensure that at most one column can fill any one position in the final permutation. Finally, constraints (2.47) ensure that element (i, j) may be in the final OPSM if and only if both row i and column j are included.

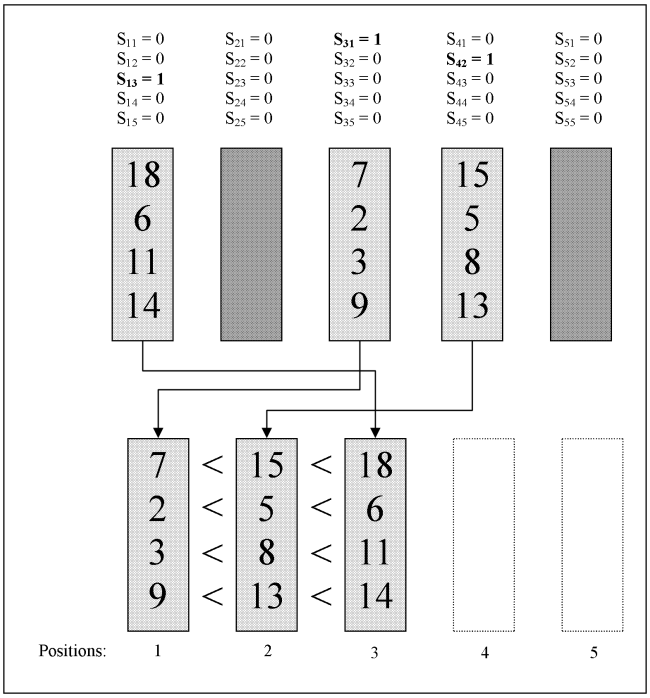


Figure 2.6: Relationship between s_{jk} variables and column-position interactions

Formulation GF-OPSM has $m+n+mn+n^2$ variables in total, of which $m+n+n^2$ are binary, and at most $2n+2mn+(n-1)+\frac{1}{2}mn(n-1)^2$ constraints. Unfortunately, this formulation did not perform well in computational testing, motivating the subsequent approach.

2.3.5 Mathematical Modeling of OPSM: Compact Formulation

The key idea of this approach is the formulation of a smaller linear 0–1 program that corresponds to a restricted version of the original OPSM problem. Algorithmically, this formulation can then be solved in an iterative manner, in conjunction with derived valid inequalities and other enhancements, to find a solution to the initial problem.

2.3.5.1 Compact Formulation One of the main difficulties of the GF-OPSM formulation of Section 2.3.4 is the complexity of determining the proper column placements. Instead of allowing for column permutation, we next temporarily consider the simpler problem of identifying the largest submatrix exhibiting a simple increasing pattern with fixed column order. Under this scenario, the OPSM problem simplifies to simultaneously finding:

- a set of row indices $\{f_1, f_2, \ldots, f_k\} \subseteq \mathcal{F}_0$,
- a set of column indices $\{s_1, s_2, \ldots, s_\ell\} \subseteq \mathcal{S}_0$, such that $s_1 < s_2 < \ldots < s_\ell$, and
- for all $i = 1, \ldots, k$ and $j = 1, \ldots, \ell - 1$ we have that $a_{f_i,s_j} < a_{f_i,s_{j+1}}$; i.e., we do not permit *inversions* on values corresponding to the set of column indices on included rows.

These conditions motivate the Compact Formulation, or CF-OPSM. As in GF-OPSM, we introduce binary variables x_i for each row, binary variables y_j for each column, and binary variables z_{ij} for each row and column combination. Given this setup, finding the largest submatrix obeying the strictly increasing pattern can be found by solving the following 0–1 programming problem:

$$\text{(CF-OPSM)}: \ \max \sum_{i}^{m} \sum_{j}^{n} z_{ij} \tag{2.49}$$

subject to

$$z_{ij} + z_{ik} \leq x_i \ \ \forall\, j < k \text{ and } a_{ij} \geq a_{ik}, \ i = 1, \ldots, m, \tag{2.50}$$

$$z_{ij} \geq x_i + y_j - 1, \ z_{ij} \leq x_i, \ z_{ij} \leq y_j, \quad \forall\, i, j, \tag{2.51}$$

$$x_i \in \{0, 1\}, \ y_j \in \{0, 1\}, \ z_{ij} \in \{0, 1\} \ \ \forall\, i, j. \tag{2.52}$$

The objective function of CF-OPSM is identical to that used in GF-OPSM, aiming to find the largest OPSM by area. Constraint set (2.50) ensures that, for any of included row i's entries taken pairwise, with both $j < k$ and $a_{ij} \geq a_{ik}$, then at most one of the two z_{ij} entries may be included in the OPSM. That is, for any row i included in the OPSM, constraints (2.50) prevent inversions on included columns. An alternative view of these constraints is that, when row x_i is included in the OPSM, they represent pairwise clique inequalities that are

sufficient to characterize the CF-OPSM polyhedron. Constraint set (2.51) ensures $z_{ij} = 1$ whenever both row i and column j are selected to be in the OPSM. Formulation (2.49) – (2.52) has $m + n + mn$ binary variables and at most $3mn + \frac{1}{2}mn(n-1)$ constraints.

2.3.5.2 Basic Iterative Algorithm Suppose the first row of A is in the final solution. If we then permute the columns of A so that the elements of the first row appear in increasing order, then the solution of CF-OPSM, together with the constraint $x_1 = 1$ will provide the largest OPSM that includes row 1. Repeating this approach on each subsequent row gives the largest submatrix for each respective row. Afterwards, assuming there is a single largest OPSM (we address the possibility of multiple optimal solutions later), then k rows will share the largest objective value; these rows constitute $\mathcal{F}_0$, and together with the set of included columns $\mathcal{S}_0$ form the largest OPSM of A. This motivates the following iterative algorithm.

Algorithm 3. *[Basic Iterative Algorithm]*

1. *Assign* $h := 1$.

2. *Permute columns of* A *such that the entries in row* h *occur in increasing order; call new matrix* $\hat{A}_h$.

3. *Generate formulation CF-OPSM for matrix* $\hat{A}_h$. *Add additional constraint* $x_h = 1$.

4. *Solve the corresponding linear 0–1 formulation, let* Z^h *be the optimal objective value, and store off the optimal solution for row* h.

5. ***If*** $h < m$, *assign* $h := h + 1$ *and* ***Go To*** *2.*

6. *Assign* $Z^* := \max_h Z^h$.

7. ***Return*** Z^*, *its corresponding optimal solution, and* ***Stop***.

For each row h, Algorithm 3 formulates and solves an instance of the CF-OPSM integer program, identifying the largest submatrix having an strictly increasing pattern according to permuted matrix A_h. Because we iteratively consider each row $h = 1, \ldots, m$, Algorithm 3 finitely terminates. At its conclusion, after searching over all rows $h = 1, \ldots, m$, Algorithm 3 returns the largest OPSM by area corresponding to input matrix A.

2.3.5.3 Valid Inequalities The pairwise constraints (2.50) are reminiscent of classical pairwise clique inequalities. Indeed, while it is known that pairwise clique inequalities are sufficient to represent the independent set polyhedron, they typically are not facet-defining [79]. However, these pairwise inequalities can be strengthened into facet-defining inequalities by identifying the *maximal* clique to which a given node belongs.

In a similar manner, pairwise inversion inequalities (2.50) can be strengthened. In general, if for row i the relationship $a_{ij_1} \geq a_{ij_2} \geq \ldots \geq a_{ij_k} \geq \ldots \geq a_{ij_\ell}$ holds, where $j_1 < j_2 < \ldots < j_k < \ldots < j_\ell$, then the corresponding inversion inequality:

$$z_{ij_1} + z_{ij_2} + \ldots + z_{ij_k} + \ldots + z_{ij_\ell} \leq x_i \tag{2.53}$$

is valid to impose on CF-OPSM. The set $a_{ij_1}, a_{ij_2}, \ldots, a_{ij_k}, \ldots, a_{ij_\ell}$ defines a *maximal* decreasing subsequence among the elements of row i if we cannot augment the current decreasing subsequence with an additional element. In this case we will refer inequality (2.53) as a *maximal inversion* inequality.

Theorem 6. *Maximal inversion inequalities are facets of the convex hull of integer solutions to the CF-OPSM polyhedron.*

Proof. For a given row i, let $C = \{j_1, j_2, \ldots, j_k, \ldots, j_\ell\}$ correspond to the column indices representing a maximal inversion of size ℓ on row i. Now if row i is selected for any optimal OPSM, then at most one of the elements j_k may be included in the OPSM. Thus the maximal inversion inequalities (2.53) are valid inequalities for the CF-OPSM polyhedron. To demonstrate that maximal inversion inequalities are facet-defining for the CF-OPSM polyhedron, which has full dimension, we need to identify $mn + m + n$ affinely independent, feasible points that satisfy the maximal inversion inequality at equality. Such a set of vectors can be constructed in the following manner.

The origin is feasible and trivially satisfies (2.53) at equality. The vectors consisting of $y_j = 1,\ j = 1, \ldots, m$, and all other components zero, give m more such vectors. Now for row i containing our maximal inversion, we can construct $\ell \leq m$ additional vectors using each of the ℓ elements of the maximal inversion by setting $x_i = 1$, $y_{j_k} = 1$, and $z_{ij_k} = 1$, with all other components zero. Let column j_q correspond to one of the $m - \ell$ elements not in the columns of the maximal inversion C. For each such column j_q, we can construct an additional vector by

taking $x_i = 1$, $y_{j_1} = 1$, $z_{ij_1} = 1$, $y_{j_q} = 1$, and $z_{ij_q} = 1$. In this manner we can construct $m - \ell$ additional affinely independent, feasible vectors satisfying (2.53) at equality. Finally, for each row $h \neq i$ ($n-1$ total), we can construct $(m+1)$ additional vectors as follows. One such vector is $x_h = 1$, with all other components zero; the other m vectors have the form $x_h = 1$, $y_j = 1$, and $z_{hj} = 1$ for $j = 1, \ldots, m$. This last step constructs an additional $(n-1) \times (m+1) = mn-m+n-1$ such vectors, giving a total of $1+m+\ell+(m-\ell)+nm-m+n-1 = mn+m+n$ affinely independent, feasible vectors satisfying (2.53) at equality. ∎

Because maximal inversion inequalities (2.53) define facets of the CF-OPSM polyhedron, this leads to the natural question of how to quickly find such relationships, i.e., maximal decreasing subsequences. An $O(n \log n)$ algorithm was given in [86], utilizing binary search, to solve the longest decreasing subsequence (LDS) problem; we outline the algorithm with the following pseudocode.

Algorithm 4. *[FindLDS]*

1. *Declare arrays* $M[n]$ *and* $P[n]$*, and variable* L*.*

2. *Assign* $M[0] := 0$ *and* $L = 0$*.*

3. *Assign* $j := 1$*.*

4. *Do binary search for largest* $q \leq L \ : \ X[M[q]] > X[j]$ *(**If** none exists, assign* $q := 0$*).*

5. $P[j] := M[q]$*.*

6. ***If*** $q = L$ ***Or*** $X[j] > X[M[q+1]]$*, assign* $M[q+1] := j$ *and* $L := \max(L, (q+1))$*.*

7. *Assign* $j := j + 1$*.*

8. ***If*** $j < n$*,* ***Go To*** *4.*

Let us briefly describe the key idea behind this method. Array X stores the sequence over which we want to identify the longest decreasing subsequence. For a given row h of A, $X[1] = a_{h1}, X[2] = a_{h2}, \ldots, X[n] = a_{hn}$. Array $M[q]$ holds the index k of the smallest value $X[k]$ such that $k \leq j$ and there is a decreasing subsequence of length q ending at $X[k]$. The predecessor array P contains the position of the predecessor of $X[k]$ in the longest

decreasing subsequence ending at $X[k]$. The algorithm uses binary search on the discrete interval $[0, L]$ to locate the index $M[q] < j$ corresponding to the largest value continuing the longest decreasing subsequence at index j. After making a single pass through all n elements, it finds the longest decreasing subsequence over the entire sequence.

Algorithm 4 generates maximal decreasing subsequences for row h of data matrix A that we can use to construct maximal inversion inequalities. In general, there may be an exponential number of maximal decreasing subsequences in data matrix A, and so while each subsequence yields a facet-defining inequality, locating all of them would likely be prohibitive. Instead, we propose a modification of Algorithm 4 to identify a single maximal decreasing subsequence passing through *each* element of a given sequence X.

For all elements $j = 1, \ldots, n$, apply Algorithm 4 and find the longest decreasing subsequence up to and including element j; store off this subsequence. Now apply the same algorithm in reverse (*FindLIS*), starting with element n, stepping through the FOR loop in reverse, until element j is reached. This will generate the longest *increasing* subsequence up to and including element j. Again, store off this subsequence. Combining the longest decreasing subsequence up to and including element j on a forward pass, together with the longest increasing subsequence up to and including element j on a reverse pass, gives the longest decreasing subsequence through each element $j = 1, \ldots, n$. For each element j, Algorithm 4 and its counterpart *FindLIS* each take $O(n \ log \ n)$ time to complete. With n elements in the sequence, this implies $O(n^2 \ log \ n)$ time. Because in the worst case we consider sequences corresponding to rows $h = 1, \ldots, m$, the above described approach to find maximal decreasing subsequences has an overall run time of $O(mn^2 \ log \ n)$.

2.3.5.4 Nodal Constraints An alternative set of valid inequalities called *nodal* constraints can effectively impose the necessary inversion restrictions currently enforced in CF-OPSM using pairwise clique inequalities (2.50) [74, 75]. Nodal constraints are defined in the context of a graph G, where a constraint for each node n_i is generated based on its neighborhood of adjacent nodes. This definition can easily be extended to our context of sequences, where for a given row i of A and its elements $a_{i1}, \ldots, a_{ij}, \ldots, a_{in}$, we define the *neighborhood* N_{ij} of element a_{ij} as the set of all elements a_{ik} such that either: (i) $k > j$ and

$a_{ik} \leq a_{ij}$, or (ii) $k < j$ and $a_{ik} \geq a_{ij}$. Based on this definition, for row i we introduce nodal constraints for each element a_{ij} as:

$$n_{ij} z_{ij} + \sum_{k \in N_{ij}} z_{ik} \leq n_{ij} x_i \; \forall \; j, \tag{2.54}$$

where we set the value n_{ij} as $n_{ij} = |N_{ij}|$. The nodal constraint (2.54) for element a_{ij} efficiently represents all of the inversion restrictions implied by pairwise clique inequalities (2.50). This is because, if variable $z_{ij} = 1$ for element a_{ij}, then constraint (2.54) forces the nonnegative variables z_{ik} corresponding to all neighbors $a_{ik} \in N_{ij}$ to be zero. However, if $z_{ij} = 0$, then no restrictions are forced upon the neighbors of element a_{ij}.

For any row i there are exactly n nodal constraints, one for each column. Thus there are exactly mn total nodal constraints per CF-OPSM instance. Considering that the number of inversions on a typical row of A is $O(n^2)$, using inequalities (2.54) in place of (2.50) greatly reduces the total number of constraints and corresponding size of the model, especially because a given CF-OPSM formulation has inversion restrictions for $O(m)$ rows. Our computational testing reveals that the CF-OPSM formulation using nodal constraints retains almost all of the tightness of the original CF-OPSM formulation (see Section 2.3.6.2 for further discussion).

2.3.5.5 Further Enhancements Next we present additional strategies aimed at improving the running time of Algorithm 3.

Turning Off Previous Rows. For current row h, observe that for all previous rows $i = 1, \ldots, h-1$, we can set $x_i = 0$. This is valid because previously solved row i is either: (i) not in the OPSM; or (ii) in the OPSM, but since we have already identified the optimal solution (OPSM) for row i, it is not necessary to locate that solution again. Thus, in either case, $x_i = 0$ is a valid inequality for all rows $i = 1, \ldots, h-1$.

Valid Lower Bounds. Another observation is that, for any row $h > 1$, a valid lower bound on the objective value for all future rows is $\bar{Z} = \left(\max_{k=1,\ldots,h-1} Z^k \right) + 1$. That is, in order to

locate a larger OPSM than the best found in previous rows, the current objective value Z^h must outperform $\bar{Z}$. Thus the inequality:

$$\sum_{i}^{m}\sum_{j}^{n} z_{ij} \geq \bar{Z} \tag{2.55}$$

is valid for all rows $h > 1$. If for current row h no feasible solution exists having objective value $Z^h \geq \bar{Z}$, we are free to move on to the next row. To allow for *multiple* optimal solutions in our code, simply removing the increase of one in the definition of $\bar{Z}$ will permit this possibility. Furthermore, CPLEX 11.0 [57] offers the solution pool feature, which has the capability to find all solutions within a certain user-defined tolerance of optimality, as well as to find diverse solutions to MIPs. Thus, this feature could also be used to find and store all optimal OPSMs.

LP-based Fathoming. Because linear programming relaxations generally solve much more quickly than their integer counterparts, after generating the necessary IP formulation for iteration h we choose to solve its LP relaxation by allowing $x_i \in [0,1]$, $y_j \in [0,1]$, and $z_{ij} \in [0,1]$. Solving this LP relaxation, denote the optimal objective value as $Z^*_{h_{LP}}$. If the obtained LP bound $Z^*_{h_{LP}}$ does not at least match $\bar{Z}$, do not proceed solving the corresponding 0–1 programming problem. As the integral optimal solution to this instance cannot yield a larger optimal objective value than $Z^*_{h_{LP}}$, i.e., $Z^*_h \leq Z^*_{h_{LP}} \leq \bar{Z}$, we are free to move on to the next row.

LP-based Presorting. For every row $h = 1, \ldots, m$, generate and solve the m LP relaxations of the respective IP formulations and record the corresponding optimal objective values. Re-sort the rows of the original data matrix A in the following manner. In the first row place the row corresponding to the largest optimal objective value. Order all remaining rows $2, \ldots, m$ into increasing order corresponding to their recorded optimal objective values. Knowing that the row with the highest LP relaxation optimal objective value gives an upper bound on the largest OPSM, placing this row first will presumably provide the best chance of finding a row contained in the largest actual OPSM, thereby achieving the largest possible initial solution $\bar{Z}$.

Re-sorting the remaining rows into increasing order can prove very beneficial when combined with other enhancements. In conjunction with valid inequality (2.55), this often forces the instances of IP formulations for rows $h > 1$ to become infeasible, as they likely cannot

find a feasible solution satisfying such a restrictive inequality. Thus, little time is typically spent on such rows. Also, when this step is combined with the fathoming process above, we are often able to fathom quickly, thereby providing additional savings on computation time. This approach proves very useful in our computational experiments.

Stopping Criteria. Setting $x_i = 0$ for all rows $i < h$ allows us to derive a valid upper bound on the largest potential objective function value for row h as $W = n \times (m - h + 1)$, corresponding to $z_{ij} = 1 \ \forall \ j, \ i = h, \ldots, m$. Thus, if the objective value of the current best solution $\bar{Z} \geq W$, we can safely terminate the algorithm; it is not possible for a larger submatrix to exist in the remaining rows.

2.3.5.6 Enhanced Iterative Algorithm

Algorithm 5 incorporates many of these improvements for solving the OPSM problem.

Algorithm 5. *[Enhanced Iterative Algorithm]*

1. *Perform LP-based presorting of rows of A utilizing either pairwise inequalities* (2.50) *or nodal constraints* (2.54) *in the respective LP formulations. Assign* $h := 1$.

2. *Permute columns of A such that the entries in row h occur in increasing order; call new matrix* $\hat{A}_h$.

3. *Generate formulation CF-OPSM for matrix* $\hat{A}_h$ *using either pairwise inequalities* (2.50) *or nodal constraints* (2.54).

3(a). *Add additional constraint* $x_h = 1$, *and* ***For all*** $k < h$, *add constraints* $x_k = 0$.

3(b). *Generate mn maximal decreasing subsequences via modified Algorithm 4 (see discussion in Section 2.3.5.3); add respective valid inequalities* (2.53).

3(c). ***If*** $h > 1$ *add valid lower bound* (2.55).

3(d). ***If*** $h > 1$ *solve LP relaxation of the respective IP formulation using either pairwise inequalities* (2.50) *or nodal constraints* (2.54). *Let* $Z^*_{h_{LP}}$ *be the obtained optimal objective function value.* ***Go To*** *6 if* $Z^*_{h_{LP}} \leq \bar{Z}$.

4. *Solve the obtained IP formulation. Let* Z^h *be the obtained optimal objective function value, and store off the optimal solution for row h.*

5. *Assign* $\bar{Z} := \max_{k=1,\ldots,h} Z^k$.

6. ***If*** $h < m$, *assign* $h := h + 1$. *Check* ***Stopping Criteria*** *and* ***Go To*** *2, if necessary.*

7. ***Return*** $Z^* := \bar{Z}$, *its corresponding optimal solution, and* ***Stop***.

2.3.6 Computational Experiments and Results

We use both synthetic and real biological data to verify the effectiveness of our proposed methods, with our findings from synthetic data testing aiding in our choice of the best configuration for testing real data. To perform the optimization, we used the callable library of CPLEX 11.0 [57] and coded our algorithms in C++.

2.3.6.1 Experiments with Synthetic Data Our testing environment consisted of a Windows XP machine equipped with a 2.4GHz Pentium 4 processor and 2GB of RAM.

Algorithmic Parameters. STOP [7], an automated tool to tune optimization software parameters, provided us with a suite of parameters that gave us marked performance improvements. Specifically, we adjusted the CPLEX default parameter settings for the MIP Emphasis parameter to *Feasibility over Optimality*, the Implied Bound, Clique, and Gomory Fractional Cuts parameters to *Aggressive*, the VariableSelect (Branching) parameter to *Strong Branching*, and the RINS Heuristic parameter to *Every Iteration*.

Moreover, we always implemented the **Stopping Criteria** from Section 2.3.5.5. Additionally, in Steps **1.** and **3(d).** of Algorithm 5, we use the CPLEX barrier algorithm, an interior-point method, rather than the simplex algorithm (which proved to be less efficient for our larger instances) to perform the LP-relaxation optimization. Because only the optimal objective values of these relaxations are necessary for our methods, and not the optimal solutions themselves, we also turn off the expensive crossover routines (BARCROSSALG), thereby recovering additional computational savings.

Synthetic Data Generation. In order to create our synthetic data, we coded a test instance generator according to the procedure outlined in [9]. This generator plants an order-preserving submatrix into an otherwise randomly generated data matrix. It first randomly chooses the indices for planted rows and columns. Subsequently, it randomly chooses an

ordering for planted columns. Finally, the generator randomly assigns ranks to the data matrix in a way that is consistent with the planted submatrix. The input is the number of rows m and columns n of A, the number of columns s in the planted OPSM, and the probability p of a given row being included in the submatrix to be planted. Using this information the test instance generator randomly embeds a submatrix according to the input parameters specified, while simultaneously identifying (for the user's benefit) the actual submatrix planted within A.

Table 2.5: Algorithmic variations used in computational testing for the OPSM problem

	Algorithmic Variations												
Formulation / Enhancement	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
General Formulation	•												
All Pairwise Ineqs. (2.50)		•	•	•	•	•	•					•	•
Nodal Ineqs. (2.54)								•	•	•	•		
Turning Off Previous Rows				•	•	•	•	•	•	•	•	•	•
Max Inversion Ineqs. (2.53)						•	•	•	•	•	•	•	•
Valid LB Ineq. (2.55)										•	•	•	•
LP-based Presorting			•		•		•		•		•		•

Two scenarios are proposed for testing synthetic data in order to determine the best algorithm configuration for further testing on real data.

Synthetic Data Test Scenario: Vary Algorithms. The first test scenario evaluates the performance of variations of our algorithm on some smaller test instances. Specifically, there are six levels of test sets, with data matrix sizes (m, n) ranging from $(20, 10)$ to $(50, 20)$; for each level, there are three test instances. The embedded OPSMs comprise approximately 25% of the overall data matrix size; that is, we set $p = 0.5$ and $s = 0.5n$, to give a mean embedded OPSM size of $p \times m \times s = 0.25mn$. Regarding algorithms, we first test the General Integer Programming Formulation (Section 2.3.4) and record the run times for CPLEX 11.0 [57] to locate an OPSM of optimal size on the generated test instances. We compare these results against variations of the iterative algorithm (Section 2.3.5); starting

with Algorithm 3 (Basic Iterative Algorithm), we sequentially augment this algorithm with valid inequalities and further enhancements, until reaching Algorithm 5 (Advanced Iterative Algorithm). Run times are recorded for the amount of time necessary to find an optimal OPSM for each algorithm and test instance combination, and as there are three test cases, where possible we provide the mean run time for each test level. The algorithmic variations are reported in Table 2.5; Algorithm **A** is the General Formulation of Section 2.3.4, while Algorithms **B** through **M** are variations of the Iterative Algorithm.

Synthetic Data Test Scenario: Vary OPSM Size. The second test scenario is to vary the size of the planted OPSM. For the two mean embedded OPSM sizes of $0.25mn$ and $0.2mn$ (i.e., 25% and 20% of the overall size of data matrix A), we create 15 levels of test sets, with each test set again having three test instances. Algorithm 5 (Advanced Iterative Algorithm, version **K**, which we later ascertain to be the best variation according to Table 2.6) is used to determine how changes in input data size affect our algorithm.

2.3.6.2 Synthetic Data: Results and Discussion

The results of the first test scenario are reported in Table 2.6, containing m, n, and the optimal OPSM size Z^*. All times are in seconds, with the fastest run times in bold. Where possible, every fourth line details the average of the previous three lines, and any average run time that was not within 5 times of the fastest run time is indicated by "–". The General Formulation, along with Algorithms **B**, **C**, **D**, and **E**, cannot compete with the more advanced iterative algorithms. Algorithms **F** and beyond are more competitive, coinciding with their inclusion of the facet-defining maximal inversion inequalities (2.53).

One general trend is that, holding all else the same, test instances of equal m and n having larger embedded OPSMs tend to solve more quickly than those with smaller OPSMs. This is evident, for example, in the second test instance of size $m = 30$ and $n = 15$; the optimal OPSM size of Z^* was 64, much less than the expected size of $0.25mn = 112.5$. Correspondingly, the run times for this instance are much larger than the first and third test instances for the same levels of (m, n).

Table 2.6: Comparison of algorithmic run times on synthetic test instances

m	**n**	Z^*	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
20	10	50	–	–	–	–	–	24.5	10	17.8	**7.9**	17.3	8.3	24.3	9.9
20	10	50	–	–	–	–	–	14.7	13.8	14.8	12.1	4.9	12	**4**	13.7
20	10	40	–	–	–	161.3	175.4	94.6	125.2	150.5	139.8	87.6	**42.5**	48.6	50.8
Average			–	–	–	87.4	–	44.6	49.7	61.1	53.3	36.6	**21**	25.6	24.8
30	10	70	–	–	–	–	–	–	–	–	–	21.6	23.5	**9.5**	20.2
30	10	85	–	–	–	–	–	–	8.9	–	**8.6**	–	8.9	–	8.6
30	10	50	–	–	–	–	–	–	–	–	–	169.1	207.1	**78.5**	136.8
Average			–	–	–	–	–	–	–	–	–	98.5	79.8	60.8	**55.2**
30	15	144	–	–	–	–	–	2.8	–	2.8	–	1.3	–	**1.2**	–
30	15	64	–	–	–	–	–	–	–	–	–	2,106.8	**466.7**	1,074	828.6
30	15	144	–	–	–	–	–	3	–	2.6	–	1.5	–	**1.4**	–
Average			–	–	–	–	–	–	–	–	–	703.2	**160.8**	358.9	281.7
40	15	160	–	–	–	–	–	8.5	11.8	6.9	11.9	3.7	11.5	**3.4**	12.3
40	15	168	–	–	–	–	–	9.3	12.4	8.8	13.1	5.1	13.4	**4.7**	13
40	15	168	–	–	–	–	–	–	**12.3**	–	12.4	–	12.8	–	12.8
Average			–	–	–	–	–	–	**12.2**	–	12.5	–	12.6	–	12.7
40	20	210	–	–	–	–	–	–	16.7	–	**14.7**	–	15.6	–	17.3
40	20	230	–	–	–	–	–	12.5	15.8	9.4	14.7	3.6	15.5	**3.3**	–
40	20	220	–	–	–	–	–	9.9	–	10.3	14.7	4	–	**3**	–
Average			–	–	–	–	–	–	16.2	–	**14.7**	–	15.6	–	17.2
50	20	210	–	–	–	–	–	162.5	146.6	181.8	155.9	99	156.8	**79.9**	147.6
50	20	200	–	–	–	–	–	–	140.2	–	234.9	–	233.7	–	**117**
50	20	170	–	–	–	–	–	–	–	–	–	–	**967.9**	–	1,648.9
Average			–	–	–	–	–	–	–	–	–	–	**452.8**	–	637.8

Furthermore, the results of Table 2.6 indicate that the fastest algorithm is one of Algorithms **K**, **L**, and **M**. Algorithm **L** has reasonable run times only for those test instances that happen to have their first row in a rather large OPSM. For such instances, because Algorithm **L** does not perform **LP-based Presorting** and does include the **Valid LB Inequality** (2.55), it has a favorable initial optimal objective value from the first row, uses the lower bounding inequality (2.55), but does not include the additional overhead of performing the **LP-based Presorting** step. In general, however, because there is no way to know whether the first row is included in a rather large OPSM, we prefer to use the **LP-based Presorting** step present in Algorithms **K** or **M**. This is because the additional computational time of solving m LP relaxations (via the efficient barrier algorithm) and sorting the rows accordingly almost always recovers that time by finding a first row with a large (sometimes optimal) OPSM.

Algorithms **K** and **M** seem roughly comparable, though we slightly prefer Algorithm **K**, because it has six of the fastest run times, while **M** reflects only two. More notable, however, is on difficult test instances where the value of Z^* is much less than the expected size of $0.25mn$ (e.g., instance 2 of $(30, 15)$ and instance 3 of $(50, 20)$), Algorithm **K** has the fastest run times. Indeed, this trend continues as the values of m and n grow, with the benefits of the smaller nodal inequality formulation in Algorithm **K** outweighing the slightly tighter formulation (but increasingly unwieldy due to greater constraint matrix expansion) given by the pairwise inversion inequalities in Algorithm **M**.

Furthermore, the nodal inequality formulation of Algorithm **K** does not adversely impact Step **1.** (Presorting), because every one of the m subproblems incurs a small gain in the LP relaxation optimal objective value. Thus the relative rankings of the optimal objective function values, needed for Step **1.**, remained more or less constant over all rows.

Regardless of whether an algorithm uses **LP-based Presorting**, repeatedly solving integer programs is the most time-consuming step. This is why Step **1.** (Presorting) of Algorithm 5 is in general advantageous, because after solving the first integer program, we use its optimal solution as a bound for future integer programs, serving to avoid the solution of additional integer programs (as detailed in Section 2.3.5.5, **Valid Lower Bounds** and **LP-based Fathoming**).

Table 2.7: Run times for Algorithm K to find synthetic OPSMs

Data		**Algorithm K**	
Rows	**Cols**	$0.25mn$	$0.2mn$
20	10	< 1	< 1
30	10	1	1
30	15	2	1
40	15	< 1	2
40	20	< 1	12
50	20	4	24
100	20	2	8
100	30	2	19
100	40	4	3
100	50	5	6
200	50	29	32
400	50	139	166
600	50	286	337
800	50	503	546
1,000	50	850	916

Table 2.7 displays the results of the second test scenario. It highlights the trend that Algorithm **K** has an easier time locating OPSMs of larger expected size. Also, we should mention that the General Formulation does not perform well in identifying OPSMs in even modest size input matrices; the maximum size it was able to handle in a reasonable amount of time was $(30, 10)$, in around two hours of computation.

2.3.6.3 Experiments with Real Data Our testing environment consisted of a Windows XP machine equipped with a Dual-Core Intel Xeon 3GHz processor and 3GB of RAM. In our experiments we considered two real data sets, which we briefly describe below.

Breast Cancer Data Set (BRCA). The BRCA breast tumor data set has $m = 3,226$ genes (rows) and $n = 22$ samples (columns). Of these samples, eight have brca1 mutations, another eight have brca2 mutations, and the remaining six represent sporadic breast tumors. This was the same data set utilized by [9]; for further details regarding this data set, we refer the reader to [49].

Human Gene Expression Index (HuGE). Our second data set was derived from the Human Gene Expression (HuGE) Index data set [55], the same data set used for checkerboard pattern detection in Section 2.2.7.1. We preprocessed the HuGE data set to remove all rows containing at least one incomplete or missing entry, and also retained samples from three of the main tissues (Brain, Liver, and Muscle), leaving $1,125$ genes and 23 columns. Other experiments with HuGE appear, for example, in [19].

Detecting Statistical Significance. An important contribution in [8, 9] is the derivation of statistically significant OPSM patterns. As they explain, for a set $T \subset \{1, \ldots, n\}$ of columns of size γ together with a linear ordering $\pi = (t_1, t_2, \ldots, t_\gamma)$, the probability that a random row supports a given model (T, π) is $(1/\gamma!)$. Further, because the rows are assumed as independent, the probability of having at least ρ rows supporting model (T, π) is the ρ-tail of the $(m, (1/\gamma!))$ binomial distribution. Because there are $n \cdot (n-1) \cdots (n-\gamma+1)$ ways to choose a complete model of size γ, they provide an upper bound on the probability of having a model of size γ with at least ρ rows as:

$$U(\gamma, \rho) = n \cdots (n - \gamma + 1) \sum_{i=\rho}^{m} \binom{m}{i} \left(\frac{1}{\gamma!}\right)^i \left(1 - \frac{1}{\gamma!}\right)^{(m-i)}. \tag{2.56}$$

In the BRCA breast cancer data set, it is reported in [8] that three statistically significant OPSMs were found, one with $\gamma = 4$ tissues (columns) and $\rho = 347$ genes (rows) with $U(4, 347) = 8.83 \cdot 10^{-51}$, another with 6 tissues and 42 genes with $U(6, 42) = 8.85 \cdot 10^{-19}$, and a third involving 8 tissues and 7 genes with $U(8, 7) = 0.0497$.

2.3.6.4 Finding OPSMs in Real Data Sets The BRCA data set is considerably larger than our synthetic data sets. It is not surprising, then, that initial testing on these data using Algorithm **K** indicated that while CPLEX locates rather good feasible solutions, it experiences difficulty in closing the integrality gap from above. We believe this large gap was due to poor linear relaxations, resulting from both the linearization of 0–1 variable products as well as incomplete information resulting from not using all maximal clique inequalities. In order to achieve faster convergence on real data sets, it was therefore necessary to make some modifications to our solution approach.

Reducing the Feasible Region and Strengthening the Formulation. In order to facilitate a more rapid closing of the integrality gap, consider temporarily restricting the number of rows in an OPSM to be no more than a constant ρ. Similarly, let the number of columns be no more than γ, so that:

$$\sum_{i}^{m} x_i \leq \rho, \tag{2.57}$$

$$\sum_{j}^{n} y_j \leq \gamma. \tag{2.58}$$

Then adding constraints (2.57) – (2.58) to any of the CF-OPSM variations, with well-chosen values of ρ and γ, will result in a formulation having a greatly reduced feasible region, one that CPLEX can much more easily handle. Furthermore, the optimal solution to this restricted problem forms a *valid* OPSM that is *feasible* for the original formulation without (2.57) – (2.58). Moreover, we can strengthen constraints (2.57) – (2.58) by appropriate multiplication of binary variables, as per the reformulation linearization technique (RLT) [92]. This yields the following additional valid inequalities:

$$\sum_{i}^{m} z_{ij} \leq \rho \cdot y_j \quad \forall\ j, \tag{2.59}$$

$$\sum_{j}^{n} z_{ij} \leq \gamma \cdot x_i \quad \forall\ i, \tag{2.60}$$

$$\gamma \sum_{i}^{m} x_i + \rho \sum_{j}^{n} y_j - \sum_{i}^{m} \sum_{j}^{n} z_{ij} \leq \gamma\rho, \tag{2.61}$$

$$\sum_{i}^{m}\sum_{j}^{n} z_{ij} \leq \gamma\rho, \tag{2.62}$$

where (2.59) and (2.60) are derived from (2.57) and (2.58) by multiplication of y_j and x_i, respectively, while (2.61) is derived by moving the right-hand sides of (2.57) and (2.58) to the left and multiplying.

The main idea in the solution approach of [8] is to first restrict the number of columns to a fixed value γ, and then search for OPSMs over the rows. With this in mind, by replacing (2.58), (2.60) and (2.61) with:

$$\sum_{j}^{n} y_j = \gamma, \tag{2.63}$$

$$\sum_{j}^{n} z_{ij} = \gamma \cdot x_i \quad \forall\ i, \tag{2.64}$$

$$\gamma\sum_{i}^{m} x_i + \rho\sum_{j}^{n} y_j - \sum_{i}^{m}\sum_{j}^{n} z_{ij} = \gamma\rho, \tag{2.65}$$

respectively, we are also able to search for feasible OPSMs with fixed value γ. This discussion then motivates an embedded algorithm that iteratively increases ρ and γ toward m and n.

Embedded Algorithm with Partial OPSMs. For a given row h, define strictly increasing sequence $\{\rho\}_1^\pi = \{\rho_1, \ldots, \rho_\pi\}$ and nondecreasing sequence $\{\gamma\}_1^\pi = \{\gamma_1, \ldots, \gamma_\pi\}$, and let $\rho = \rho_1$, $\gamma = \gamma_1$. The optimal solution from Algorithm **K** augmented with constraints (2.57) – (2.58) and respective valid inequalities from (2.59) – (2.62) is a *partial* OPSM, *feasible* to the overall OPSM problem for row h. Now assume that both constraints (2.57) and (2.58) are non-binding for this optimal solution. Then it can be shown that a larger OPSM cannot be obtained for row h by increasing ρ and γ. On the other hand, if these constraints are tight, then by increasing the value of ρ to ρ_2 and γ to γ_2, we obtain a problem with a strictly larger feasible region, and can likely improve the previous iteration's solution. Furthermore, the partial OPSM from the previous iteration is also feasible for the larger region, and we can use it as a *warm start* to provide a valid lower bound. Iterating in this manner is a valid approach that converges to the largest OPSM for row h. Because integer programming solvers

can typically handle smaller feasible regions more easily, this strategy provides significant computational improvements.

The stopping condition of this approach, which is outlined in Algorithm 6, is either 1) when both constraints (2.57) and (2.58) are not binding, or 2) when we have used the final values in our sequence (i.e., $\rho = \rho_\pi$ and $\gamma = \gamma_\pi$). Note that it is not necessary to run this embedded algorithm over all m rows; for example, we can designate it to be run for only the first r rows. Then, for $r = 0$, Algorithm 6 is essentially Algorithm 5.

Algorithm 6. *[Embedded Algorithm]*

1. *Perform LP-based presorting of rows of A utilizing CF-OPSM formulation with nodal constraints (2.54) in the respective LP formulations. Assign $h := 1$, and read in sequences $\{\rho\}$, $\{\gamma\}$, and embedded algorithm iteration limit r.*

2. *Assign embedded iteration counter $\ell := 1$, $\rho := \rho_\ell$, and $\gamma := \gamma_\ell$.*

3. *Permute columns of A such that the entries in row h occur in increasing order; call new matrix $\hat{A}_h$.*

4. *Generate formulation CF-OPSM for matrix $\hat{A}_h$ using nodal constraints (2.54).*

4(a). *Add additional constraint $x_h = 1$, and **For all** $k < h$, add constraints $x_k = 0$.*

4(b). *Generate maximal decreasing subsequences via modified FindLDS and FindLIS Algorithms (see Algorithm 4 in Section 2.3.5.3); add respective valid inequalities (2.53).*

4(c). ***If** $h > \max\{1, r\}$, add valid lower bound (2.55).*

4(d). ***If** $h > \max\{1, r\}$, solve LP relaxation of the IP formulation. Let $Z^*_{h_{LP}}$ be the obtained optimal objective function value. **If** $Z^*_{h_{LP}} \leq \bar{Z}$, **Go To** 9.*

4(e). ***If** $h \leq r$, add constraints (2.57) – (2.62) (or in the case of $\gamma_1 = \gamma_\pi$, instead add the constraints (2.57), (2.59), (2.62), and (2.63) – (2.65)) to formulation using current values of ρ and γ.*

5. ***If** $\ell > 1$ **And** $h \leq r$, read in initial feasible solution from warm start file.*

6. *Solve the obtained IP formulation. Let Z^h be the obtained optimal objective function value. **If** $h \leq r$, write optimal solution to a warm start file.*

7. ***If*** $h > r$, ***Go To*** *8.* ***Else*** $h \leq r$, *so* ***If*** $\ell = \pi$ ***Or*** *both constraints* (2.57) *and* (2.58) *are non-binding (or just constraint* (2.57) *should constraint* (2.58) *not be present), also* ***Go To*** *8.* ***Else****, assign* $\ell := \ell + 1$, $\rho := \rho_\ell$, *and* $\gamma := \gamma_\ell$, *and* ***Go To*** *4.*

8. *Assign* $\bar{Z} := \max_{k=1,\ldots,h} Z^k$.

9. ***If*** $h < m$, *assign* $h := h + 1$. *Check* ***Stopping Criteria****, and* ***If*** *it is necessary to continue, then* ***If*** $h \leq r$, ***Go To*** *2. Otherwise* ***Go To*** *3.*

10. ***Return*** $Z^* := \bar{Z}$ *and* ***Stop****.*

Depending on the sequences $\{\rho\}$, $\{\gamma\}$ and whether constraints (2.58) or (2.63) are chosen for Algorithm 6, upon termination we can guarantee that discovered solutions are either:

(a) optimal, if $\rho_\pi = m$, $\gamma_\pi = n$ and formulation with (2.58) is used in Step **4(d).**, or

(b) optimal for a particular value of γ, if formulation with (2.63) is used in Step **4(d).**, or

(c) feasible and of high quality, corresponding to the values in the sequence $\{\gamma\}$.

We should also make note of the solver tuning we performed. Though there are only n column variables y_j in the CF-OPSM formulation ($n << m$), they are quite powerful, appearing in every composite product z_{ij}. Giving priority to (upward) branching in CPLEX on this small subset of variables forces many y_j variables to 1, and doing so in conjunction with (2.58) or (2.63) serves to quickly eliminate portions of the branch-and-bound tree due to the maximal inversion inequalities (2.53) causing poor linear program relaxation values. Moreover, turning on CPLEX's barrier algorithm on both the root and child nodes of the branch-and-bound tree for a given row h helped to solve subproblems more quickly than the simplex algorithm. Lastly, CPLEX's tuning tool recommended that we run the RINS heuristic every *50* iterations, set the CUTPASS parameter to *1*, and set the VARIABLE-SELECT (BRANCHING) parameter to *pseudo-reduced costs*, which we implemented.

In summary, Algorithm 6 and the associated solver tunings enables us to quickly find strong feasible solutions. Moreover, any feasible solutions the algorithm outputs in Step **6.** are available for immediate data analysis purposes; it is not necessary to wait for convergence to the final solution. Lastly, for repeated runs on the same data set, Step **1.** needs only be completed once; the presorted order of the rows can then be written to a file for subsequent runs. Indeed, we use this tactic to reduce our run times in the following discussion.

2.3.6.5 BRCA Data: Results and Discussion It was reported in [9] that statistically significantly patterns were located in the BRCA data set [49] of size $(\gamma, \rho) = (4, 347)$, $(\gamma, \rho) = (6, 42)$ and $(\gamma, \rho) = (8, 7)$. We used Algorithm 6 with the initial values of γ (fixed) and ρ set to match their largest dimensions in order to compare the performance of our algorithm.

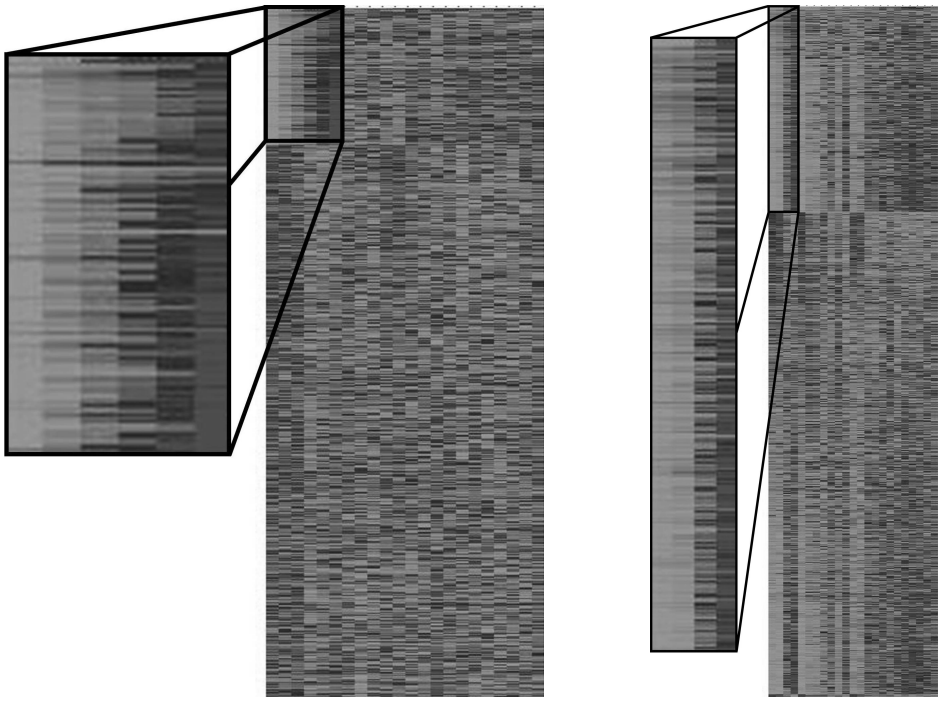
Knowing that our approach will eventually converge to the optimal solution, we present some feasible solutions from our results in Table 2.8, along with their run times. The second column displays initial values of ρ that match the largest reported in [9], while the third and fourth columns display sequential results from allowing Algorithm 6 to iterate. For every case, we were able to locate OPSMs of larger size than were reported in [9], in reasonable amounts of time. It is worth noting here that the $(8, 14)$ OPSM we identified has $U(8, 14) = 5.88 \cdot 10^{-17}$, improving upon the $U(8, 7) = 0.0497$ value of [9]. Figure 2.7a displays an exemplary OPSM found using BRCA. Also, because it was performed only once (taking 623 minutes), we omit the run time for Step **1.** (LP-based presorting).

Table 2.8: BRCA OPSMs found with Algorithm 6

Cols γ	**Rows** ρ **(Time**† **in min)**		
4	**347** (220)	**520** (586)	**798** (1,974)
6	**42** (71)	**63** (166)	**127** (2,121)
8	**7** (17)	**10** (435)	**14** (2,370)

2.3.6.6 HuGE Data: Results and Discussion As in the previous experiment, we used Algorithm 6 with a sequence of fixed values for γ. Our results on this data set are displayed in Table 2.9. Included in the display are the statistical significance levels for the embedded OPSMs in HuGE. Figure 2.7b displays an exemplary OPSM found using HuGE. Again, as it was performed only once (taking 64 minutes), the run time for Step **1.** (LP-based presorting) has been omitted from Table 2.9.

We feel these results represent favorable run times for finding large OPSMs in HuGE. The statistical significance of these solutions indicate that they are of high quality. Only for fixed $\gamma = 5$ did it take more than one day to locate such large OPSMs; all other solutions were found in well under 12 hours.



(a) $(6, 127)$ in BRCA (truncated) (b) $(4, 335)$ in HuGE

Figure 2.7: Heatmaps of OPSMs in real data using Algorithm 6

Table 2.9: OPSMs in HuGE data using Algorithm 6; statistical significance in final column

Cols (γ)	**Rows** (ρ)	**Time**† **(min)**	$U(\gamma, \rho)$
3	569	335	$2.14 \cdot 10^{-146}$
4	335	489	$2.23 \cdot 10^{-176}$
5	180	1,909	$1.84 \cdot 10^{-158}$
6	95	437	$7.43 \cdot 10^{-125}$
7	49	524	$6.98 \cdot 10^{-87}$
8	22	449	$8.87 \cdot 10^{-46}$
9	11	311	$1.79 \cdot 10^{-24}$

2.4 CONCLUDING REMARKS

To the best of our knowledge, we provide the first mathematical programming formulations for two specific biclustering criteria, and moreover they are the first approaches that provide a guarantee of locating *optimal*-sized patterns.

2.4.1 Checkerboard Pattern

We consider the case of locating checkerboard patterns using unsupervised learning, extending previous work on supervised biclustering that establishes *consistent biclustering* that is justified by the conic separation property [19]. We discuss the computational complexity of consistent biclustering and prove that the problem of finding the largest conditionally biclustering admitting submatrix is NP-hard. We formulate and present a fractional 0–1 program for unsupervised biclustering under the biclustering consistency conditions defined in [19]. Because the fractional 0–1 program contains nonlinear constraints, we present an equivalent linear mixed 0–1 programming reformulation that can be handled using mixed integer programming solvers.

2.4.2 OPSM Pattern

We address exact solution methodologies for the order-preserving submatrix (OPSM) problem. We discuss computational complexity issues related to finding fixed patterns in matrices, and propose two exact approaches to solve the OPSM problem to optimality. The first is a general nonlinear 0–1 programming formulation that can be linearized in straightforward fashion, while the second is an iterative algorithm that makes use of a smaller nonlinear 0–1 program (again easily linearized) of a restricted version of the initial problem. We discuss various algorithmic enhancements for the latter approach to improve solution times, enabling us to solve the OPSM problem to optimality for synthetic instances with up to 1,000 rows and 50 columns in a reasonable amount of time. We also discuss some additional enhancements to aid in the solution approach for real biological data sets that enable us to identify OPSMs larger than those reported in [8, 9].

3.0 OPTIMIZATION IN COMPUTATIONAL BIOLOGY

We next discuss using optimization in the context of computational biology to gain greater insight into the behavior of protein/DNA interactions. We introduce an optimization model that predicts binding free energy levels ($\Delta\Delta G_{Bind}$) that most closely match experimental energy observations ($\Delta\Delta G_{Exp}$) at distinct protein/DNA binding sites. We also discuss an approach to identify a best minimal set of explanatory interaction code parameters (e.g., hydrogen bonds, desolvation penalties, water modulation factor) that quantify these energies. This technique is designed to protect against both overfitting as well as underfitting the model. Our approach enables the possibility of efficiently designing highly specific zinc finger protein transcription factors, allowing for a better understanding of how they regulate gene expression. Such technology can be beneficial to gene therapy, for instance in the design of zinc fingers that can target HIV and other diseases.

3.1 ACKNOWLEDGMENT

Much of this chapter's content originally appeared in an open access article, and is reproduced with kind permission from Oxford University Press: N.A. Temiz, A. Trapp, O.A. Prokopyev, and C.J. Camacho, "Optimization of Minimum Set of Protein/DNA Interactions: A Quasi-Exact Solution with Minimum Over-fitting," *Bioinformatics*, 26 (3), pp. 319–325, 2010.

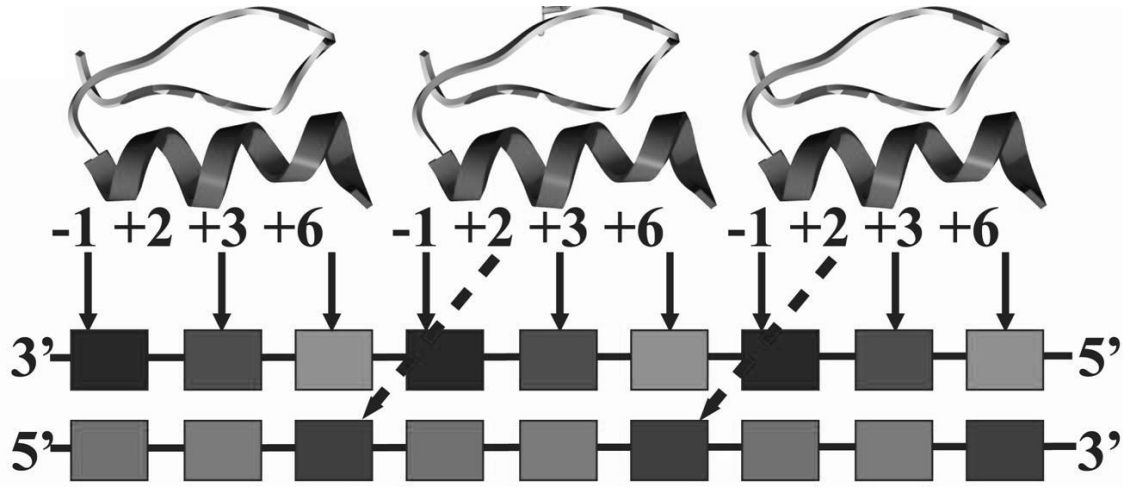


Figure 3.1: Typical interaction network of an EGR-like ZF

3.2 INTRODUCTION

Zinc finger (ZF) protein transcription factors bind to specific DNA targets and enable the transfer of genetic information from DNA to mRNA [97]. Determining their cognate DNA binding sites is a particular challenge that contributes to the general lack of knowledge of how protein transcription factors regulate gene expression; there are few reliable experimental techniques that can map the specificity of ZF/DNA interactions to other targets [21]. Computational techniques that aid in decoding these interactions can shed light on gene behavior and thereby contribute to restoring cell functionality.

It is possible to design new zinc fingers that bind to different DNA targets through mutations on the protein. These mutant ZF/DNA complexes may lead to a better understanding of the binding interactions, and in principle should select the lowest possible binding free energy conformation. Using high-quality experimental data from seven mutant complexes of ZF-I [65] and three mutants of ZF-III [5], Temiz and Camacho [98] manually constructed an interaction code utilizing potential (i.e., low binding free energy) structural interaction networks of mutant ZF/DNA complexes based upon measures of distance rather than (imperfect) scoring functions. Their work stood on the premise that changes in the affinity

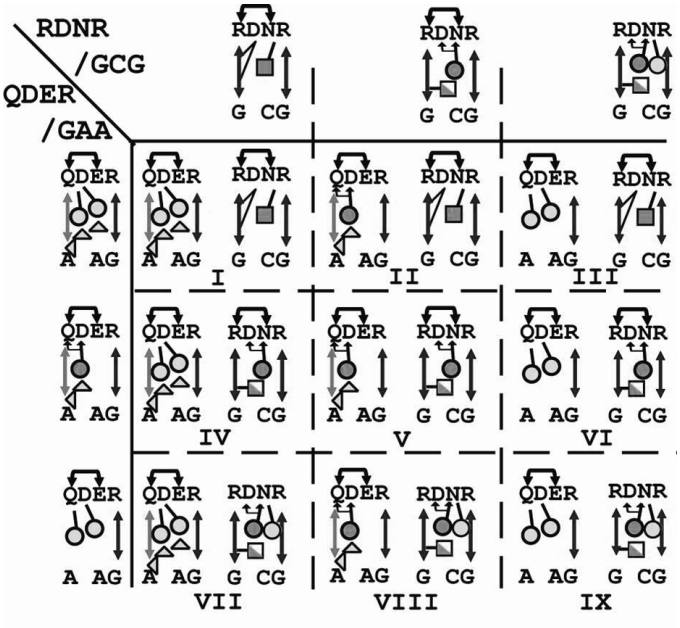


Figure 3.2: Nine ways that potential submodels can minimize binding free energy

of a complex due to mutations are uniquely determined by changes in the contact energies and solvation factors between the structures. For a given complex, an interaction network represents one possible way that the ZF protein can bind to the DNA through an expression that quantifies the associated binding free energy. Figure 3.1 (from [38]) displays an exemplary interaction network for a mutant complex, characterizing one potential option for the ZF/DNA binding.

Hereafter, we refer to structural interaction networks as *submodels*; they express a mutant complex's binding free energy as a (non)linear combination of a set of interaction code parameters (variables) that may include hydrogen bonds (H-bonds) and desolvation penalties, as well as a unique water modulation factor. This same set of parameters is used to construct every submodel of every complex. Figure 3.2 (from [99]) illustrates two mutant complexes (top and left), each having three submodels. It demonstrates the $3^2 = 9$ possibilities to minimize the binding free energy, which are determined by the choices of parameter levels in the interaction code. For additional information on these experimental techniques as well as graphical interpretations, we refer the interested reader to [98, 99].

3.3 MATHEMATICAL MODELING OF PROTEIN/DNA INTERACTIONS

We can model the problem of decoding protein/DNA interactions as a nonlinear mixed 0-1 program that seeks to minimize the distance between experimental energy observations and energy expressions that are quantified by combinations of variable parameter levels, while simultaneously satisfying physical constraints.

3.3.1 Nonlinear Mixed 0-1 Formulation

Let the set of ZF/DNA complexes be denoted as $\mathcal{I}$ and indexed by i, $i \in \{1, \ldots, |\mathcal{I}|\}$. Let $\mathcal{K}$ be the set of interaction code parameters, indexed by $k \in \{1, \ldots, |\mathcal{K}|\}$. Individual interaction code parameters e_k represent H-bonds and desolvation penalties, and are modeled using nonnegative continuous variables e_k with upper bounds M_k^e (typically a small positive number). The unique water modulation factor λ is nonnegative, continuous and strictly less than 1. Associate with each complex i a set $\mathcal{S}_i$ of submodels; each submodel $j \in \mathcal{S}_i$ is represented by the energy expression $\sum_k^{\mathcal{K}} w_{ij}^k(\lambda) e_k$, which in turn is a (non)linear combination of interaction code parameters. Also, w_{ij}^k is either a known coefficient, or a known function of λ, and if a particular parameter e_k does not appear in submodel $j \in \mathcal{S}_i$, we simply set $w_{ij}^k = 0$. For complex i and submodel $j \in \mathcal{S}_i$, binary variable $x_{ij} = 1$ when submodel j has been assigned to complex i, and $x_{ij} = 0$ otherwise. Finally, input data E_i are the energies of complex $i \in \mathcal{I}$ obtained experimentally; these are the values that the energy expressions of chosen submodels should most closely match through predicted parameter levels.

$$\min_{\lambda, x, e} \sum_{i \in \mathcal{I}} \left| E_i - \sum_{j \in \mathcal{S}_i} \left\{ x_{ij} \cdot \sum_k^{\mathcal{K}} w_{ij}^k(\lambda) e_k \right\} \right| \tag{3.1}$$

subject to

$$\sum_{j \in \mathcal{S}_i} x_{ij} = 1 \quad \forall\, i, \tag{3.2}$$

$$x_{ij} \cdot \sum_k^{\mathcal{K}} w_{ij}^k(\lambda) e_k \leq x_{ij} \cdot \sum_k^{\mathcal{K}} w_{ih}^k(\lambda) e_k \quad \forall\, i,\ \forall\, j \in \mathcal{S}_i,\ \forall\, h \in \mathcal{S}_i, h \neq j, \tag{3.3}$$

$$0 \leq \lambda < 1, \quad 0 \leq e_k \leq M_k^e \ \ \forall\, k, \quad x_{ij} \in \{0, 1\} \ \ \forall\, i,\ \forall\, j \in \mathcal{S}_i. \tag{3.4}$$

Expressions (3.1)– (3.4) is a nonlinear mixed 0–1 program. Objective (3.1) minimizes the cumulative difference between selected submodel energies and experimental data, while constraint set (3.2) ensures that exactly one submodel j is selected for every complex i. Constraint set (3.3) ensures that if submodel j is selected for complex i, then it must have the minimum energy among all other submodels $j \in \mathcal{S}_i$. Variable restrictions are given in (3.4).

3.3.2 Equivalent Nonlinear Mixed 0–1 Reformulation

While succinct, formulation (3.1) – (3.4) has nonlinearities in (3.1) and (3.3). The following equivalent reformulation removes most by introducing new variables and constraint sets.

$$\min_{\lambda,x,e,t,y} \sum_{i\in\mathcal{I}} t_i \tag{3.5}$$

subject to

$$t_i \geq E_i - \sum_{j\in\mathcal{S}_i} y_{ij} \quad \forall\, i, \tag{3.6}$$

$$t_i \geq -E_i + \sum_{j\in\mathcal{S}_i} y_{ij} \quad \forall\, i, \tag{3.7}$$

$$\sum_{j\in\mathcal{S}_i} x_{ij} = 1 \quad \forall\, i, \tag{3.8}$$

$$0 \leq y_{ij} \leq M^y_{ij} x_{ij} \quad \forall\, i,\ \forall\, j \in \mathcal{S}_i, \tag{3.9}$$

$$y_{ij} \leq \sum_{k}^{\mathcal{K}} w^k_{ij}(\lambda) e_k \quad \forall\, i,\ \forall\, j\ \in \mathcal{S}_i, \tag{3.10}$$

$$y_{ij} \geq \sum_{k}^{\mathcal{K}} w^k_{ij}(\lambda) e_k - M^y_{ij}(1 - x_{ij}) \quad \forall\, i,\ \forall\, j \in \mathcal{S}_i, \tag{3.11}$$

$$\sum_{k}^{\mathcal{K}} w^k_{ij}(\lambda) e_k \leq \sum_{k}^{\mathcal{K}} w^k_{ih}(\lambda) e_k + M^y_{ij}(1 - x_{ij}) \quad \forall\, i,\ \forall\, j \in \mathcal{S}_i,\ \forall\, h \in \mathcal{S}_i, h \neq j, \tag{3.12}$$

$$0 \leq \lambda < 1, \quad 0 \leq e_k \leq M^e_k \ \ \forall\, k, \quad x_{ij} \in \{0,1\} \ \ \forall\, i,\ \forall\, j \in \mathcal{S}_i. \tag{3.13}$$

Formulation (3.5) – (3.13) is a mixed 0–1 program that is nonlinear only in λ, that is, it becomes linear for a fixed λ. The newly introduced variables y_{ij} represent the binary product nonlinearities present in $x_{ij} \sum_k^{\mathcal{K}} w_{ij}^k(\lambda) e_k$. Constraint sets (3.9) – (3.11) implicitly enforce that y_{ij} equals $x_{ij} \sum_k^{\mathcal{K}} w_{ij}^k(\lambda) e_k$ if $x_{ij} = 1$, and $y_{ij} = 0$ if $x_{ij} = 0$. Constants M_{ij}^y are large enough to upper bound y_{ij}. Continuous variables t_i eliminate the absolute values in objective (3.1) by leveraging the minimization direction while simultaneously upper bounding both the positive and negative magnitudes in constraint sets (3.6) – (3.7). Constraint (3.12) equivalently represents (3.3) in a linear fashion, maintaining that selected submodel j has the minimum energy among all other submodel energies.

3.3.3 Final Linear Mixed 0–1 Reformulation

Formulation (3.5) – (3.13) eliminates most of the nonlinearities in (3.1) – (3.4), including the absolute values of objective (3.1) as well as the binary product nonlinearities in (3.1) and (3.3). However, there remains the presence of nonlinearities in some of the submodels $\sum_k^{\mathcal{K}} w_{ij}^k(\lambda) e_k$. These energy expressions are linear combinations of coefficients $w_{ij}^k(\lambda)$ and continuous variables e_k, where coefficients $w_{ij}^k(\lambda)$ represent either a constant, or else the product of a constant and a term involving λ. Whereas the latter might cause difficulties due to the product of continuous variables λ and e_k, in our particular case these expressions have a special structure that we can exploit. By representing continuous and bounded parameter λ using its binary representation, we are able to introduce additional variables and constraint sets that completely transform formulation (3.1) – (3.4) into a mixed integer program without nonlinearities.

Assuming we desire to represent λ within an accuracy of $\epsilon = 10^{-\kappa}$, where κ is some positive integer, then with $Z = \left\lceil \frac{log10}{log2} \kappa \right\rceil$ binary variables z we can write $\lambda \approx \sum_{h=1}^{Z} 2^{-h} z_h$. In our particular case the products involving λ and e_k appearing in some submodels $\sum_k^{\mathcal{K}} w_{ij}^k(\lambda) e_k$ show up in two distinct forms:

i) $(1 - \lambda) e_k$, and

ii) $e_k / (1 - \lambda)$.

To reformulate i) using the binary representation of λ, we introduce new continuous variable b_k as:

$$b_k = e_k - \lambda e_k = e_k - \sum_{h=1}^{Z} 2^{-h} z_h e_k. \tag{3.14}$$

Then b_k can be used to represent any product $(1-\lambda)e_k \approx e_k - \sum_{h=1}^{Z} 2^{-h} z_h e_k$, where each $2^{-h} z_h e_k$ term contains a product of binary variable z_h and continuous variable e_k. Define new continuous variable $s_{hk} = z_h e_k,\ h = 1, \ldots, Z$. According to Proposition 1 of Section 2.2.5.3, the following four constraint sets can impose this equality:

$$0 \leq s_{hk} \leq e_k \ \forall\ h,\ k, \quad \text{and} \quad e_k - M_k^e(1 - z_h) \leq s_{hk} \leq M_k^e z_h \ \forall\ h,\ k. \tag{3.15}$$

Constraint sets (3.15) implicitly enforce the $s_{hk} = z_h e_k$ relationship that allows form i) to be rewritten as $b_k = e_k - \sum_{h=1}^{Z} 2^{-h} s_{hk}$. Thereafter any occurrence of form i) can be replaced with variable b_k.

To reformulate ii), we introduce new continuous variable u_k:

$$u_k = e_k/(1 - \lambda), \tag{3.16}$$

equivalently writing $e_k = u_k - \lambda u_k$. The term λu_k can be linearized in a similar manner as previously described with b_k. Let $v_{hk} = z_h u_k,\ h = 1, \ldots, Z$, and let M_k^u be a constant upper bound for u_k. The four constraint sets are:

$$0 \leq v_{hk} \leq u_k \ \forall\ h,\ k, \quad \text{and} \quad u_k - M_k^u(1 - z_h) \leq v_{hk} \leq M_k^u z_h \ \forall\ h,\ k. \tag{3.17}$$

We can subsequently replace any occurrence in the form of expression ii) with u_k by enforcing $e_k = u_k - \sum_{h=1}^{Z} 2^{-h} v_{hk}$ on the v_{hk} variables together with constraint sets (3.17). The result of this discretization-linearization procedure is a linear mixed 0–1 programming problem that can be tackled using standard MIP solvers such as CPLEX.

3.3.4 Mitigating Overfitting and Underfitting

One of the inherent benefits of our model is its extensibility, of which we take advantage to protect against statistical overfitting. Specifically, our initial interaction code may contain too many parameters due to the relative uncertainty of the ZF/DNA interactions. We next discuss an algorithmic approach that iteratively eliminates the least descriptive element among a set of potential parameter relationships by adding the corresponding constraint to the formulation. While the phenomenon of overfitting is in general nearly impossible to completely eliminate, our technique ensures we can maintain a satisfactory final assignment by stopping whenever the quality of the final solution falls below an adjustable threshold criteria, thereby providing an additional safeguard against underfitting the model. We next present our sequential routine designed to reduce superfluous parameters.

Algorithm 7. *[Reduce Parameters]*

1. *Choose threshold value α and set of potential parameter relations $\mathcal{R}$.*
2. *Generate linearized reformulation, optimize, and compute R^2.*
3. ***If** $R^2 \leq \alpha$ **Or** $\mathcal{R} = \emptyset$, **Go To** 7.*
4. *Choose relation $r \in \mathcal{R}$ having the closest relation (measured by smallest magnitude absolute value difference), and set $\mathcal{R} \leftarrow \mathcal{R} \backslash \{r\}$.*
5. *Add relation r to linearized reformulation and re-optimize.*
6. *Compute R^2, and **Go To** 3.*
7. ***Stop**.*

Algorithm 7 resembles a standard regression technique called *iterative backward elimination*. As long as $R^2 > \alpha$, it continues to optimize, compute resultant R^2 values, add the relation causing the least conflict, and re-optimize. Algorithm 7 requires a value α that indicates an acceptable threshold for correlation coefficient R^2, as well as a set $\mathcal{R}$ of possible relations (for example $e_{10} = 0.5e_4$, or $e_8 = 1.081e_1$); α and $\mathcal{R}$ are predetermined according to specific domain knowledge. In our experiments, $\mathcal{R}$ was proposed by our biomedical collaborators Dr. Camacho and Dr. Temiz (see [97] for more details).

3.4 COMPUTATIONAL EXPERIMENTS AND RESULTS

3.4.1 Test Data and Environment

Our initial test set ZF-I [65] consisted of $|\mathcal{I}| = 35$ mutant complexes, each having between two and four possible submodels [98]. There were twelve parameters in our original interaction code, including eleven e_k parameters representing fundamental interactions (six H-Bonds, five desolvation penalties) along with a single (implicit) water modulation factor λ. We set $\kappa = 4$ to ensure that our binary representation of λ has an accuracy of at least $\epsilon < 10^{-4}$, and after all reformulations were completed there were 786 variables (625 continuous, 161 binary) and 2,510 constraints in the resultant model. Our testing environment consisted of Windows XP, a Dual-Core Intel Xeon 3GHz processor, 3GB of RAM, and CPLEX 11.0 [57] with default settings to optimize the reformulated mixed 0-1 programs.

Table 3.1: Parameter reductions

# Parameters	R^2
12	0.99857
11	0.99855
7	0.9983

3.4.2 Test Results

The majority of the subproblems in Steps **2.** and **5.** of Algorithm 7 solved to optimality in five to ten minutes; the longest run was under thirty minutes. The default settings of CPLEX 11.0 were all that were necessary for the problems to solve to optimality in such short run times.

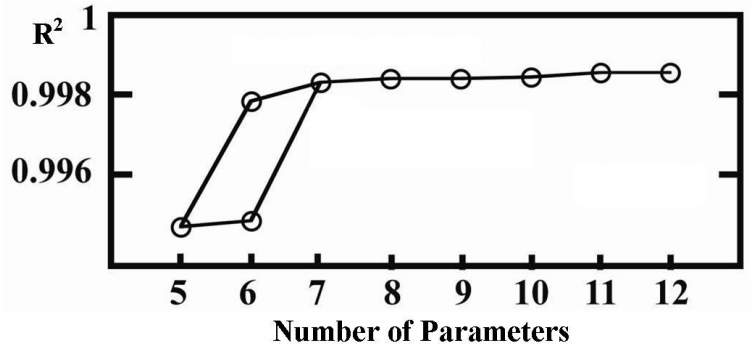


Figure 3.3: Parameter reduction effects on ZF-I

In general, we observed that the more relations from $\mathcal{R}$ that had been added in Step **5.** of Algorithm 7, the less time it took to solve the corresponding subproblem to optimality.

Beginning with the twelve original parameters of our original interaction code as well as an acceptable R^2 level α, we ran Algorithm 7 until the R^2 value of the current iteration decreased below the predetermined value of α. For our data, the R^2 value remained relatively unchanged after enforcing the first five parameter relations. After this point, however, a sharp drop-off was observed in the R^2 value upon enforcing any additional relations. Thus seven parameters remained, including three H-bonds, three desolvation penalties, and the water modulation factor λ.

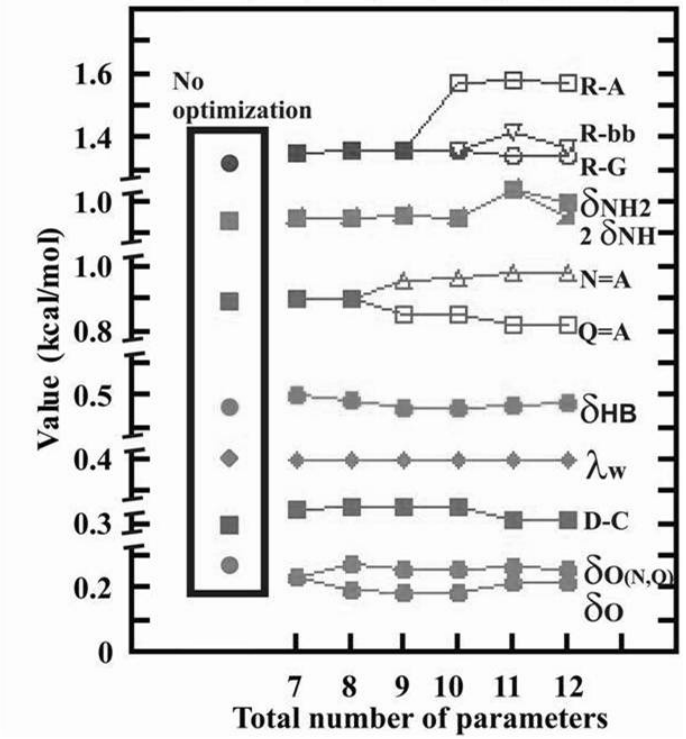


Figure 3.4: Single parameter reduction

After these initial marginal decreases in R^2 values, further reductions appeared to be detrimental to the mapping of the binding data. Table 3.1 details the parameter reductions together with the corresponding R^2 values. Graphically, Figure 3.3 (from [99]) depicts this phenomenon collectively with respect to R^2, while Figure 3.4 (from [99]) demonstrates the convergence of individual parameters using the output obtained from Algorithm 7. The visual discrepancy that appears after the reduction from seven to six parameters is due to the subsequent choice of parameter reduction [99].

3.4.3 Validating and Reassessing Submodels with Optimization Results

We validated the optimization results from the ZF-I data set [65] by measuring their performance on related ZF-II and ZF-III data sets having 23 and 31 mutant complexes, respectively [5, 88].

Figure 3.5 (from [99]) illustrates the performance of ZF-I's optimal parameter levels through their simple evaluation in the submodels of ZF-II and ZF-III. It depicts the levels obtained from each iteration of Algorithm 7 on ZF-I, from twelve parameters down to seven. This led to rather high levels of R^2, as depicted in Figure 3.6 (from [99]), where the R^2 of ZF-II is a robust 0.97, while for ZF-III it is still a very respectable 0.93. While not as strong as the correlation values for ZF-I (likely because our submodel designs are based upon experimental data from ZF-I), they are particularly noteworthy because no optimization was performed on either the ZF-II or ZF-III data sets. We attribute these relatively high R^2 values to the intrinsic significance of these seven parameters in characterizing zinc finger protein/DNA bindings on related, yet distinct data sets.

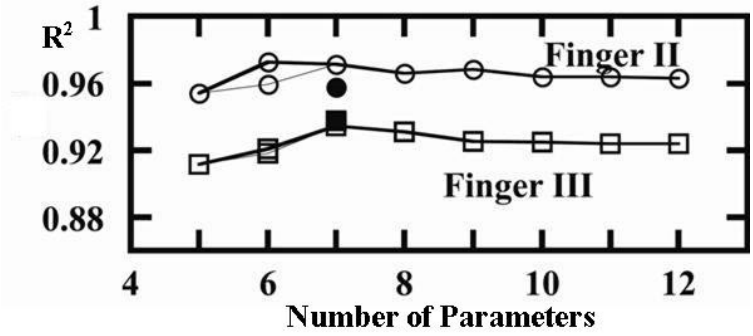


Figure 3.5: ZF-I's optimal parameters on ZF-II, ZF-III

We also systematically eliminated all seven ZF-I mutants that were used to construct the original code in [98]. The subsequent results of these optimization runs yielded an almost equivalent interaction code, serving as additional evidence that our optimization approach is capturing the underlying molecular interactions. Moreover, our observations also helped lead to two structural updates for the submodels corresponding to the ZF-II and ZF-III data sets, an added benefit that has improved the accuracy of the current experimental models [97, 99].

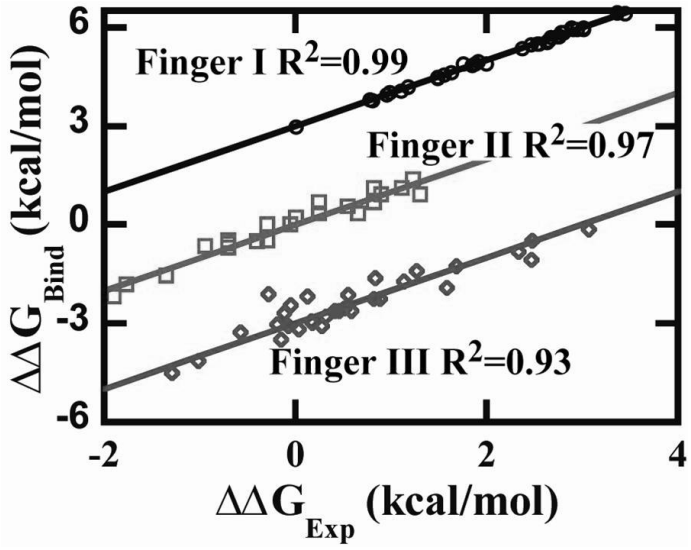


Figure 3.6: Predicted vs. experimental free energy

3.5 CONCLUDING REMARKS

Our objective in decoding the interactions between zinc finger protein transcription factors and DNA was to predict the structure and stability of protein/DNA complexes using a mathematical programming-based approach. We developed a nonlinear mixed 0–1 program and subsequently introduced several linearizations and variable substitutions that transform the problem into an equivalent (albeit somewhat larger in size) mixed 0–1 program, able to be solved with MIP solvers such as CPLEX. The resulting formulation exhibited relatively short run times to solve to optimality, yielding optimal parameter levels that improved upon existing (and labor-intensive) methods [98]. Additionally, through Algorithm 7 we addressed the issues of overfitting and underfitting. In so doing, we were able to reduce the number of fundamental interaction parameters from twelve to seven, while retaining a high R^2 value. Our findings indicate a very strong fit to actual experimental data on both optimized data set ZF-I, as well as non-optimized data sets ZF-II and ZF-III. Additionally, they helped lead to structural updates for two submodels by highlighting potential context-dependent effects in submodel definitions [97].

In the future, our framework can serve as a foundation to build more elaborate models, as it is readily scalable to larger data sets, and it can be easily extended by adding additional constraints and variables (for example, to forbid certain bindings, or to introduce a new interaction parameter into the optimization model). Additionally, our models could also be incorporated into a larger database and server system, enabling open access by researchers at large. As more complete information is received, the optimization model can then be automatically adjusted and re-optimized without manual intervention, thereby keeping the database up-to-date.

4.0 OPTIMIZATION IN SUSTAINABLE ENERGY DEVICES

The final application of optimization in this thesis is in the context of engineering design. In particular, we use optimization to enhance the design of a thermoacoustic engine.

4.1 ACKNOWLEDGMENT

Most of the content in this chapter is based on the following manuscript: A.C. Trapp[†], F. Zink[‡], O.A. Prokopyev[†], L. Schaefer[‡]. "Thermoacoustic Heat Engine Modeling and Optimization." Technical report, Univ. of Pittsburgh, Departments of Industrial Engineering[†] and Mechanical Engineering[‡], 2011.

4.2 INTRODUCTION

Thermoacoustic devices rely upon sound waves rather than mechanical pistons to drive the thermodynamic process. Thermoacoustic engines (TAEs) use hot heat input to generate intense sound; their main use is to drive thermoacoustic refrigerators (TARs), which cool by transforming intense sound from their surroundings [59, 82, 94, 104]. While TARs are a reality today, they are not yet competitive with current refrigeration technology in terms of energy or cost efficiency, even though they compare favorably in that they contain no toxic chemicals and have no moving parts.

A basic standing wave thermoacoustic engine consists of a resonance tube that is closed on one end and open on the other. The main component is a porous regenerative unit called

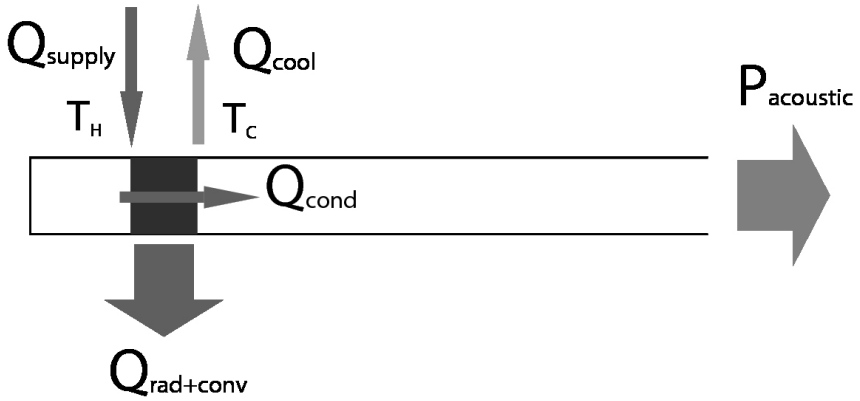


Figure 4.1: Inputs and outputs of thermoacoustic engine

the *stack*, which sits inside the resonance tube between two heat exchangers. One heat exchanger is used to supply heat at high temperature (on the order of several hundred °C), while the other withdraws heat from the system at ambient temperature. Locating the stack near the closed end of the resonance tube causes the interior gas to experience large pressure oscillations yet relatively small displacement. Amplification of pressure disturbances in the working gas then occur due to the temperature gradient across the regenerative unit, which, once a steady state has been attained, generates loud acoustic sound. The regenerative unit is responsible for both creating the sound and cooling, as well as the viscous losses and heat flows that inhibit thermoacoustic energy conversion. Figure 4.1 (from [114]) illustrates the features of a thermoacoustic engine, including several heat flows Q, acoustic power P, as well as the stack which appears as the dark square near the closed end of the resonance tube [114].

From the literature, it appears that thermoacoustic device designers tend to prefer parametric studies, where a single parameter is varied while holding all others constant, over optimization (see Section 1.3). Such parametric studies are useful, but unfortunately are unable to capture the nonlinear interactions inherent in multiple variable models. Addi-

tionally, the optimization approaches typically used in such studies guarantee only locally optimal solutions. In contrast, the model we develop allows for the simultaneous varying of multiple parameters, and we also identify globally optimal values for these variables. We also conduct multiobjective optimization over several contrasting objectives representing both acoustic and thermal properties, enabling us to generate the efficient frontier of Pareto optimal points from the optimal solutions to such contrasting objectives.

The remainder of this chapter is organized in the following manner: the fundamental components of our mathematical model characterizing the standing wave thermoacoustic Sterling heat engine are presented in Section 4.3. In Section 4.4, we discuss optimization with respect to a single objective, identifying globally optimal values of the variables with respect to the considered objective function while still satisfying the constraints. Section 4.5 considers multiobjective optimization, and concluding remarks are discussion in Section 4.6. A concise summary of the terminology we use is presented in the preliminary materials.

4.3 MATHEMATICAL MODELING OF THERMOACOUSTIC ENGINES

In this section we introduce the mathematical model we use to represent the underlying dynamics of thermoacoustic systems.

4.3.1 Model Components

The stack exhibits a symmetry which allows us to reduce the problem domain to two dimensions. We attribute two constant temperature boundaries, one convective boundary, and one adiabatic boundary to account for the thermal behavior of the device in the reduced domain. In the thermal calculations, we are primarily interested in the temperature distribution achieved in the domain, and discuss several approaches to determine the relevant temperature profiles. Acoustically, we represent the stack's work flow and viscous resistance using expressions constructed from several structural variables, that are in turn involved in a number of structural constraints. The variables are the parameters[1]that we allow to be

varied, while structural constraints are equations and inequalities that enforce restrictions on permissible variable combinations. We measure the quality of a given set of variable values using an objective function comprised of multiple components.

Multiobjective optimization is concerned with the optimization of more than one objective function that are conflicting in nature [70]. They are conflicting in the sense that, if optimized individually, they do not share the same optimal solutions. When optimized simultaneously, weights are typically added to each objective to allow the user to place desired emphasis. In this context, a Pareto optimal solution is one in which there does not exist another solution that strictly improves one of the considered objective components without worsening another objective component. The set of Pareto optimal solutions over all objective weights can then be used to generate the efficient frontier.

Variables. We characterize the fundamental properties of the stack using the following five lower- and upper-bounded variables:

- L: Stack length,
- H: Stack height,
- Z: Stack placement,
- d_c: Channel diameter, and
- N: Number of channels.

Figure 4.2 (from [114]) depicts these structural variables [114]. Both the stack length L and height H take real values between their bounds, where the stack height is defined as the radius of a cross section of the resonance tube. The placement of the stack in the axial direction of the resonator is modeled by continuous variable Z; near the closed end of the resonance tube its value approaches 0 from above. We take the maximum length of the resonator tube to be a quarter-wavelength, i.e., $Z_{max} = \frac{\lambda}{4}$, implying that Z can effectively range from Z_{min} to $Z_{max} - L$ to properly account for the stack length. Because the geometry of the porous stack is based on the monolith structure used in experimentation [115], we model it using square channels, and represent the channel size with continuous variable d_c, so that the channel perimeter

[1] We differentiate between the terms *variables* and *parameters*, in that we use the term *variables* to indicate the structural components we allow to fluctuate in order to improve the objective, and the term *parameters* to indicate either known quantities (i.e., constants) or auxiliary quantities that are completely dependent on the values of the structural variables and other constant parameters.

$\Pi_c = 4d_c$ and area $A_c = d_c^2$. We allow d_c to range from the thermal penetration depth δ_κ to $\mathcal{F}\delta_\kappa$, where $\mathcal{F}$ is an integer-valued multiplier on the thermal penetration depth. If we do not properly cap the size of the stack's channels, the key interaction between the gas and the wall does not occur, thus hindering the amplification of acoustic waves [93]. Hence we take $\mathcal{F}$ to be four because it results in a channel dimension that still yields thermoacoustic performance. Finally, we model the number of channels N within the stack as an integer-valued variable.

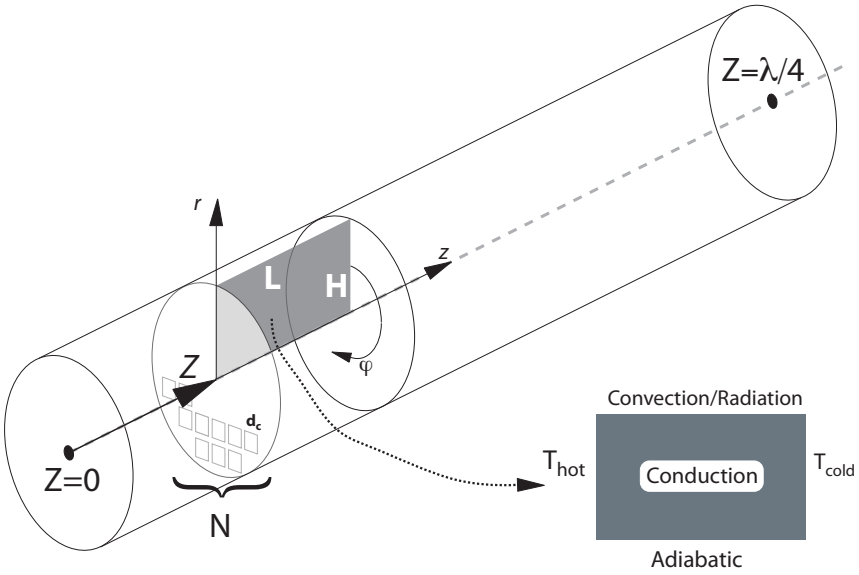


Figure 4.2: Computational domain and boundary conditions for variables L, H, Z, N, d_c

Objectives. Our objective function contains the following five components, each of which has a weighting factor w_i to provide appropriate user-defined emphasis:

- W: Work output,
- R_ν: Viscous resistance,
- Q_{conv}: Convective heat flow,
- Q_{rad}: Radiative heat flow, and
- Q_{cond}: Conductive heat flow.

The two acoustic objectives are the work output W of the thermoacoustic engine and the viscous resistance R_ν through the stack. The thermal objectives include both the convective

heat flow Q_{conv} and the radiative heat flow Q_{rad} that we evaluate at the top boundary of the stack, as well as the conductive heat flow Q_{cond} that we evaluate at the end of the resonance tube. Because work is the only objective to be maximized, we instead minimize its negative magnitude so that objective directions concur in all components.

As is common in multiobjective optimization, the objective function components in our model are of varying and conflicting magnitudes and units. We can restore this imbalance by incorporating normalization factors on each component weight w_i. Thus, without loss of generality, we make the assumption that weights w_i are normalized in our following discussions, which makes each objective function component unitless and nonnegative in magnitude. Section 4.5.1 provides further details on the normalization procedure we implement.

Structural Constraints. In addition to having lower and upper bounds, variables may only take values that satisfy certain physical properties governing the engine. One such property is that the total number of channels N of a given diameter d_c is limited by the cross-sectional radius of the resonance tube H. This relationship yields the constraint $\mathcal{A}N(d_c+t_w)^2 \leq \pi H^2$, where t_w represents the wall thickness around a single channel, and $\mathcal{A}$ represents the ratio of the area of a filled circle to its optimal packing by smaller square channels. From observations on optimal packings (see, e.g., [41]), $1 \leq \mathcal{A} \leq 1.5$, so we set $\mathcal{A} = 1.25$. Other model constraints equate auxiliary parameters used in the optimization.

4.3.2 Mathematical Programming Formulation

The following mathematical model (4.1) – (4.26) is a nonlinear mixed integer program.

$$\text{(MPF)} \quad \min_{L,H,Z,d_c,N} \quad \zeta = w_1(-W) + w_2 R_\nu + w_3 Q_{conv} + w_4 Q_{rad} + w_5 Q_{cond} \tag{4.1}$$

subject to

$$\mathcal{A}N(d_c + t_w)^2 \leq \pi H^2, \tag{4.2}$$

$$\begin{aligned} W &= 1/4\Pi_c\omega \left[\delta_\kappa \frac{(\gamma-1)p^2}{\rho c^2(1+\epsilon)}(\Gamma-1) - \delta_\nu \rho u^2\right] LN \\ &= \omega \left[\delta_\kappa \frac{(\gamma-1)p^2}{\rho c^2(1+\epsilon)}(\Gamma-1) - \delta_\nu \rho u^2\right] LNd_c, \end{aligned} \tag{4.3}$$

$$R_\nu = \frac{\mu \Pi_c}{A_c^2 \delta_\nu} \frac{L}{N} = \frac{4\mu}{\delta_\nu} \frac{L}{N d_c^3}, \tag{4.4}$$

$$Q_{conv} = H \int_0^{2\pi} \int_0^L h(T_s) \left(T_s - T_\infty\right) dz d\varphi, \tag{4.5}$$

$$Q_{rad} = H \; k_b \int_0^{2\pi} \int_0^L \epsilon \left(T_s^4 - T_\infty^4\right) dz d\varphi, \tag{4.6}$$

$$Q_{cond} = \int_0^{2\pi} \int_0^H \left(k_{rr} \frac{\partial T}{dr} + k_{zz} \frac{\partial T}{dz}\right) dr d\varphi, \tag{4.7}$$

$$Q_{cond}|_{z=L_{max}} = \int_0^{2\pi} \int_0^H \left(k_{zz} \frac{\partial T}{\partial z}\right) dr d\varphi. \tag{4.8}$$

Heat flow equations (4.5) – (4.8) depend on the following additional parameters:

$$h(T_s) = \frac{k_g}{2H} Nu, \tag{4.9}$$

$$Nu = 0.36 + \frac{0.518 Ra_D^{\frac{1}{4}}}{\left[1 + \left(\frac{0.559}{Pr}\right)^{\frac{9}{16}}\right]^{\frac{4}{9}}}, \tag{4.10}$$

$$Ra_D = Gr \; Pr = \frac{g\beta(T_s - T_\infty)}{\nu\alpha} (2H)^3, \tag{4.11}$$

$$Pr = \frac{\nu}{\alpha}, \tag{4.12}$$

$$k_{rr} = \frac{k_s k_g (t_w + d_c)}{k_s t_w + k_g d_c}, \tag{4.13}$$

$$k_{zz} = \frac{k_s t_w + k_g d_c}{t_w + d_c}. \tag{4.14}$$

The work expression (4.3) depends on the following four parameters:

$$\epsilon = \frac{(\rho c_p \delta_\kappa)_g}{(\rho c_p \delta_s)_s} \frac{tanh\left((i+1) y_0/\delta_\kappa\right)}{tanh\left((i+1) l/\delta_s\right)}, \tag{4.15}$$

$$u_{max} = \frac{p_{max}}{\rho c}, \tag{4.16}$$

$$p = p_{max} \cos\left(\frac{2\pi Z}{\lambda}\right), \tag{4.17}$$

$$u = u_{max} \sin\left(\frac{2\pi Z}{\lambda}\right). \tag{4.18}$$

The variables are subject to the following restrictions:

$$L_{min} \leq L \leq L_{max}, \tag{4.19}$$

$$H_{min} \leq H \leq H_{max}, \tag{4.20}$$

$$\delta_\kappa \leq d_c \leq \mathcal{F}\delta_\kappa, \tag{4.21}$$

$$Z_{min} \leq Z \leq Z_{max} - L, \tag{4.22}$$

$$N_{min} \leq N \leq N_{max}, \tag{4.23}$$

$$L,\ H,\ Z,\ d_c \in\ \mathbb{R}_+; N \in\ \mathbb{Z}_+. \tag{4.24}$$

The following boundary conditions must also be enforced:

i) Constant hot side temperature (T_h),

ii) Constant cold side temperature (T_c),

iii) Adiabatic boundary, modeling the central axis of the cylindrical stack:

$$\left.\frac{\partial T}{dr}\right|_{r=0} = 0, \quad \text{and} \tag{4.25}$$

iv) Free convection and radiation to surroundings (at T_∞) with temperature dependent heat transfer coefficient (h), emissivity (ε), and thermal conductivity (k):

$$k \left.\frac{\partial T}{dr}\right|_{r=H} = h\left(T_s - T_\infty\right) + \epsilon k_b \left(T_s^4 - T_\infty^4\right). \tag{4.26}$$

We denote by x the solution vector of structural variables, i.e., $x = [L, H, d_c, Z, N]$. Constraint (4.2) relates the channel diameter d_c and the number of possible channels N to the radius H of the cross-sectional area, while equations (4.3) – (4.7) express our five objective function components of interest. Equations (4.3) and (4.4) calculate the work W and viscous resistance R_ν, respectively, as functions of L, d_c, Z, and N (and indirectly H through (4.2)). Equations (4.5) – (4.8) represent heat flows. Equations (4.9) – (4.18) express parameters used in objective function components, (4.19) – (4.24) restrict variables values, and (4.25) – (4.26) represent heat flow boundary conditions. Note that u_{max} and p_{max} are related[2] at $z_o = 0$ as evidenced in equation (4.16).

Remark 1. *In equation (4.15), the real part of ϵ is observed to tend to $\frac{\sqrt{3}}{2}$, and we set ϵ to this value.*

Remark 2. *We set the hot-side temperature $T_h = \nabla T L + T_c$, where ∇T and T_c are predetermined values. Note that the constant temperature gradient ∇T is an approximation and its validity is assumed over the entire domain of structural variables (i.e., $L \in [L_{min}, L_{max}]$, $H \in [H_{min}, H_{max}]$, etc.). This behavior corresponds with experimental observations that clearly indicate a positive correlation between the stack length L and hot side temperature T_h in order to successfully sustain the thermoacoustic energy conversion. Additional details can be found in Section 4.3.3.1.*

Remark 3. *While the heat transfer coefficient h, in this case for natural convection, depends on the surface temperature T_s (a function of z), this value is calculated separately and treated as constant; see Section 4.3.3.2 for a related discussion.*

Remark 4. *We assume that constraint (4.2) is satisfied when variables H, N, and d_c are at their lower bounds, so that $\mathcal{A}N_{min}(d_{c_{min}} + t_w)^2 \leq \pi H_{min}^2$ holds.*

[2] p_{max} is determined either by an informed choice based on domain knowledge, or via simulation.

4.3.3 Approximating the Heat Flows

We next discuss how we arrived at equations (4.5) – (4.8), (4.25), and (4.26), including their approximation.

4.3.3.1 Estimating the Temperature Distribution Given an input H (and L), it is necessary to find the solution of the 2D temperature distribution in our reduced domain, subject to boundary conditions detailed above. Due to the nature of the boundary conditions, the analytical solution is very difficult. Numerical solvers such as COMSOL Multiphysics [29], MATLAB Finite Element Toolbox [68], etc. are another option to determine the temperature distribution. However, this precision comes at high computational cost. Considering that the temperature distribution is required for the estimation of the heat fluxes, only the temperature distribution at the shell surface and the temperature gradient at the cold side are of interest. For this purpose it is reasonable to reduce the temperature calculations to those two relevant values. The temperature distribution along the top surface can be well-approximated by an exponentially decaying temperature distribution throughout the domain. This behavior was determined through an analysis of the finite element solution. The final surface temperature distribution as a function of axial direction z is given by:

$$T_s = T_h e^{ln\left(\frac{T_c}{T_h}\right)\frac{z}{L}}. \tag{4.27}$$

This distribution is assumed to be valid on the surface characterized by $(z,\ r = H)$ and approximates the physical temperature distribution. This same temperature distribution is used to determine the axial temperature gradient at the cold side (required for the conductive heat flux). Considering again the rectangular domain, we can see that the temperature gradient in the center (i.e. bottom, $r = 0$) will vary linearly from T_h to T_c. Assuming that the temperature gradient at the cold side is exponential for all r will result in an underestimation of the conductive heat flux.

4.3.3.2 Determining the Heat Fluxes

The temperature distribution in (4.27) is then used to determine the convective and radiative heat transfer to the surroundings via:

$$Q_{conv} = 2\pi H h \int_0^L (T_s - T_\infty)\, dz. \tag{4.28}$$

As noted in Remark 3, the temperature dependent heat transfer coefficient $h(T)$ is determined in a preprocessor (derived from the appropriate Nusselt law, as stated in equation (4.10) and an average surface temperature), and is not considered as part of the integral. The radiative heat transfer (in the general case) is written as:

$$Q_{rad} = 2\pi H k_B \varepsilon \int_0^L \left(T_s^4 - T_\infty^4\right) dz, \tag{4.29}$$

which depends on the surface emissivity ε and Stefan-Boltzmann constant k_B, both of which are assumed to be independent of temperature.

After integrating we derive the following heat flow expressions:

$$Q_{conv} = 2\pi H L h \left[\frac{T_h}{ln\left(\frac{T_c}{T_h}\right)} \left(\frac{T_c}{T_h} - 1\right) - T_\infty \right], \quad \text{and} \tag{4.30}$$

$$Q_{rad} = 2\pi H L k_B \varepsilon \left[\frac{T_h^4 \left(e^{4ln\left(\frac{T_c}{T_h}\right)} - 1\right)}{4ln\left(\frac{T_c}{T_h}\right)} - T_\infty^4 \right]. \tag{4.31}$$

In the present case, this approximation of the temperature distribution (equation (4.27)) is also utilized to determine the conductive heat flow at $z = L$. The temperature distribution throughout the $2D$ domain implies that this estimate will fall between the extremes of:

i) the physical case (under anisotropic material properties and physical boundary conditions), and

ii) the assumption of constant temperature gradient determined as $\frac{dT}{dz} = \frac{T_h - T_c}{L}$, as the latter case only exists at the adiabatic boundary $z,\ r = 0$ and quickly loses validity.

At the top surface $z,\ r = H$ the exponential distribution is assumed, so we determine the temperature gradient using this temperature distribution. Determining

$$\left.\frac{\partial T}{\partial z}\right|_{z=L} = \frac{T_c}{L} ln\left(\frac{T_c}{T_h}\right) \tag{4.32}$$

and implementing this in the general statement of the Fourier law of thermal conduction, we can express this heat flow as:

$$Q_{cond} = \frac{k_{zz}}{L}\pi H^2 T_c ln\left(\frac{T_h}{T_c}\right). \tag{4.33}$$

This expression for the conductive heat flow depends on the effective thermal conductivity in the z-direction as defined in equation (4.14). Using mild assumptions, equations (4.30), (4.31) and (4.33) give expressions for the heat flows that, while still nonlinear, no longer require external finite element solvers to evaluate.

4.4 SINGLE OBJECTIVE OPTIMIZATION

We have presented a mathematical model that characterizes the essential elements of a standing wave thermoacoustic engine. Based on the discussion in Section 4.3.3.2, our nonlinear model can be solved independently of finite element solvers. In the following discussion we analyze restricted cases of our objectives, and identify general tendencies of the structural variables to influence individual objective components.

4.4.1 Acoustic Emphasis

The following two sections analyze the cases where objective function (4.1) is restricted to optimizing work and viscous resistance, respectively.

4.4.1.1 Emphasizing Work

Setting the objective function weights to $w_2 = w_3 = w_4 = w_5 = 0$ and $w_1 = 1$, the problem reduces to constraints (4.2), (4.3), (4.16) – (4.18), and variable restrictions (4.19) – (4.24). Objective function (4.1) becomes:

$$\min_{L,H,d_c,Z,N} \quad \zeta_W = (-W). \tag{4.34}$$

By incorporating (4.16) – (4.18) into the initial term of equation (4.3) (which is a function of Z through p and u), and defining $f_W(Z)$ as:

$$f_W(Z) = \omega\delta_\kappa \frac{(\gamma - 1)\left[p_{max}\cos\left(\frac{2\pi Z}{\lambda}\right)\right]^2}{\rho c^2(1+\epsilon)}(\Gamma - 1) - \omega\delta_\nu\rho\left[u_{max}\sin\left(\frac{2\pi Z}{\lambda}\right)\right]^2, \tag{4.35}$$

we can then express work as:

$$W = f_W(Z)LNd_c. \tag{4.36}$$

Because work W has a physically nonnegative interpretation, this implies $f_W(Z) \geq 0$, and because for our problem parameters $Z \leq \frac{\lambda}{4} - L$, it is favorable to set $Z^* = Z_{min}$. Also, because it appears nowhere else in the reduced problem, we set $L^* = L_{max}$. Regarding the remaining terms N and d_c, increasing either also improves the objective, but consumes limited resources as per constraint (4.2). Setting $H^* = H_{max}$ to allow both N and d_c to increase, equation (4.2) simplifies to:

$$\mathcal{A}N(d_c + t_w)^2 \leq \pi H_{max}^2. \tag{4.37}$$

Letting $c_W = -f_W(Z_{min})L_{max}$ and substituting equation (4.3) into (4.34) and rearranging gives:

$$\min_{d_c,N} \quad \zeta_W = c_W N d_c, \tag{4.38}$$

subject to (4.21), (4.23), (4.24), and (4.37).

It follows from (4.21), (4.37) and (4.38) that d_c takes an upper bound of:

$$d_c = \min\left\{\mathcal{F}\delta_\kappa, \sqrt{\frac{\pi}{\mathcal{A}N}}H_{max} - t_w\right\}. \tag{4.39}$$

The first component of (4.39) is constant, and the second is monotonically decreasing in N. From (4.37) it also follows that:

$$N \leq \left\lfloor \frac{\pi H_{max}^2}{\mathcal{A}(d_c + t_w)^2} \right\rfloor, \tag{4.40}$$

so we define $N_{min} = 1$ and, because $\delta_\kappa \leq d_c$, we define $N_{max} = \left\lfloor \frac{\pi H_{max}^2}{\mathcal{A}(\delta_\kappa + t_w)^2} \right\rfloor$. Now considering the continuous value of N for which the two components in (4.39) are equal, let $\tilde{N} = \frac{\pi H_{max}^2}{\mathcal{A}(\mathcal{F}\delta_\kappa + t_w)^2}$. This leaves us with two cases:

i) for $N : N_{min} \leq N \leq \left\lfloor \tilde{N} \right\rfloor$, we have $d_c = \mathcal{F}\delta_\kappa$, and

ii) for $N : \left\lceil \tilde{N} \right\rceil \leq N \leq N_{max}$, we have $d_c = \sqrt{\frac{\pi}{\mathcal{A}N}} H_{max} - t_w$.

Let us temporarily consider relaxing the integer restriction on N from (4.24), and let N_c take continuous values over the domain of N, i.e. $N_{min} \leq N_c \leq N_{max}$. Viewing the two cases above in light of N_c and (4.38) gives:

i) $\zeta_W = c_W N_c \mathcal{F}\delta_\kappa$ for $N_c : N_{min} \leq N_c \leq \tilde{N}$.

Because $c_W < 0$, ζ_W is a monotonically decreasing function in terms of N_c, and so the optimal value of N_c over this domain is the largest value it can obtain, $N^* = \tilde{N}$.

ii) $\zeta_W = c_W N_c \left(\sqrt{\frac{\pi}{\mathcal{A}N_c}} H_{max} - t_w \right)$ for $N_c : \tilde{N} \leq N_c \leq N_{max}$.

For this case the first and second derivatives of ζ_W are, respectively:

$$\frac{d\zeta_W}{dN_c} = c_W \left[\sqrt{\frac{\pi}{4\mathcal{A}N_c}} H_{max} - t_w \right], \tag{4.41}$$

$$\frac{d^2\zeta_W}{dN_c^2} = -c_W \sqrt{\frac{\pi}{16\mathcal{A}N_c^3}} H_{max}. \tag{4.42}$$

Because $c_W < 0$, over the domain $N_c : \tilde{N} \leq N_c \leq N_{max}$ the second derivative of $\zeta_W > 0$, implying convexity of ζ_W and so ζ_W has a single global minimum. Setting the first derivative in (4.41) equal to zero and solving, the minimal value of ζ_W occurs at $N_c = \frac{\pi H_{max}^2}{4\mathcal{A}t_w^2}$. Because of the convexity of ζ_W in this region, then if $\tilde{N} \leq \frac{\pi H_{max}^2}{4\mathcal{A}t_w^2} \leq N_{max}$, we have $N_c^* = \frac{\pi H_{max}^2}{4\mathcal{A}t_w^2}$. Otherwise, $\tilde{N} > \frac{\pi H_{max}^2}{4\mathcal{A}t_w^2}$, and in this case ζ_W is increasing on the interval $\left[\tilde{N}, N_{max} \right]$, and so $N_c^* = \tilde{N}$.

In light of the previous two cases, to ensure $N \in \mathbb{Z}_+$ we have:

$$N^* \in \left\{ \left\lfloor \tilde{N} \right\rfloor, \left\lceil \tilde{N} \right\rceil, \left\lfloor \frac{\pi H_{max}^2}{4\mathcal{A}t_w^2} \right\rfloor, \left\lceil \frac{\pi H_{max}^2}{4\mathcal{A}t_w^2} \right\rceil \right\}, \quad \text{and} \tag{4.43}$$

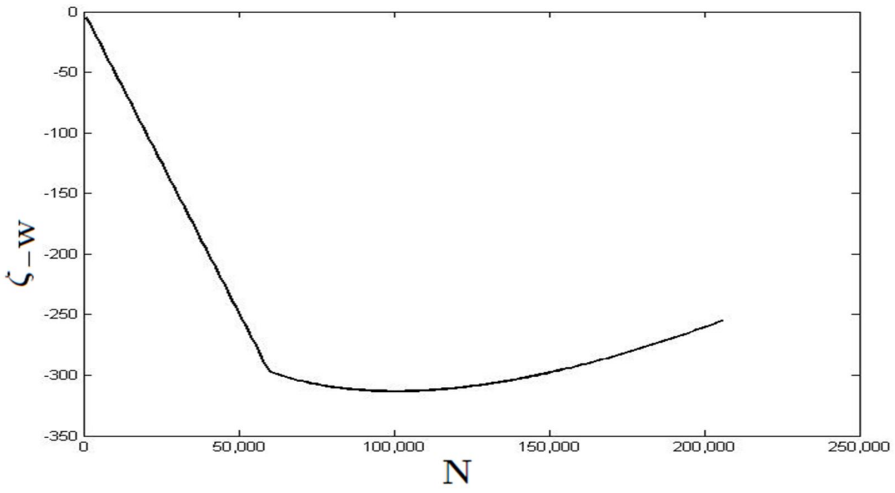


Figure 4.3: ζ_W plotted as a function of N and showing minimum

$$d_c^* = \begin{cases} \mathcal{F}\delta_\kappa & \text{if } N^* = \left\lfloor \tilde{N} \right\rfloor; \\ \sqrt{\frac{\pi}{\mathcal{A}N^*}} H_{max} - t_w & \text{otherwise.} \end{cases} \tag{4.44}$$

We then choose from (4.43) and (4.44) the values of N and corresponding d_c that minimize (4.38). Based upon our specific problem data, a global minimizer of ζ_W is:

$$x^* = \left[L_{max}, H_{max}, \sqrt{\frac{\pi}{\mathcal{A}N^*}} H_{max} - t_w, Z_{min}, \left\lceil \frac{\pi H_{max}^2}{4\mathcal{A}t_w^2} \right\rceil \right].$$

Figure 4.3 plots ζ_W as a function of N, showing the value of $N^* = \left\lceil \frac{\pi H_{max}^2}{4\mathcal{A}t_w^2} \right\rceil$ that minimizes ζ_W.

We can physically interpret this optimal solution as making the stack as long and wide as possible, and making the thermoacoustically active surface area as large as possible by increasing the number of channels N and the channel diameter d_c. Additionally, positioning the stack near the closed end maximizes the available pressure amplitude for the thermodynamic cycle and thus work output W.

4.4.1.2 Emphasizing Viscous Resistance

We emphasize R_ν by setting objective function weights $w_1 = w_3 = w_4 = w_5 = 0$ and $w_2 = 1$. The problem simplifies to (4.2), (4.4), and (4.19) – (4.24). Objective function (4.1) becomes:

$$\min_{L,H,d_c,Z,N} \quad \zeta_{R_\nu} = R_\nu. \tag{4.45}$$

Set Z^* to any value between its lower and upper bounds, e.g. $Z^* = Z_{min}$, and set $L^* = L_{min}$. Variables N and d_c are constrained by (4.2); setting $H^* = H_{max}$ affords the greatest flexibility for N and d_c to increase. Let $c_{R_\nu} = \frac{4\mu L_{min}}{\delta_\nu}$, then (4.45) can be rewritten as:

$$\min_{d_c,N} \quad \zeta_{R_\nu} = \frac{c_{R_\nu}}{N d_c^3}, \tag{4.46}$$

subject to (4.2), (4.23), (4.21), and (4.24). Much of the discussion in Section 4.4.1.1 concerning d_c still holds, e.g. equations (4.37), (4.39) and (4.40). Maintaining our definition of $\tilde{N}$, the following two cases remain:

i) for $N : N_{min} \leq N \leq \left\lfloor \tilde{N} \right\rfloor$, we have $d_c = \mathcal{F}\delta_\kappa$, and
ii) for $N : \left\lceil \tilde{N} \right\rceil \leq N \leq N_{max}$, we have $d_c = \sqrt{\frac{\pi}{\mathcal{A}N}} H_{max} - t_w$.

Instead of minimizing ζ_{R_ν} as in (4.46), let us instead maximize $\overline{\zeta}_{R_\nu} = N d_c^3$, as the optimal values of N^* and d_c^* are identical. As in Section 4.4.1.1 we temporarily consider relaxing the integer restriction on N from (4.24), allowing N_c to take continuous values over the domain of N, i.e. $N_{min} \leq N_c \leq N_{max}$. Viewing these two cases in light of N_c and $\overline{\zeta}_{R_\nu}$ gives:

i) $\overline{\zeta}_{R_\nu} = N_c(\mathcal{F}\delta_\kappa)^3$ for $N_c : N_{min} \leq N_c \leq \tilde{N}$.
   Here, $\overline{\zeta}_{R_\nu}$ is a monotonically increasing function in terms of N_c, and so the optimal value of N_c over this domain is the largest value it can obtain, $\tilde{N}$.
ii) $\overline{\zeta}_{R_\nu} = N_c \left(\sqrt{\frac{\pi}{\mathcal{A}N_c}} H_{max} - t_w \right)^3$ for $N_c : \tilde{N} \leq N_c \leq N_{max}$.
   Over this interval, differentiating $\overline{\zeta}_{R_\nu}$ gives:

$$\frac{d\overline{\zeta}_{R_\nu}}{dN_c} = \frac{3t_w^2 \pi^{\frac{1}{2}} H_{max}}{2\mathcal{A}^{\frac{1}{2}}} N_c^{-\frac{1}{2}} - \frac{1}{2}\left(\frac{\pi}{\mathcal{A}} H_{max}^2\right)^{\frac{3}{2}} N_c^{-\frac{3}{2}} - t_w^3, \tag{4.47}$$

   and upon a second differentiation, we obtain:

$$\frac{d^2\overline{\zeta}_{R_\nu}}{dN_c^2} = \frac{3\pi^{\frac{3}{2}} H_{max}^3}{4\mathcal{A}^{\frac{3}{2}}} N_c^{-\frac{5}{2}} - \frac{3t_w^2}{4}\left(\frac{\pi}{\mathcal{A}} H_{max}^2\right)^{\frac{1}{2}} N_c^{-\frac{3}{2}}. \tag{4.48}$$

The second derivative of $\overline{\zeta}_{R_\nu} > 0$ over the entire domain $N_c : \tilde{N} < N_c \leq N_{max}$, implying $\overline{\zeta}_{R_\nu}$ is convex. Thus the maximum over this domain occurs at one of the endpoints of the interval, i.e., $N_c^* \in \left\{\tilde{N}, N_{max}\right\}$.

From these two cases, and because $N \in \mathbb{Z}_+$ we have:

$$N^* \in \left\{ \left\lfloor \tilde{N} \right\rfloor, \left\lceil \tilde{N} \right\rceil, N_{max} \right\}, \quad \text{and} \tag{4.49}$$

$$d_c^* = \begin{cases} \mathcal{F}\delta_\kappa & \text{if } N^* = \left\lfloor \tilde{N} \right\rfloor; \\ \sqrt{\frac{\pi}{\mathcal{A}N^*}} H_{max} - t_w & \text{otherwise.} \end{cases} \tag{4.50}$$

We then choose from (4.49) and (4.50) the values of N and corresponding d_c that minimize (4.46). For our specific problem parameters, a global minimizer for ζ_{R_ν} is:

$$x^* = \left[L_{min}, H_{max}, \mathcal{F}\delta_\kappa, Z_{min}, \left\lfloor \tilde{N} \right\rfloor \right].$$

Figure 4.4 plots ζ_{R_ν} as a function of N, illustrating the the value of $N^* = \left\lfloor \tilde{N} \right\rfloor = \left\lfloor \frac{\pi H_{max}^2}{\mathcal{A}(\mathcal{F}\delta_\kappa + t_w)^2} \right\rfloor$ that minimizes ζ_{R_ν}.

This result can be physically interpreted as reducing the individual (viscous) resistance of each channel to its minimum by decreasing their length ($L^* = L_{min}$) and then bundling as many of those small resistances in parallel to further lower the net resistance. This is illustrated by the addition of the respective inverse resistances when arranged in parallel to determine a net resistance:

$$R_{net} = \left[\sum_{i=1}^{N} \frac{1}{R_i} \right]^{-1}. \tag{4.51}$$

For instance, in the case where all R_i have the same value, this equation reduces to $R_{net} = \frac{R_i}{N}$, indicating that increasing N leads a lowered resistance.

4.4.2 Thermal Emphasis

We have thus far considered how acoustic objectives W and R_ν are affected by changes in the structural variables. We next discuss the individual thermal objectives by isolating each heat flow objective function component.

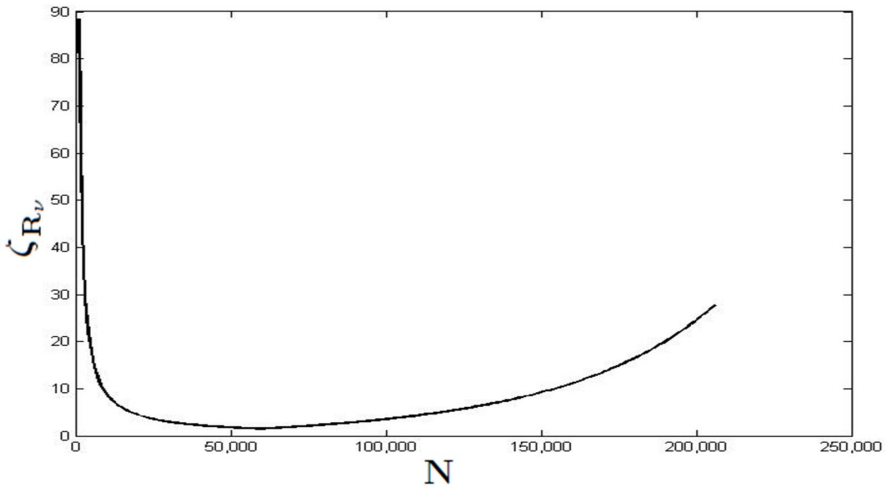


Figure 4.4: ζ_{R_ν} plotted as a function of N and showing minimum

4.4.2.1 Emphasizing Convective / Radiative Heat Fluxes We can emphasize Q_{conv} by setting objective function weights $w_1 = w_2 = w_4 = w_5 = 0$ and $w_3 = 1$. The problem then reduces to constraints (4.2), (4.9) – (4.12), variable restrictions (4.19) – (4.24), and expression (4.30). Alternatively, we can emphasize Q_{rad} by setting objective function weights $w_1 = w_2 = w_3 = w_5 = 0$ and $w_4 = 1$, so that the only constraints and variable restrictions that are active are (4.2), (4.15), (4.19) – (4.24), and (4.31). For these restricted optimization problems, objective function (4.1) becomes, respectively:

$$\min_{L,H,d_c,Z,N} \quad \zeta_{Q_{conv}} = Q_{conv}; \qquad \min_{L,H,d_c,Z,N} \quad \zeta_{Q_{rad}} = Q_{rad}. \tag{4.52}$$

Neither of these restricted models are dependent on Z, so Z^* can be set to any value between its lower and upper bounds (note our assumption that h is not dependent on Z in Remark 3). Considering H, for Q_{cond} it can be shown from equations (4.9) – (4.12) that h is proportional to $H^{-1/4}$. Because the resulting exponent on the H variable remains positive in equation (4.30), it is still desirable to set $H^* = H_{min}$. For Q_{rad} we also set H to $H^* = H_{min}$ based on (4.31). Setting $N^* = N_{min}$ and $d_c^* = d_{c_{min}}$ ensures that H can take its minimum value in constraint (4.2) (see assumption in Remark 4).

The global optimum $x^* = [L_{min}, H_{min}, d_{c_{min}}, Z_{min}, N_{min}]$ minimizes both $\zeta_{Q_{conv}}$ and $\zeta_{Q_{rad}}$. For Q_{cond}, this optimum reduces the surface area and limits the temperature range in the stack, thereby minimizing the convective heat flow. Similarly for Q_{rad}, this optimum lowers driving potential and surface area to reduce the radiative heat flow.

4.4.2.2 Emphasizing Conductive Heat Flux We emphasize Q_{cond} by setting objective function weights $w_1 = w_2 = w_3 = w_4 = 0$ and $w_5 = 1$, so that only (4.2), (4.14), (4.19) – (4.24), and (4.33) are active. Objective function (4.1) becomes:

$$\min_{L,H,d_c,Z,N} \quad \zeta_{Q_{cond}} = Q_{cond}. \tag{4.53}$$

Similar to previous sections, this model is not dependent on Z, so that Z^* can be set to any value between its lower and upper bounds. Equation (4.14) can be rearranged as:

$$k_{zz} = k_g + \frac{t_w(k_s - k_g)}{(t_w + d_c)}, \tag{4.54}$$

and so merging (4.54) with (4.33) and rearranging gives:

$$Q_{cond} = \pi T_c \left[k_g + \frac{t_w(k_s - k_g)}{(t_w + d_c)}\right] \left[ln\left(\frac{\nabla T L + T_c}{T_c}\right)\frac{1}{L}\right] H^2. \tag{4.55}$$

Because $\left[ln\left(\frac{\nabla T L + T_c}{T_c}\right)\frac{1}{L}\right] > 0$, then setting L^* to L_{max} decreases Q_{cond}, improving (4.53). We can also improve Q_{cond} by both decreasing H and increasing d_c. However, there is tension in constraint (4.2) between decreasing H and increasing d_c. Because N appears only on the left-hand side of (4.2), we can set $N^* = N_{min} = 1$ to allow d_c and H the most flexibility. Letting $c_{Q1} = \pi T_c k_g \left[ln\left(\frac{\nabla T L_{max} + T_c}{T_c}\right)\frac{1}{L_{max}}\right]$ and $c_{Q2} = \pi T_c \left[t_w(k_s - k_g)\right] \left[ln\left(\frac{\nabla T L_{max} + T_c}{T_c}\right)\frac{1}{L_{max}}\right]$, and noting both are positive, then substituting these into (4.55) and (4.53) gives the following optimization problem over two continuous variables:

$$\min_{H,d_c} \quad \zeta_{Q_{cond}} = c_{Q1} H^2 + c_{Q2}\frac{H^2}{(t_w + d_c)}, \tag{4.56}$$

subject to constraint (4.2) and variable restrictions (4.20), (4.21), and (4.24).

For fixed H, it follows from our discussions and (4.2) that d_c will take the value of:

$$d_c = \min\left\{\mathcal{F}\delta_\kappa, \sqrt{\frac{\pi}{\mathcal{A}}}H - t_w\right\}. \tag{4.57}$$

Let $\tilde{H} = \frac{\mathcal{F}\delta_\kappa + t_w}{\sqrt{\frac{\pi}{\mathcal{A}}}}$ be the value of H for which the value of d_c transitions in (4.57). This leaves us with two cases:

i) for $H : H_{min} \leq H \leq \tilde{H}$, we have $d_c = \sqrt{\frac{\pi}{\mathcal{A}}} H - t_w$, and

ii) for $H : \tilde{H} \leq H \leq H_{max}$, we have $d_c = \mathcal{F}\delta_\kappa$.

For both intervals $\zeta_{Q_{cond}}$ is nondecreasing, so that $H^* = H_{min}$, implying $d_c^* = \sqrt{\frac{\pi}{\mathcal{A}}} H_{min} - t_w$. Thus $x^* = \left[L_{max}, H_{min}, \sqrt{\frac{\pi}{\mathcal{A}}} H_{min} - t_w, Z_{min}, N_{min}\right]$ is a global minimizer of $\zeta_{Q_{cond}}$.

The optimal solution with respect to minimizing the conductive heat flow differs from those of the convective and radiative heat flows. For an actual engine design this information may be useful in designing stacks that require the least amount of cooling for a given input.

4.4.3 Single Objective Optima: Variable Analysis

Table 4.1: Tendency of structural variables when optimizing individual objective components

	$(-)W$	R_ν	Q_{conv}	Q_{rad}	Q_{cond}
L	$\uparrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\uparrow$
H	$\uparrow$	$\uparrow$	$\downarrow$	$\downarrow$	$\downarrow$
$\mathbf{d_c}$	$\uparrow^\dagger$	$\uparrow^\dagger$	$\downarrow$	$\downarrow$	$\uparrow$
Z	$\downarrow^\ddagger$	$\longleftrightarrow$	$\longleftrightarrow$	$\longleftrightarrow$	$\longleftrightarrow$
N	$\uparrow^\dagger$	$\uparrow^\dagger$	$\downarrow$	$\downarrow$	$\downarrow$

Table 4.1 summarizes the results of Sections 4.4.1 and 4.4.2. It highlights the behavior of the structural variables, along the left, when individually optimizing the five objective function components that appear across the top. For these objectives, $\uparrow$ indicates an increasing tendency, $\downarrow$ indicates a decreasing tendency, and $\longleftrightarrow$ indicates no impact, while †indicates there is conflicting tension between variables. ‡indicates that Z can be set to Z_{min} in all cases, as only objective W depends on it, and decreasing it improves this objective while having no effect on the other objectives (based on our assumption in Remark 3). Also note the lack of tension in variables for the Q_{conv} and Q_{rad} heat flows, which share the same optimal solution.

4.5 MULTIOBJECTIVE OPTIMIZATION

In Section 4.4 we examine optimization over every individual component of objective function (4.1), providing analytical solutions that do not require computational solution methods to identify global optima. In this section we consider multiple objective components simultaneously, and suggest straightforward algorithmic approaches to identify optimal solutions for these cases. Before proceeding, we first discuss our approach to ensure objective function weights are normalized, commonplace in multiobjective optimization.

4.5.1 Normalizing Objective Function Components

When multiple objective function components are given nonzero weights, objective function (4.1) of (MPF) can have a predisposed bias towards those components having larger magnitudes, and unit discrepancies across the various objective components create further complications. These issues can be simultaneously addressed for each objective component by obtaining a normalization factor to offset any such disparities.

Our proposed normalization approach is based on a method described in [46]. Let a set $\mathcal{I}$ of objective components of interest from objective (4.1) be indexed by $i \in \mathcal{I}$. As (MPF) contains five objective components, $|\mathcal{I}| \leq 5$. Then for all indices $j \notin \mathcal{I}$, we set $w_j = 0$. For normalization coefficients n_i the approach uses the differences of values between certain *Utopia* and *Nadir* vectors that are of the same dimension as the number of considered objective function components $|\mathcal{I}|$, and are formed using information obtained from independent optimization of each objective function component.

The *Utopia* vector $\mathcal{U}$ is created as follows. For each $i \in \mathcal{I}$, we set $w_i = 1$ and $w_k = 0 \; \forall \; k \in \mathcal{I} : k \neq i$. Let $\mathcal{G}_i$ be the selected objective component. Optimizing the resulting reduced problem generates optimal objective function value $\mathcal{G}_i^*$ and optimal solution $x_i^* = [L_i^*, H_i^*, d_{c_i}^*, Z_i^*, N_i^*]$. Then $\mathcal{U}_i = \mathcal{G}_i^*$. After repeating this process for all $i \in \mathcal{I}$, the *Nadir* vector $\mathcal{N}$ makes use of the optimal solutions x_i^* from these optimizations, evaluating each x_i^* in the respective individual objective functions $\mathcal{G}_i^*$ over all $i \in \mathcal{I}$ to find its worst value. Thus, the Nadir vector is constructed as $\mathcal{N}_i = \max\limits_{\ell=1,\ldots,5} \{\mathcal{G}_i(x_\ell^*)\} \; \forall \; i \in \mathcal{I}$.

For each $i \in \mathcal{I}$, the differences $\mathcal{N}_i - \mathcal{U}_i$ provide the length of interval over which the optimal objective functions vary within the set of optimal solutions; note that these differences are always nonnegative. They are used to construct the normalization factors n_i as:

$$n_i = \frac{1}{\mathcal{N}_i - \mathcal{U}_i}. \tag{4.58}$$

For instance, if we consider for $\mathcal{I}$ all five of the objective components of objective function (4.1), then it can be normalized as:

$$\frac{w_1}{n_1}((-W) - \mathcal{U}_1) + \frac{w_2}{n_2}(R_\nu - \mathcal{U}_2) + \frac{w_3}{n_3}(Q_{conv} - \mathcal{U}_3) + \frac{w_4}{n_4}(Q_{rad} - \mathcal{U}_4) + \frac{w_5}{n_5}(Q_{cond} - \mathcal{U}_5). \tag{4.59}$$

We use this normalization scheme for all cases involving multiple objective function components. Note that the Utopia values are subtracted from every component so to ensure that the term is unitless and nonnegative, thereby eliminating any bias of magnitude.

4.5.2 Emphasizing Work and Viscous Resistance

We can simultaneously optimize the acoustic objectives W and R_ν by assigning objective weights $w_3 = w_4 = w_5 = 0$ with $w_1 > 0$, $w_2 > 0$. Then (MPF) reduces to constraints (4.2), (4.3), (4.16) – (4.18) and variable restrictions (4.19) – (4.24). Objective function (4.1) reduces to:

$$\min_{L,H,d_c,Z,N} \quad \zeta_{Acoustic} = w_1(-W) + w_2 R_\nu. \tag{4.60}$$

With respect to (4.35), let $\overline{c}_W = -w_1 f_W(Z_{min})$ and $\overline{c}_{R_\nu} = w_2 \frac{4\mu}{\delta_\nu}$, so that $\overline{c}_W$ and $\overline{c}_{R_\nu}$ are, respectively, the constant terms from Sections 4.4.1.1 and 4.4.1.2 without fixing L. Setting $Z^* = Z_{min}$ and $H^* = H_{max}$ as in Sections 4.4.1.1 and 4.4.1.2, and substituting $\overline{c}_W$, $\overline{c}_{R_\nu}$, W and R_ν into objective function (4.60) gives:

$$\min_{L,N,d_c} \quad \zeta_{Acoustic} = \left(\overline{c}_W N d_c + \frac{\overline{c}_{R_\nu}}{N d_c^3}\right) L, \tag{4.61}$$

subject to (4.19), (4.23), (4.21), (4.24), and (4.37). The tradeoffs between variables L, N and d_c can be investigated by first fixing N to $\overline{N} \in [N_{min}, N_{max}] \cap \mathbb{Z}$, then using $\overline{N}$ in

equation (4.39) to fix d_c to $\overline{d_c} = \min\left\{\mathcal{F}\delta_\kappa, \sqrt{\frac{\pi}{\mathcal{A}\overline{N}}}H_{max} - t_w\right\}$. Depending on the sign of the resulting coefficient on L in (4.61), L can be set to:

$$\overline{L} = \begin{cases} L_{min} & \text{if } \left(\overline{c}_W \overline{N}\ \overline{d_c} + \frac{\overline{c}_{R_\nu}}{\overline{N}\ \overline{d_c}^3}\right)\ \geq\ 0; \\ L_{max} & \text{otherwise.} \end{cases} \tag{4.62}$$

Thus for every fixed $\overline{N}$ the problem has a fixed value of $\overline{d_c}$ and $\overline{L}$. The optimal levels of L^*, N^* and d_c^* can be found by enumerating over all values $\overline{N} \in [N_{min}, N_{max}] \cap \mathbb{Z}$. We implement such an approach in MATLAB [68], which takes at most a few minutes to solve on a standard Windows XP-based machine with a 2.16GHz Intel Core2 processor with 2GB of RAM.

By iterating over multiple sets of objective function weights w_1 and w_2, the frontier of efficient points can be generated that optimize the respective acoustic objectives. This frontier is partially illustrated in Figure 4.5.

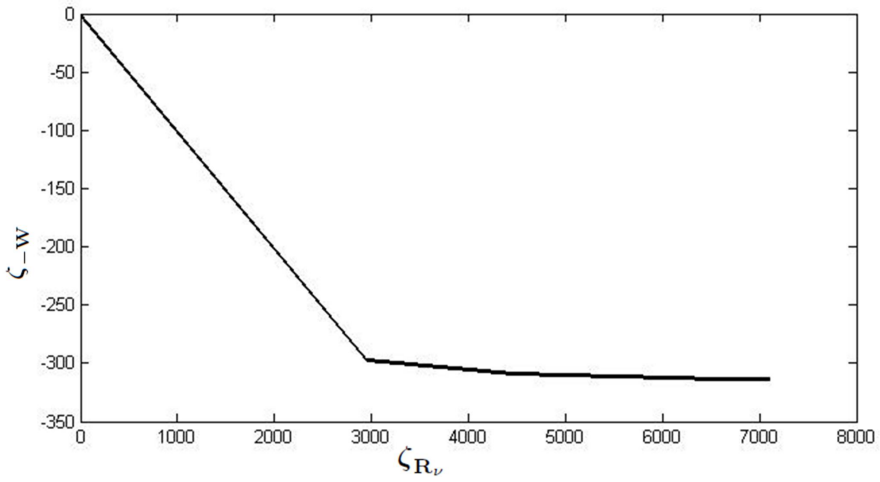


Figure 4.5: Acoustic efficient frontier: simultaneously minimizing $-W$ and R_ν

Setting the stack radius H to its largest value both maximizes the work by allowing many channels N while simultaneously reducing the viscous resistance (as per discussion in Section 4.4.1.2). Depending on the values of weights w_1 and w_2, the optimal length L is either at its lower or upper bound. Moving the stack nearer to the closed end increases the available pressure amplitude for the thermodynamic cycle that increases work output W without adversely impacting R_ν.

4.5.3 Emphasizing All Objective Components

Lastly, we simultaneously consider all five objective components by regarding work W and viscous resistance R_ν as two distinct objective components, and representing heat with a third distinct objective component Q_{all}, defined as the sum of the three heat components Q_{conv}, Q_{rad}, and Q_{cond}. We use three weights, w_W, w_{R_ν} and $w_{Q_{all}}$, and divide $w_{Q_{all}}$ equally among the three heat components comprising Q_{all}.

As in Section 4.5.2, we determine the frontier of efficient points that optimize the three weighted objectives W, R_ν, and Q_{all} by iterating over multiple values of objective function weights w_W, w_{R_ν} and $w_{Q_{all}}$. However, due to the lack of a closed form solution over the considered objective function components, this requires an optimization approach that can identify globally optimal solutions.

For fixed values of w_W, w_{R_ν} and $w_{Q_{all}}$, we call the global optimization routine DIRECT [80], a derivative free algorithm based on Lipschitzian optimization with proven finite convergence. Algorithm 8, which appears below, begins by constructing a hyper-rectangle that contains the original (continuous) variable space, and progressively improves the objective by repeatedly subdividing hyper-rectangles as it moves toward the global optimum. The particular implementation we use is due to Finkel [40], and coded in MATLAB.

Algorithm 8. *[Iterative Global Branch-and-Bound Algorithm]*

1. *Set incumbent objective function value $\overline{\mathcal{Z}} = +\infty$, objective function components weights w_W, w_{R_ν}, and $w_{Q_{all}}$, and initial lower and upper variable bounds for H, L, Z and d_c.*
2. *Assign fixed value of N as $\overline{N} := N_{min}$.*
3. *Pass to DIRECT $\overline{N}$ and variable bounds for H, L, Z, and d_c to optimize (MPF).*
4. *Obtain the resulting optimal solution and optimal objective function value $\mathcal{Z}$.*
5. ***If** $\mathcal{Z} < \overline{\mathcal{Z}}$, set $\overline{\mathcal{Z}} := \mathcal{Z}$ and save off optimal solution.*
6. *Assign $\overline{N} := \overline{N} + 1$.*
7. ***If** $\overline{N} < N_{max}$, **Go To** 3.*
8. ***Return** $\overline{\mathcal{Z}}$ and associated optimal solution, and **Stop**.*

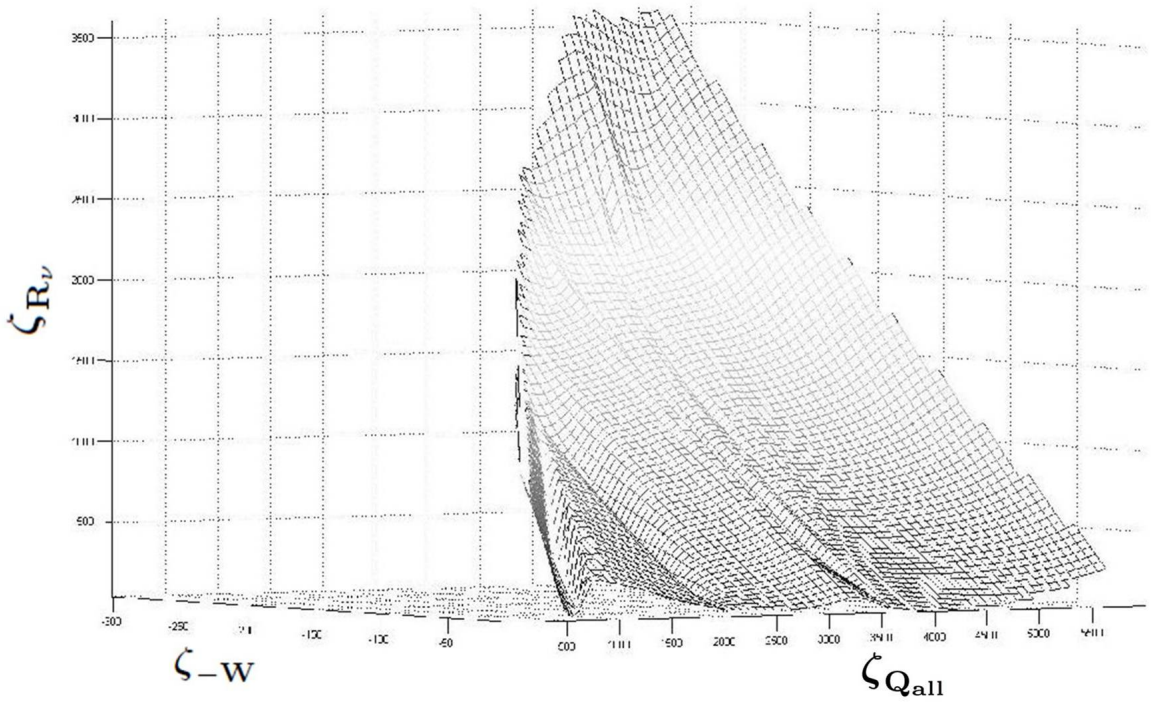


Figure 4.6: Side profile of efficient frontier: simultaneously minimizing $-W$, R_ν, and Q_{all}

Algorithm 8 generates an optimal solution corresponding to any set of weights w_W, w_{R_ν} and $w_{Q_{all}}$. These optimal solutions can then be used to construct the efficient frontier of optimal solutions, which is partially illustrated by the fitted surface appearing in Figures 4.6 and 4.7. Note the conflicting nature of the three objectives that can be observed in both profiles. Figure 4.6 provides a side profile of the efficient frontier, where the bottom left corner is improving for all three objectives, and illustrates how an improvement in a single objective component causes the remaining two objectives to worsen. Figure 4.7 depicts the same phenomenon from a top profile, where the rear corner is improving for every objective.

4.5.4 Alternative View: Maximizing Efficiency

An alternative way to simultaneously maximize work and minimize losses (viscous resistance as well as heat flows) is to consider the thermal efficiency η, which can be defined as the ratio of the work output over the sum of the work output and losses. Thus we can consider

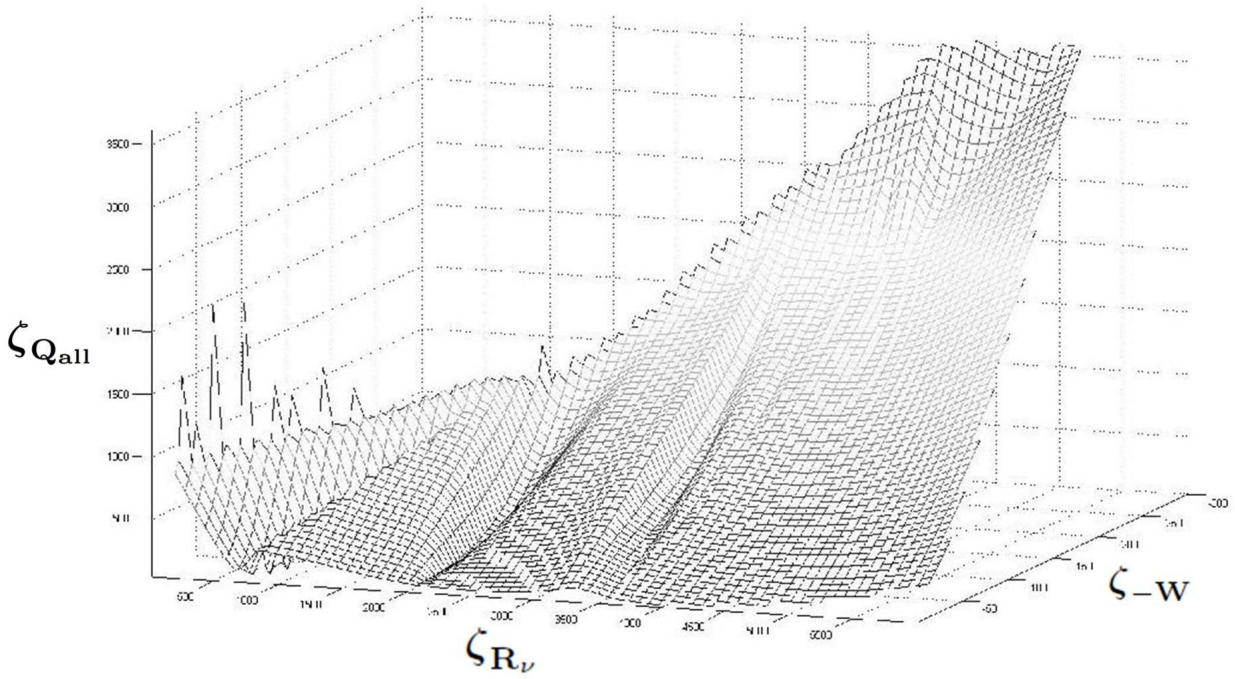


Figure 4.7: Top profile of efficient frontier: simultaneously minimizing $-W$, R_ν, and Q_{all}

the following optimization problem:

$$\max \ \frac{W}{W + \tilde{w}_2 R_v + \tilde{w}_3 Q_{conv} + \tilde{w}_4 Q_{rad} + \tilde{w}_5 Q_{cond}}, \tag{4.63}$$

subject to the original constraints of (MPF). This results in a mixed integer fractional programming problem, the numerator of which represents the work output, and the denominator being a sum of the work and combined (viscous and thermal) losses.

One way to solve fractional programs is via Dinkelbach's algorithm [36]. Briefly, Dinkelbach's algorithm eliminates the ratio in objective (4.63) by instead considering a sequence of problems that parameterize (4.63) with:

$$\eta = \frac{W}{W + \tilde{w}_2 R_v + \tilde{w}_3 Q_{conv} + \tilde{w}_4 Q_{rad} + \tilde{w}_5 Q_{cond}}, \tag{4.64}$$

and replace objective function (4.63) by:

$$\max \ W - \eta(W + \tilde{w}_2 R_v + \tilde{w}_3 Q_{conv} + \tilde{w}_4 Q_{rad} + \tilde{w}_5 Q_{cond}). \tag{4.65}$$

Dinkelbach's algorithm optimizes (4.65) subject to the original (MPF) constraints, iteratively updating its choice of η in order to identify η^* for which the maximum value of (4.65) equals zero. The sequence of choices for η finitely converge to η^*, solving the alternative representation and thus the original problem as well. Note the equivalence between the version of (MPF) as described in Section 4.5.3, and that of a single instance of (4.65) (corresponding to a fixed value of η) subject to the constraints in (MPF). Therefore solving (4.63) can be reduced to iteratively applying our procedure until the maximum of (4.65) attains zero.

4.6 CONCLUDING REMARKS

We demonstrate how optimization can improve the structural design of thermoacoustic devices. Whereas previous efforts have largely relied upon parametric studies, our approach simultaneously considers multiple variables over a set of constraints, and we includes multiple objective components in our objective function to quantify both acoustic and thermal performance. Through the objective function weights, a significant amount of personal preference is available to place desired emphasis.

We analyze cases of single objective components (two acoustic and three thermal), as well as two cases of multiobjective optimization. For the single objective cases, we analytically identify globally optimal solutions, while for the cases of multiple objectives, we generate the efficient frontier of optimal solutions for two combinations of objective weights. For both cases (the single objective as well as multiple objective approach), we show that there are nontrivial solutions to each design that have the potential to improve the energetic performance of thermoacoustic devices.

Our goal in optimizing sustainable energy devices was the identification of optimal structural variable levels so as to make the design more competitive with incumbent technology. We hope that some of the insights gained into optimal TAE designs will serve to benefit TAR technology and help supplant current refrigeration methods, thereby eliminating the need for toxic refrigerants and providing clear benefits with respect to sustainability.

5.0 CONCLUSIONS

In this dissertation we use optimization in conjunction with context-relevant algorithmic design to identify optimal underlying structures in three distinct application areas. In each study we formulate nonlinear mixed-integer programs that appropriately characterize the physical and/or structural attributes of the problem. We then develop an appropriate solution approach for each nonlinear mixed-integer program, including reformulation to linear mixed-integer programs (which off-the-shelf optimization software such as CPLEX can typically solve), algorithmic design that iteratively solves relaxations or restrictions to the problem, and analytical solutions.

Chapter 2 discusses two applications of optimization-based techniques to the field of data mining. For two specific biclustering data mining tasks, we successfully develop mathematical programs to characterize these tasks, and provide algorithms to identify meaningful biclusters in both synthetic and real biological data. Our results have significance wherever these two specific patterns may be of interest, for example in the diagnosis of disease conditions that link a subset of features to a subset of samples.

Chapter 3 presents an application of optimization in the context of computational biology. We develop a mathematical programming formulation and several subsequent linearizations that transform the problem into an equivalent mixed-integer program. Computational testing demonstrates that our approach quickly finds the optimal assignment, enabling a highly accurate prediction of protein/DNA structure using a minimal set of parameters. Our findings may have significance in gene therapy, for instance in modifying genes that produce malfunctioning proteins, or in the design of proteins that can target specific diseases.

Chapter 4 discusses an application to determine the optimal structure of a sustainable energy device, namely, the thermoacoustic engine that serves as a main driver for the ther-

moacoustic refrigerator. Moving beyond known approaches from the literature, we consider a nonlinear integer programming model with five structural variables and multiple objectives aimed at improving the engine's efficiency. We proceed to analyze the formulation and discuss optimal solutions for each of five individual objectives. We then consider the simultaneous optimization of multiple objectives. For the case of two acoustic objectives, as well as for the overall case combining all objective components, we present an efficient frontier of Pareto optimal solutions. The insight gained into optimal structural designs of TAEs can potentially contribute to enhancing the overall efficiency of the TAR to better compete with incumbent refrigeration technology.

BIBLIOGRAPHY

[1] R. Agrawal, J.E. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, Seattle, Washington, June 1998. ACM New York, NY.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[3] E. Althaus, A. Caprara, H.P. Lenhof, and K. Reinert. A branch-and-cut algorithm for multiple sequence alignment. *Mathematical Programming*, 105(2):387–425, 2006.

[4] E. Althaus, G. Klau, O. Kohlbacher, H. Lehhof, and K. Reinert. *Integer Linear Programming in Computational Biology*, pages 199–218. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, New York, 2009.

[5] K.H. Bae, Y. Do Kwon, H.C. Shin, M.S. Hwang, E.H. Ryu, K.S. Park, H.Y. Yang, D.K. Lee, Y. Lee, J. Park, H.S. Kwon, H.W. Kim, B.I. Yeh, H.W. Lee, S.W. Sohn, J. Yoon, W. Seol, and J.S. Kim. Human zinc fingers as building blocks in the construction of artificial transcription factors. *Nature Biotechnology*, 21(3):275–280, 2003.

[6] B. Balasundaram, S. Butenko, and S. Trukhanov. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10(1):23–39, 2005.

[7] M. Baz, B. Hunsaker, P. Brooks, and A. Gosavi. Automated tuning of optimization software parameters. Technical report, University of Pittsburgh Department of Industrial Engineering, 2007.

[8] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. In *Proceedings of the 6th Annual International Conference on Computational Biology (RECOMB '02)*, pages 49–57. ACM, 2002.

[9] A. Ben-Dor, B. Chor, R. Karp, and Z. Yakhini. Discovering local structure in gene expression data: The order-preserving submatrix problem. *Journal of Computational Biology*, 10(3-4):373–384, 2003.

[10] J.F. Benders. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[11] E. Besnoin. *Numerical Study of Thermoacoustic Heat Exchangers*. PhD thesis, Johns Hopkins University, 2001.

[12] P.S. Bradley and U.M. Fayyad. Refining initial points for k-means clustering. In J. Shavlik, editor, *Proceedings of the* 15^{th} *International Conference on Machine Learning (ICML'98)*, pages 91–99, San Francisco, CA, February 1998.

[13] P.S. Bradley and O.L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Proceedings of the* 15^{th} *International Conference on Machine Learning (ICML'98)*, pages 82–90, San Francisco, CA, February 1998.

[14] P.S. Bradley, O.L. Mangasarian, and W.N. Street. Clustering via concave minimization. In M.C. Mozer, M.I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9 (NIPS)*, pages 368–374, Denver, CO, December 1996. MIT Press.

[15] P.S. Bradley, O.L. Mangasarian, and W.N. Street. Feature selection via mathematical programming. *INFORMS Journal on Computing*, 10(2):209–217, 1998.

[16] P.S. Bradley, U.M. Fayyad, and O.L. Mangasarian. Mathematical programming for data mining: Formulations and challenges. *INFORMS Journal on Computing*, 11(3): 217–238, 1999.

[17] R. Brause, T. Langsdorf, and M. Hepp. Neural data mining for credit card fraud detection. In *Proceedings of the* 11^{th} *IEEE International Conference on Tools with Artificial Intelligence*, pages 103–106, Chicago, Illinois, November 1999. IEEE.

[18] S. Busygin, G. Jacobsen, and E. Krämer. Double conjugated clustering applied to leukemia microarray data. In *Workshop on Clustering High Dimensional Data and Its Applications (2nd SIAM SDM)*, Arlington, VA, April 2002.

[19] S. Busygin, O.A. Prokopyev, and P.M. Pardalos. Feature selection for consistent biclustering via fractional 0–1 programming. *Journal of Combinatorial Optimization*, 10 (1):7–21, 2005.

[20] S. Busygin, O.A. Prokopyev, and P.M. Pardalos. Biclustering in data mining. *Computers and Operations Research*, 35(9):2964–2987, 2008.

[21] T.D. Camenisch, M.H. Brilliant, and D.J. Segal. Critical parameters for genome editing using zinc finger nucleases. *Mini Reviews in Medicinal Chemistry*, 8(7):669–676, 2008.

[22] A. Caprara, G. Lancia, and S.K. Ng. Sorting permutations by reversals through branch-and-price. *INFORMS Journal on Computing*, 13(3):224, 2001.

[23] M.C. Chen. Ranking discovered rules from data mining with multiple criteria by data envelopment analysis. *Expert Systems with Applications*, 33(4):1110–1116, 2007.

[24] Y. Cheng and G.M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 93–103. AAAI Press, 2000.

[25] K.W. Cheung, J.T. Kwok, M.H. Law, and K.C. Tsui. Mining customer product ratings for personalized marketing. *Decision Support Systems*, 35(2):231–243, 2003.

[26] L. Cheung, D.W. Cheung, B. Kao, and K.Y. Yip. On mining micro-array data by order-preserving submatrix. *International Journal of Bioinformatics Research and Applications*, 3(1):42–64, 2007.

[27] H. Cho, I. Dhillon, Y. Guan, and S. Sra. Minimum sum-squared residue co-clustering of gene expression data. In *Proceedings of the 4th SIAM International Conference on Data Mining (SDM)*, pages 114–125, Lake Buena Vista, Florida, April 2004.

[28] C. Cifarelli, L. Nieddu, O. Seref, and P.M. Pardalos. K-T.R.A.C.E.: A kernel k-means procedure for classification. *Computers & Operations Research*, 34(10):3154–3161, 2007.

[29] *COMSOL Multiphysics Users Guide*. COMSOL AB, Burlington, MA, USA, 2005.

[30] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(13):273–297, 1995.

[31] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[32] I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 269–274, San Francisco, CA, August 2001. ACM New York, NY.

[33] I. Dhillon. Co-clustering software. http://www.cs.utexas.edu/users/dml/Software/cocluster.html, 2011.

[34] I.S. Dhillon and Y. Guan. Information theoretic clustering of sparse co-occurrence data. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM)*, pages 517–520, Melbourne, FL, November 2003. IEEE.

[35] I.S. Dhillon, S. Mallela, and D.S. Modha. Information-theoretic co-clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 89–98, Washington, DC, August 2003. ACM.

[36] W. Dinkelbach. On non-linear fractional programming. *Management Science*, 13(7): 492–498, 1967.

[37] R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer Verlag, New York, 1999.

[38] M. Elrod-Erickson, M.A. Rould, L. Nekludova, and C.O. Pabo. Zif268 protein-DNA complex refined at 1.6 Å : A model system for understanding zinc finger-DNA interactions. *Structure*, 4(10):1171–1180, 1996.

[39] V. Figueiredo, F. Rodrigues, Z. Vale, and J.B. Gouveia. An electric energy consumer characterization framework based on data mining techniques. *IEEE Transactions on Power Systems*, 20(2):596–602, 2005.

[40] D.E. Finkel. *DIRECT Optimization Algorithm User Guide*. Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC, March 2003.

[41] E. Friedman. Squares in circles. http://www2.stetson.edu/~efriedma/squincir, 2011.

[42] B. Gao, O. Griffith, M. Ester, and S. Jones. Discovering significant OPSM subspace clusters in massive gene expression data. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 922–928, Philadelphia, PA, August 2006. ACM New York, NY.

[43] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman & Company, 1979.

[44] F. Glover and G.A. Kochenberger. *Handbook of Metaheuristics*. Springer Verlag, 2003.

[45] H. Greenberg, W. Hart, and G. Lancia. Opportunities for combinatorial optimization in computational biology. *INFORMS Journal on Computing*, 16(3):211–231, 2004.

[46] O. Grodzevich and O. Romanko. Normalization and other topics in multiobjective optimization. In *Proceedings of the First Fields-MITACS Industrial Problems Workshop*, pages 89–102. The Fields Institute, 2006.

[47] D. Gusfield. *Haplotype Inference by Pure Parsimony*, pages 144–155. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, New York, 2003.

[48] J.A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67:123–129, 1972.

[49] I. Hedenfalk, D. Duggan, Y. Chen, M. Radmacher, M. Bittner, R. Simon, P. Meltzer, B. Gusterson, M. Esteller, M. Raffeld, Z. Yahkini, A. Ben-Dor, E. Dougherty, J. Kononen, L. Bubendorf, W. Fehrle, S. Pittaluga, S. Gruvberger, N. Loman, O. Johannsson, H. Olsson, B. Wilfond, G. Sauter, O. Kallioniemi, A. Borg, and J. Trent. Gene-expression profiles in hereditary breast cancer. *New England Journal of Medicine*, 344(8):539–548, 2001.

[50] C. Herman and Z. Travnicek. Cool sound: The future of refrigeration? Thermodynamic and heat transfer issues in thermoacoustic refrigeration. *Heat and Mass Transfer*, 42: 492–500, 2006.

[51] D. Hochbaum and A. Levin. Approximation algorithms for a minimization variant of the order preserving submatrices and for biclustering problems. Technical report, University of California, Berkeley, 2009.

[52] L.L. Hsiao, F. Dangond, T. Yoshida, R. Hong, R.V. Jensen, J. Misra, W. Dillon, K.F. Lee, K.E. Clark, P. Haverty, Z. Weng, G.L. Mutter, M.P. Frosch, M.E. MacDonald, E.L. Milford, C.P. Crum, R. Bueno, R.E. Pratt, M. Mahadevappa, J.A. Warrington, G. Stephanopoulos, G. Stephanopoulos, and S.R. Gullans. A compendium of gene expression in normal human tissues. *Physiological Genomics*, 7(2):97–104, 2001.

[53] C.L. Huang, M.C. Chen, and C.J. Wang. Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications*, 33(4):847–856, 2007.

[54] Z. Huang, H. Chen, C.J. Hsu, W.H. Chen, and S. Wu. Credit rating analysis with support vector machines and neural networks: A market comparative study. *Decision Support Systems*, 37(4):543–558, 2004.

[55] HumanGene Expression Index. HuGE Index.org Website. http://www.hugeindex.org, 2011.

[56] ILOG. *CPLEX 9.0 User's Manual*. Incline Village, NV, 2003.

[57] ILOG. *CPLEX 11.0 User's Manual*. Incline Village, NV, 2007.

[58] National Human Genome Research Institute. DNA microarray fact sheet. http://www.genome.gov/page.cfm?pageID=10000533#2, 2011.

[59] N. Kagawa. *Regenerative Thermal Machines*. International Institute for Refrigeration, Paris, 2000.

[60] J.D. Kececioglu, H.P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, and M. Vingron. A polyhedral approach to sequence alignment problems. *Discrete Applied Mathematics*, 104(1-3):143–186, 2000.

[61] Y. Kluger, R. Basri, J. Chang, and M. Gerstein. Spectral biclustering of microarray data: Coclustering genes and conditions. *Genome Research*, 13(4):703–716, 2003.

[62] G. Lancia. Mathematical programming in computational biology: An annotated bibliography. *Algorithms*, 1(2):100–129, 2008.

[63] L. Lazzeroni and A. Owen. Plaid models for gene expression data. *Statistica Sinica*, 12(1):61–86, 2002.

[64] J. Li. On optimal rule discovery. *IEEE Transactions on Knowledge and Data Engineering*, 18:460–471, 2006.

[65] J. Liu and G.D. Stormo. Quantitative analysis of EGR proteins binding to DNA: Assessing additivity in both the binding site and the protein. *BMC Bioinformatics*, 6 (1):176, 2005.

[66] S. Madeira and A. Oliveira. Biclustering algorithms for biological data analysis: A survey. *IEEE Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004.

[67] R.K. Martin. *Large Scale Linear and Integer Optimization: A Unified Approach*. Kluwer Academic Publishers, 1999.

[68] *MATLAB User's Guide*. The MathWorks, Inc., 2007.

[69] C.N. Meneses, Z. Lu, C.A.S. Oliveira, and P.M. Pardalos. Optimal solutions for the closest-string problem via integer programming. *INFORMS Journal on Computing*, 16 (4):419–429, 2004.

[70] K. Miettinen. *Nonlinear Multiobjective Optimization*. Springer, 1999.

[71] B. Minner, J. Braun, and L. Mongeau. Theoretical evaluation of the optimal performance of a thermoacoustic refrigerator. In *ASHRAE Transactions: Symposia*, volume 103, pages 873–887, 1997.

[72] D. Montgomery, E. Peck, and G. Vining. *Introduction to Linear Regression Analysis*. Wiley-Interscience, Hoboken, NJ, 2006.

[73] A.W. Moore and D. Zuev. Internet traffic classification using Bayesian analysis techniques. *ACM SIGMETRICS Performance Evaluation Review*, 33(1):50–60, 2005.

[74] A.T. Murray and R.L. Church. Constructing and selecting adjacency constraints. *INFOR*, 34(3):232–248, 1996.

[75] A.T. Murray and R.L. Church. Facets for node packing. *European Journal of Operational Research*, 101(3):598–608, 1997.

[76] United Nations. Our common future, Chapter 2: Towards sustainable development. http://www.un-documents.net/ocf-02.htm, 2011.

[77] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[78] S. Olafsson, X. Li, and S. Wu. Operations research and data mining. *European Journal of Operational Research*, 187(3):1429–1448, 2008.

[79] M.W. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5(1):199–215, 1973.

[80] C.D. Perttunen, D.R. Jones, and B.E. Stuckman. Lipschitzian optimization without the Lipschitz constant. *Journal of Optimization Theory and Application*, 79(1):157–181, 1993.

[81] C. Phua, V. Lee, K. Smith, and R. Gayler. A comprehensive survey of data mining-based fraud detection research. *arXiv:1009.6119v1*, 2005.

[82] M.E. Poese, R.W.M. Smith, S.L. Garrett, R. van Gerwen, and P. Gosselin. Thermoacoustic refrigeration for ice cream sales. In *Proceedings of* 6^{th} *IIR Gustav Lorentzen Conference*, 2004.

[83] O.A. Prokopyev, H.X. Huang, and P.M. Pardalos. On complexity of unconstrained hyperbolic 0–1 programming problems. *Operations Research Letters*, 33(3):312–318, 2005.

[84] O.A. Prokopyev, C. Meneses, C.A.S. Oliveira, and P.M. Pardalos. On multiple-ratio hyperbolic 0–1 programming problems. *Pacific Journal of Optimization*, 1(2):327–345, 2005.

[85] M.R. Rao. Cluster analysis and mathematical programming. *Journal of the American Statistical Association*, 66(335):622–626, 1971.

[86] C. Schensted. Longest increasing and decreasing subsequences. *Canadian Journal of Mathematics*, 13(2):179–191, 1961.

[87] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys CSUR*, 34(1):1–47, 2002.

[88] D.J. Segal, B. Dreier, R.R. Beerli, and C.F. Barbas. Toward controlling gene expression at will: Selection and design of zinc finger domains recognizing each of the 5'–GNN–3' DNA target sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 96(6):2758–2763, 1999.

[89] O. Seref, O.E. Kundakcioglu, O.A. Prokopyev, and P.M. Pardalos. Selective support vector machines. *Journal of Combinatorial Optimization*, 17(1):3–20, 2009.

[90] M. Sforna. Data mining in a power company customer database. *Electric Power Systems Research*, 55(3):201–209, 2000.

[91] M.J. Shaw, C. Subramaniam, G.W. Tan, and M.E. Welge. Knowledge management and data mining for marketing. *Decision Support Systems*, 31(1):127–137, 2001.

[92] H.D. Sherali and W.P. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Kluwer Academic Publishers, 1998.

[93] G. Swift. *Thermoacoustics: A Unifying Perspective for Some Engines and Refrigerators*. Acoustical Society of America, Melville NY, 2002.

[94] M.P. Tan, J.R. Broach, and C.A. Floudas. A novel clustering approach and prediction of optimal number of clusters: Global optimum search with enhanced positioning. *Journal of Global Optimization*, 39(3):323–346, 2007.

[95] M. Tawarmalani and N. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99: 563–591, 2004.

[96] J.R. Tebboth. *A Computational Study of Dantzig-Wolfe Decomposition*. PhD thesis, University of Buckingham, 2001.

[97] N. Temiz. Personal Communication, 2010.

[98] N.A. Temiz and C.J. Camacho. Experimentally based contact energies decode interactions responsible for proteindna affinity and the role of molecular waters at the binding interface. *Nucleic Acids Research*, 37(12):4076–4088, 2009.

[99] N.A. Temiz, A. Trapp, O.A. Prokopyev, and C.J. Camacho. Optimization of minimum set of protein-DNA interactions: A quasi-exact solution with minimum over-fitting. *Bioinformatics*, 26:319–325, 2010.

[100] M. Tijani, J. Zeegers, and A. de Waele. Design of thermoacoustic refrigerators. *Cryogenics*, 42:49–57, 2002.

[101] A. Trapp, O.A. Prokopyev, and S. Busygin. Finding checkerboard patterns via fractional 0–1 programming. *Journal of Combinatorial Optimization*, 20(1):1–26, 2010.

[102] A.C. Trapp and O.A. Prokopyev. Solving the opsm problem via integer programming. *INFORMS Journal on Computing*, 22(3):387–400, 2010.

[103] Y. Ueda, T. Biwa, U. Mizutani, and T. Yazaki. Experimental studies of a thermoacoustic stirling prime mover and its application to a cooler. *Journal of the Acoustical Society of America*, 72(3):1134–1141, 2003.

[104] S. Vanapalli, M. Lewis, Z. Gan, and R. Radebaugh. 120 Hz pulse tube cryocooler for fast cooldown to 50 K. *Applied Physics Letters*, 90, 2007.

[105] V.V. Vazirani. *Approximation Algorithms*. Springer Verlag, 2001.

[106] H.D. Vinod. Integer programming and the theory of grouping. *Journal of the American Statistical Association*, 64(326):506–519, 1969.

[107] G.I. Webb and S. Zhang. K-Optimal rule discovery. *Data Mining and Knowledge Discovery*, 10(1):39–79, 2005.

[108] M. Wetzel. *Experimental Investigation of a Single Plate Thermoacoustic Refrigerators*. PhD thesis, Johns Hopkins University, 1998.

[109] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[110] T. Wu. A note on a global approach for general 0-1 fractional programming. *European Journal of Operational Research*, 101(1):220–223, 1997.

[111] W. Xie and N. Sahinidis. A reduction-based exact algorithm for the contact map overlap problem. *Journal of Computational Biology*, 14(5):637–654, 2007.

[112] J. Xu, M. Li, D. Kim, and Y. Xu. Raptor: Optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–118, 2003.

[113] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: Capturing subspace correlation in a large data set. In *Proceedings of the 18th International Conference on Data Engineering*, pages 517–528, San Jose, CA, February 2002. IEEE.

[114] F. Zink. Personal Communication, 2011.

[115] F. Zink, H. Waterer, R. Archer, and L. Schaefer. Geometric optimization of a thermoacoustic regenerator. *International Journal of Thermal Sciences*, 48(12):2309–2322, 2009.

[116] L. Zoontjens, C. Howard, A. Zander, and B. Cazzolato. Modelling and optimisation of acoustic inertance segments for thermoacoustic devices. In *Proceedings of ACOUSTICS 2006*, 2006.