Spring 2010

Calculation of sensor redundancy degree for linear sensor systems

Santhosh Govindaraj
University of Iowa

CALCULATION OF SENSOR REDUNDANCY DEGREE FOR LINEAR SENSOR SYSTEMS

by

Santhosh Govindaraj

A thesis submitted in partial fulfillment
of the requirements for the Master of
Science degree in Industrial Engineering
in the Graduate College of
The University of Iowa

May 2010

Thesis Supervisor: Assistant Professor Yong Chen

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

MASTER'S THESIS

_____

This is to certify that the Master's thesis of

Santhosh Govindaraj

has been approved by the Examining Committee for the thesis requirement for the Master of Science degree in Industrial Engineering at the May 2010 graduation.

Thesis Committee: _____

Yong Chen, Thesis Supervisor

_____

Andrew Kusiak

_____

Pavlo Krokhmal

To My Parents

ACKNOWLEDGEMENETS

I would like to express my gratitude to Dr. Yong Chen for being my thesis advisor. I thank him for his patience, guidance and the help he provided during the process of writing this thesis. I also like to thank Professor Andrew Kusiak and Professor Pavlo Krokhmal for serving on my examining committee.

ABSTRACT

The rapid developments in the sensor and its related technology have made automation possible in many processes in diverse fields. Also sensor-based fault diagnosis and quality improvements have been made possible. These tasks depend highly on the sensor network for the accurate measurements. The two major problems that affect the reliability of the sensor system/network are sensor failures and sensor anomalies. The usage of redundant sensors offers some tolerance against these two problems. Hence the redundancy analysis of the sensor system is essential in order to clearly know the robustness of the system against these two problems. The degree of sensor redundancy defined in this thesis is closely tied with the fault-tolerance of the sensor network and can be viewed as a parameter related to the effectiveness of the sensor system design.

In this thesis, an efficient algorithm to determine the degree of sensor redundancy for linear sensor systems is developed. First the redundancy structure is linked with the matroid structure, developed from the design matrix, using the matroid theory. The matroid problem equivalent to the degree of sensor redundancy is developed and the mathematical formulation for it is established. The solution is obtained by solving a series of ℓ^1-norm minimization problems. For many problems tested, the proposed algorithm is more efficient than other known alternatives such as basic exhaustive search and bound and decomposition method.

The proposed algorithm is tested on problem instances from the literature and wide range of simulated problems. The results show that the algorithm determines the degree of redundancy more accurately when the design matrix is dense than when it is sparse. The algorithm provided accurate results for most problems in relatively short computation times.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1

INTRODUCTION

There has been increase in the need for high quality information related to a process as it is not only required for effective control and evaluation of the process but also for fault diagnosis and in-process quality improvements. The collection of sensors that are used to obtain such information - measurements for a number of variables related to a process is called a sensor network. One type of such sensor networks called the linear sensor systems has been used in wide variety of applications and is the one concentrated in this work. The recent developments in sensor technology have enabled many processes in diverse fields to be automated. High reliability of sensor network is called for as they are used in applications where unavailability of the true measurements can jeopardize the safety, lead to high cost of damage/downtime, and which are hostile/ infeasible environments for human measurement.

Two major problems that degrade the reliability of the sensor network are sensor failures and sensor anomalies. Inclusion of redundant sensors is the most common way to deal with these two undesired effects. Redundancy of measurements, obtained from redundant sensors, increases the fault-tolerance capability of sensor networks. In particular, the analysis of sensor redundancy structure finds how many sensor failures the system can tolerate. The objective of this research is to develop an efficient algorithm to find the degree of sensor redundancy for a linear sensor system, which is defined in the following subsections.

1.1 Linear Sensor System

A sensor system is called a *linear sensor system* if the measurements of the sensors can be related to the variables of interest by a linear model. Consider a process in which p variables have to be determined/estimated using n sensors with $n > p$. The

following linear model can be used to link the measurements of the sensors and the true values of the variables,

$$\mathbf{y} = \mathbf{H}\,\mathbf{u} + \mathbf{e} \tag{1.1}$$

where $\mathbf{y} = [y_1, y_2,.. y_n]^T$ is a n-dimensional vector representing the measurements of n sensors, $\mathbf{u} = [u_1, u_2,.. u_p]^T$ is a p-dimensional vector representing the variables that are to be monitored, $\mathbf{H}$ is an n x p design matrix (also called measurement matrix) whose individual element's value depends upon the location of the sensor and its relationship with the variables, and $\mathbf{e} = [e_1, e_2,.. e_n]^T$ is a n-dimensional vector of measurement errors or noise in the sensor system.

Such linear models for the sensor system have been adopted and studied in various fields such as manufacturing processes (Khan et. al 1998, Jin and Shi 1999, Apley and Shi 2001), power plant (Dorr et al. 1997), chemical processes (Ali and Narasimhan 1993, Madron and Veverka 1992, Bagajewicz and Sanchez, 1999), and electrical power systems (Mili et al. 1990). More details of these models and applications are provided in Section 1.3.

1.2 Degree of Sensor Redundancy

The degree of sensor redundancy of a system (δ) is equal to the maximum number of *any* sensor failures a system can withstand and still provide enough measurements to uniquely identify all the variables. From the above definition it is clear that upon failure of any subset of sensors of size less than or equal to δ, all the variables are still observable.

The degree of sensor redundancy can be obtained from the measurement matrix **H**. Each row of **H** corresponds to a sensor in a linear sensor system and each column corresponds to a variable. An element of **H**, $\mathbf{H}_{ij}$, $i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, p\}$, is zero if the sensor corresponding to the i^{th} row has no relation with the variable corresponding

the j^{th} column. We assume r(**H**) = p so that all variables can be identified uniquely when there is no sensor failure. As mentioned earlier since each row of the design matrix **H** corresponds to a sensor, we can represent the failure of a sensor / group of sensors by removal of its corresponding row/rows from **H**. If the rank of **H** is reduced below p after removal of the rows corresponding to the failed sensors, then the variables cannot be determined uniquely and this condition is termed as loss of observability.

Let d denote the number of sensor failures (irrespective of which ones) and $\mathbf{H}_{(-d)}$ denotes the matrix after the removal of any d rows (sensors) from **H**. Then the degree of sensor redundancy can be defined formally as:

$$\text{Sensor Redundancy } \delta(\mathbf{H}) = \min \{d - 1\} \quad \text{s.t. } \text{rank}(\mathbf{H}_{(-d)}) < \text{r}(\mathbf{H}) \tag{1.2}$$

1.3 Literature Review

This section reviews the literature for the applications of the linear sensor systems, necessities for sensor redundancy analysis, and estimation procedures developed for the degree of sensor redundancy.

1.3.1 Linear Sensor System

Researchers have found that many processes in various engineering domains that can be modeled with the linear model as shown in (1.1). Some of these were mentioned in section 1.1. Here we shall see about them in more details.

Manufacturing process fault diagnosis is one of the major areas in which the linear sensor systems have been applied extensively. The usage of sensors here is to determine the fixture faults if any because such faults can produce significant dimensional variations in the final product obtained through a multi-stage assembly process. Considerable amount of work related to the development of fault diagnosis methods in complex multi-stage manufacturing process used the linear model. The linear

model (1.1) links the product quality measurements (**y**) with the process (fixture) faults (**u**) and **H** is determined from the sensor and tooling layout. Apley and Shi (2001) used the linear model in an auto body assembly process. The authors developed a statistical technique, based on principal component analysis and factor analysis, to identify any fixture faults if present using the information obtained from the linear sensor system. Zhou et al. (2003) formulated the fault-quality diagnostic model as a mixed linear model and developed a methodology to solve the diagnosis problem by solving its equivalent problem of variance component analysis. They show that the capability of diagnosis by the sensor system depends heavily on the design of the sensor system. Similar usage of linear models in assembly processes can be seen in the work of Ding et al. (2002).

Array Signal Processing is another field where linear model (1.1) is used. The problem here is to determine the arrival direction of a signal and is called Direction of Arrival Estimation (DOA). It uses a number of sensors organized in certain patterns, or arrays, to detect signals from narrowband sources in the far field. The narrowband represents a known narrow range of frequencies within which lies the source signals to be detected. The measurements in the sensors **y** can be linked with the source signals **u** through the linear model (1.1). The matrix **H** is called the array response and is determined by the adjustments in the sensors directions while shifting towards the source signals. This technique has been used in several applications including in radars, sonar and mobile communications (Sidiropoulos et al. 2000).

Calibration of Wireless Sensor Networks: It is a process of estimating the so called calibration parameters in a wireless sensors in an ad-hoc networking (Cho et al. 2009b). An ad-hoc network does not have a fixed network topology but one that usually changes frequently. The knowledge of the locations of the sensors is required in most of the cases to understand the process under observation and to make further decisions if necessary. Since installing a global positioning system (GPS) one with each sensor is not practical owing to the high power consumption and cost associated, only a subset of

sensors called the anchor nodes are installed with the GPS. The locations of the remaining sensors called the non-anchor nodes are determined relative to the anchor nodes. This is achieved by first measuring the distances between the non-anchor nodes and the anchor nodes and then determining the location of non-anchor nodes using some geometric principle. The distances between two sensors can be calculated using the knowledge of time difference of arrival (TDOA) between the radio frequency (RF) and an acoustic signal. But these two signals were observed to be affected by metal and temperature & moisture respectively resulting in inaccurate measurements. To overcome this problem, first in off-line setting the original (true) inter node distances is measured and then for the same setting the TDOA is measured. A mathematical (linear) model, relating these two is then developed by determining the so called calibration parameters. The matrix **H** in (1.1) corresponds to the TDOA between all nodes, **y** corresponds to the true inter node distances and the **u** corresponds to the calibration parameters. To find the inter nodes distances (**H**) and hence the location of the sensors during the operation, the calibration parameters (**u**) and the true distances (**y**) are used.

Optimal Design of the sensor network is another major topic in which considerable amount of work has been carried out by the researchers. The objectives of the designing process include determining the location of the placement of sensors, number of sensors to be used for obtaining the desired level of reliability. The general list of constraints could include maximum cost of installation and operation, technical feasibility, space limitation, lack of proper access for calibration and complexity (Ali and Narasimhan, 1995). The very initial works were related to identifying the subset of sensors required to guarantee the observability of all the desired variables using graph theory. Later researchers developed a number of efficient optimal designing methods for variety of objectives and meeting variety of constraints. Related works can be seen in Bagajewicz and Sanchez (1999), Khan et al. (1998), Madron and Veverka (1992).

1.3.2 Redundancy Analysis

This section reviews the literature on works related to the analysis of redundancy structure of the sensor system and computation procedures for the degree of sensor redundancy. We focus in particular the linear sensor systems.

The traditional methods of ensuring the reliability of a sensor by preventive maintenance activities like off-line gauge repeatability and reproducibility calibration is often time consuming and expensive (Dorr et al.1997). Also the uncalled for maintenance as part of repetitive preventive maintenance could lead to material degradation and this could cause monetary wastage in cases where expensive sensors are used (Dorr et al. 1997).

Staroswiecki et al. (2004) considered a system in which maintenance operations cannot take place to restore any sensor failures and analyzed the redundancy structure of the system to find which subsets of sensors when failed will cause system level failure. System level failure corresponds to the state of the system when not all variables can be estimated uniquely. Their procedure to calculate the degree of sensor redundancy is equivalent to testing the rank of **H** in (1.2) after removal of d rows corresponding to the failed sensors beginning with the value of $d = 1$. This procedure is very similar to the basic exhaustive search explained in section 2.2. The authors also developed an algorithm for designing fault tolerant sensor network with the required reliability and redundancy. The computation time of the exhaustive search method grows exponentially with the number of sensors. So it is only applicable for small problems.

Cho et al. (2004, 2007) developed a more efficient algorithm based on branch and decomposition to determine the degree of sensor redundancy for problems with special sparse and clustered structures. They used the matroid theory, a major mathematical tool from the area of combinatorics, to analyze the redundancy structure of the sensor network system. Their algorithm makes use of the sparse nature of **H** in some engineering applications. By certain decomposition technique the exhaustive search is carried out on a

number of smaller size matrices after satisfying certain conditions. However when **H** does not have a sparse structure the branch and decomposition algorithm is not efficient. More details of this algorithm are explained in section 2.2.

Apart from the applications mentioned above the determination of the redundancy degree is also discussed in robust statistics (Cho et al. 2009a). Consider the linear model presented in (1.1). Classical method of estimating **u** via Least Squares depends on strict assumption that error term **e** is normally distributed. When this violated it is difficult to get accurate estimation for **u**. The accuracy also degrades when **y** has outliers. The main aim of robust statistics is to produce the estimators **u** that are unaffected by the above circumstances. Breakdown point is one of the important measures used to quantify the robustness of the estimator and is defined as the fraction of incorrect observation that could result in inaccurate estimation of **u**. The higher the breakdown point of an estimator, the more robust it is. Least Trimmed Squares is one of the methods in robust statistics to determine the maximum breakdown point. It involves the estimation of a variable called trimming parameter and the δ defined in (1.2) is used for its calculation.

1.4 Thesis Outline

In this thesis, we will develop a novel approximate algorithm to evaluate the degree of sensor redundancy in a linear sensor system. Our algorithm is much more efficient than the existing methods, especially for problems without sparse and clustered structures. Beyond this introduction, Chapter 2 of this thesis provides background information on matroid theory with standard definitions and axioms sufficient enough to understand our method. Chapter 3 explains the characterization of the redundancy degree problem as a matroid problem. The mathematical formulation for this problem and subsequent determination of its solution using ℓ^0-, ℓ^1- norm minimization techniques is then presented. The chapter then concludes with presentation of the entire proposed algorithm. Chapter 4 gives details of several data sets used to test the performance of the

algorithm and is followed by the discussion of the results. Finally Chapter 5 provides conclusion and suggestions for further research.

CHAPTER 2

BACKGROUND AND FOUNDATION TECHNIQUES

As mentioned in the earlier chapter, we characterize our problem of determining the degree of sensor redundancy using the matroid structure linked with the measurement matrix of the sensor system. While chapter three explains in detail the exact relation between the degree of sensor redundancy and a concept of matroid theory called (the smallest) cocircuit, this chapter provides the definition of a matroid, major axioms and properties associated with the matroid sufficient to envisage our objective as a matroid problem. Second section of this chapter reviews the four existing algorithms capable of identifying the smallest cocircuit, which is a problem directly related to finding the redundancy degree of a linear sensor system.

2.1 Matroid

The term 'matroid' was introduced by Whitney in 1935 to describe a system with an abstract linear dependence relation. A matroid does not have a single unique definition as it can be expressed in many different but equivalent ways. It captures the generic and fundamental properties of independence of a large class of useful structures such as matrices and graphs which share some common properties (Chen 2006). A matroid M is generally denoted by a pair $(E, \mathcal{I})$ where E is called the *ground set* and $\mathcal{I}$ is a collection of *independent subsets* of E, and satisfies the *independence augmentation axiom*. This definition is explained with an example in the next section.

There had been development of a large class of matroids ever since the introduction of two basic matroids: *vectorial matroid* and *graphic matroid* - matroids defined over matrix and graph respectively. Since our work consists of application of a matroid over the (measurement) matrix, unless noted otherwise, in this thesis we do not differentiate a matroid and a vectorial matroid.

A brief description about the graphical matroid and some applications of the matroid theory are provided in the subsequent sections.

2.2 Vectorial Matroid

The definition of a matroid and its axioms are explained in detail with the help of a vectorial matroid, but they are *applicable to any other matroid*. To understand the vectorial matroid, consider the below p x n matrix with (p=3, n=5),

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Let us denote the columns of **A** from left to right by 1 through 5 and represent it by $E = \{1, 2, 3, 4, 5\}$. Any subset of E is said to be linearly *independent* if none of its elements can be written as a linear combination of other elements in the same subset. For our example matrix **A**, the collection of independent sets represented by $\mathscr{I}$' are as follows,

$\mathscr{I}$ = {{Ø}, {1}, {2}, {3}, {4}, {5}, {1, 2}, {1, 3}, {1, 4}, {1, 5}, {2, 3}, {2, 4}, {2, 5}, {3, 4}, {3, 5}, {4, 5}, {1, 2, 4}, {1, 2, 5}, {1, 3, 4}, {1, 3, 5}, {1, 4, 5}, {2, 3, 4}, {2, 3, 5}, {3, 4, 5}}

Whitney (1935) mentioned that any subset of columns of a matrix is either linearly independent or linearly dependent and the following two theorems must hold:

(a) Any subset of an independent set is independent.

(b) If N_p and N_{p+1} are independent sets of p and $p + 1$ columns respectively, then N_p together with some column of N_{p+1} forms an independent set of $p+1$ columns.

Many systems other than matrices also consist of sets satisfying the above two conditions. Hence we can treat these two conditions as axioms and define any system

obeying (a) and (b) as a *matroid*. Let us denote the matroid defined on a matrix **A** by M(**A**).

As mentioned earlier there are several equivalent ways of defining a matroid and one of which is presented below. In terms of independent sets: the pair (E, $\mathscr{I}$), where E is a nonempty finite ground set, and $\mathscr{I}$ is nonempty collection of independent subsets of E, is called a matroid M upon satisfying the following properties,

i) An empty set is an independent set

ii) Any subset of an independent set is independent

iii) If I_1 and I_2 are two independent sets with $I_1 \subseteq I_2$, then there exists an element e contained in I_2 that is not in I_1 such that $I_1 \cup \{e\}$ is independent.

The last property (iii) is called the independence augmentation axiom.

2.3 Base, Circuit, Rank of a Matroid

An independent subset is called a maximal independent set if an addition of one new element to it will make it a dependent set. Such a set is called as *Base* under matroid theory terminology and all of such maximal independent sets have the same number of elements. The rank of a matroid is defined as the cardinality of any base. For a vectorial matroid defined on a matrix **A**, the rank of the matroid is always equal to the rank of the matrix, denoted by r(**A**), which is the maximum number linearly independent rows or columns of **A**, both of which are always equal for a given matrix. Representing the collection of such bases by B, for our example matrix,

B = {{1, 2, 4}, {1, 2, 5}, {1, 3, 4}, {1, 3, 5}, {1, 4, 5}, {2, 3, 4}, {2, 3, 5}, {3, 4, 5}}

Any set containing the base is called a *spanning set*. From the property (ii) mentioned earlier it is clear that the independent sets $\mathscr{I}$ is a collection of all subsets of each of the bases. In terms of bases, we can also define a matroid M as a pair (E, B),

where E is a nonempty finite ground set, and B is nonempty collection of bases, upon satisfying the following properties (Nieto and Marcin 2000),

i) No base properly contains another base

ii) If B_1 and B_2 are bases and for every element e present in B_1, there exists an element f in B_2 such that $(B_1 - \{e\}) \cup \{f\}$ is also a base.

The bases can be viewed as a collection of subsets of E with least number of columns whose rank is equivalent to the rank of the matrix.

All subsets of E not contained in independent sets ($\mathscr{I}$) are linearly dependent and are called as *dependent sets*. For the example matrix **A**, the dependent sets represented by $\mathscr{D}$ are as follows,

$\mathscr{D}$ = {{1, 2, 3}, {2, 4, 5}, {1, 2, 3, 4}, {1, 2, 3, 5}, {1, 2, 4, 5}, {1, 3, 4, 5}, {2, 3, 4, 5}, {1, 2, 3, 4, 5}}

A dependent set that, after removal of any one element, becomes an independent set is called a minimal dependent set. In matroid terminology such a set is called *circuit* of a matroid. Since all the proper subsets of a dependent set must be independent in order to be a circuit, *no proper subset of a circuit will be a circuit*. Representing by $\mathbb{C}$ the collection of circuits, for our example matroid M(**A**),

$\mathbb{C}$ = {{1, 2, 3}, {2, 4, 5}, {1, 3, 4, 5}}

As we can see from the example, the number of elements in the circuits could vary and need not be constant as with the case of bases. Another definition of matroid can be given terms of circuits: The pair (E, $\mathbb{C}$), where E is a nonempty finite ground set, and $\mathbb{C}$ is nonempty collection of circuits, is called a matroid upon satisfying the following conditions (Nieto and Marcin 2000),

i) No circuit properly contains another circuit

ii) If C_1 and C_2 are two different circuits sharing a common element c, then $(C_1 \cup C_2)$ will contain a circuit that does not have c.

The cardinality or size of the smallest circuits is called the *girth* (*g*) of the matroid and for the example matrix it is the size of the first two circuits, which is 3.

2.4 Dual Matroid

Every finite matroid has a dual matroid and this dual is unique. This is generally denoted by M*. Similar to the original matroid, the dual matroid also has got its own collection of independent sets, bases, dependent sets and circuits. Even though the names and relationships among these terms of a dual matroid are similar to those of the original matroid, they do not infer the same meaning over the matrix as the terms of original matroid does. This can be better understood with the example matroid M(**A**) specified over **A** introduced earlier.

The dual matroid also has got the same number of bases as that of the original matroid and are called cobases. A *cobase* is obtained by taking the compliment of the base of the original matroid, for example if *b* is one of the bases of M, then $b^* = \{E - b^*\}$ is its corresponding cobase (and is a base of M*). The dual matroid is generally defined as a pair (*E*, B*), where *E* is a nonempty finite ground set, and B* is nonempty collection of cobases. From this definition it is clear that the dual of this dual matroid M* is the same as the original matroid M.

Similar to the independent sets of a matroid being a collection of all subsets of all of its bases, the independent sets of a dual matroid is the collection of all subsets of all its bases (or cobases of the original matroid). These are called *coindependent sets* of the original matroid and are represented by $\mathscr{I}^*$. All the other subsets of *E* are called *codependent sets* and represented by $\mathscr{D}^*$. For our example matrix **A** we can obtain the cobases and subsequently coindependent sets and codependent sets from the previously found bases of M as shown below,

B = {{1, 2, 4}, {1, 2, 5}, {1, 3, 4}, {1, 3, 5}, {1, 4, 5}, {2, 3, 4}, {2, 3, 5}, {3, 4, 5}}

B* = {{3, 5}, {3, 4}, {2, 5}, {2, 4}, {2, 3}, {1, 5}, {1, 4}, {1, 2}}

$\mathscr{I}$* = {{Ø}, {1}, {2}, {3}, {4}, {5}, {1, 2}, {1, 4}, {1, 5}, {2, 3}, {2, 4}, {2, 5}, {3, 4}, {3, 5}}

$\mathscr{D}$* = {{1, 3}, {4, 5}, {all subsets of size 3, 4, 5}}

Circuits of the dual matroid are called *cocircuits* of the original matroid. To understand these we shall follow the same relation defined earlier between circuits and dependent sets of a matroid arbitrarily without trying to find the intuitive meaning upon applying them on the matrix. Following the condition that circuits are the minimal dependent sets we get,

C* = {{1, 3}, {4, 5}, {1, 2, 4}, {1, 2, 5}, {2, 3, 4}, {2, 3, 5}}

Another easier way to generate this set is to first find the smallest circuits and then obtain the bigger ones. These can be explained by below two steps,

1) Circuits are the minimal dependent set: following this condition we can see that first two sets of $\mathscr{D}$*, namely {1, 3} and {4, 5} (smallest size among the codependent sets) are the circuits of the dual matroid. Add them to C*.

2) No proper subset of a circuit will be a circuit: perform the following check on the dependent sets of next bigger size, if present (for our example - 3). Only the codependent set which do not contain any of the cocircuits present in C* as one of its proper subset is eligible to be a circuit of the dual matroid (or new cocircuit of the original matroid) and is added to C*. The process is continued until we check all the codependent sets of present size, after which we continue with the codependents sets of next bigger size and so on. Logical thinking can help us set the maximum size of the codependent sets to check for cocircuits as (n - r + 1).

The cardinality or size of the smallest cocircuits is called the ***cogirth*** (**g***) of the matroid and for the example matrix it is the size of the first two cocircuits which is 2.

2.5 Graphic Matroid

Similar to the vectorial matroid, these sets can also be explained for a graphic matroid - matroid defined over a graph. Let us consider a undirected graph **G** and its corresponding graphic matroid M(**G**). The edges connecting the nodes are used to define different sets in graphic matroid, analogous to the columns of matrix being used in vectorial matroid. The rank of a graph is equal to the number of edges present in the spanning tree, which is the minimal number of edges that connect all vertices is a base of M(**G**). The independent sets of M(**G**), subsets of bases, are the forests of the graph **G**. A circuit of M(**G**) is a collection of edges that form cycle in the graph. The dual of the graphic matroid is called the cographic matroid and is denoted by M*(**G**). The cocircuits of M(**G**) are the cut sets of **G**. Since our work deals with vectorial matroid, detailed explanation of the graphic matroid is not presented here but can be found in Conforti and Rao (1987), Ryan and Lee (1992).

2.6 Applications of Matroid Theory

Due to the matroid's characteristic to observe the linear independence/ dependence properties of various structures including matrices and graphs, researchers have been able to find many real time, useful applications whose mathematical models are similar/ related to the combinatorial problems found in different kind of matroids. Ryan and Lee (1992) in their work have done an extensive literature study about various algorithms then developed and applications related to matroid theory. In the general applications section, the authors mention problems under the categories of job sequencing, minimizing reliability / cost ratio, assignment, asymmetric traveling salesman, capturing approximate linear dependencies in multivariate data, scene analysis can be solved using matroid theory. Two sections, one each, are devoted to problems in the areas of electrical systems and statics.

We can find more applications of matroid theory in network related problems mainly because of the graphic structure. Some other popular matroids developed, similar to but different than graphic matroid, are matching, transversal, fano, gammoid. The details of these matroids can be found in Oxley (1992). Seymour (1977) introduced so called max-flow min-cut matroids and established the conditions for a matroid in order to be of this type. Truemper(1987) later developed algorithm to determine if a given matroid is max-flow min-cut matroid or not. Also he developed algorithms for determining the maximum flow and shortest routes which up on following his proof and work leads up to requirement of determining the shortest circuits and cocircuits of matroids meeting some specific conditions.

Cho, Chen and Ding (2004) used the matroid theory to find the degree of sensor redundancy in a linear sensor network. We shall see in chapter 5 that this problem can be solved by either finding the cogirth / girth (cardinality of the smallest cocircuit/ smallest circuit) of two different matroids obtained from the measurement matrix **H**. The next section presents some existing algorithms that can be used for finding the cogirth/ girth of a matroid.

2.7 Cocircuit & Circuit Algorithms

Later in chapter 3, we explain the relationship between sensor redundancy degree δ and the smallest cocircuit of a matroid constructed using the design matrix **H** introduced in (1.1). Since cocircuits of M(**A**) is equivalent to circuits of M*(**A**), finding circuits of the latter matroid is equally sufficient. In fact our algorithm also uses this technique and more details are presented in the next chapter. In this section brief descriptions of four existing algorithms that are capable of determining the smallest cocircuits / circuits are presented. For all the algorithms explained below let M be the matroid defined on a matrix **A** of p-rows and n-columns ($p < n$).

2.7.1 Exhaustive Search

This basic search method is used to find the size of the smallest cocircuit. Let the sub-matrices of **A** formed after removal of any *d* columns be represented by $\mathbf{A}_{(-d)}$.

Algorithm for Exhaustive Search

1. Let $d = 1$
2. If there is any sub-matrix such that $\text{rank}(\mathbf{A}_{(-d)}) < \text{r}(\mathbf{A})$, Cogirth of M = *d*, stop.
3. $d = d + 1$, go to step 2.

This method is practical only for very small matrices to have a reasonable computation time. A slight increase in the value of cogirth causes huge increase in the computation time as it would be proportional to $\sum_{d=1}^{g^*} {}_nC_d$. We shall see its inefficiency in the results in chapter 4. Hence there needs to be more sophisticated and intelligent way for determining the cogirth.

2.7.2 Circuit Enumeration

Boros et al. (2005) developed this technique to find all the circuits of a matroid. They showed that all the circuits can be enumerated in incremental polynomial time. Before describing this algorithm we shall recap the second property (explained in chapter 2) a pair (*E*, C) must obey in order to be a matroid - If C_1 and C_2 are distinct circuits of M with a common element $e \in C_1 \cap C_2$, then there exists another circuit $C_3 \subseteq C_1 \cup C_2 \setminus e$. This is also called as circuit axiom. The collection of circuits C is said to be closed only if the circuit axiom is satisfied.

Algorithm for Circuit Enumeration

1. Select a base B^0 of M and $x \in E \setminus B^0$.
2. Find the unique *fundamental circuit* of *x* in base B^0 denoted by $\text{C}(B^0, x)$, such that $x \in \text{C}(B^0, x) \subseteq B^0 \cup x$.
3. Repeat step 2 for each new $x \in E \setminus B^0$, Denote this collection by $\mathcal{F}(B^0) = \{\text{C}(B^0, x) \mid x \in E \setminus B^0\}$.

4. Set C = $\mathcal{F}(B^0)$
5. Check if C is closed using the circuit axiom; if yes go to step 7.
6. One new circuit ($C_3 \subseteq C_1 \cup C_2 \setminus e$) is generated for each violation to satisfy the axiom. All the new circuits are added to C. Go to step 5.
7. Stop; C is the family of all circuits of M.

Step 3 generates $n - r$ (M) fundamental circuits in the base B^0. Steps 5-6 in the above algorithm are iterative. Even though our requirement is to find the smallest circuits, we need to allow the algorithm to complete listing all the circuits to make sure we identify the smallest one accurately. Though this algorithm sometimes performs much better than the exhaustive search, the computation time is still high for matrices of large dimension and when there are large number of circuits.

2.7.3 Modified Circuit Enumeration

Chen (2006) mentioned that in step 5 of the circuit enumeration algorithm, while checking for closeness of C with respect to the circuit axiom, there is a waste of computation time in testing many pairs of circuits more than once. He offered a modified procedure to avoid repeated examination of a previously tested pair by systematic testing of the circuits in C for the violation of the circuit axiom.

The modified procedure is as follows. First three steps of the circuit enumeration algorithm are performed to generate the fundamental circuits $\mathcal{F}(B^0)$. If there are less than two circuits in $\mathcal{F}(B^0)$ they are the only circuits and the algorithm stops. Else only first two circuits in $\mathcal{F}(B^0)$ are added to C. The second circuit in C is tested for circuit axiom against the first and if required to satisfy the circuit axiom a new circuit is added at the end of C and list would now has three circuits. Let us say the process of testing a circuit (say primary circuit denoted by θ) in C for closeness of circuit axiom against each of the remaining circuits present before θ in the list as a single iteration testing. This procedure is repeated for each circuit that has been added to C following its order of inclusion and

continues until there are no more circuits in C to be selected as primary circuit. This can be considered as one cycle. Then the next fundamental circuit (if any in $\mathcal{F}(B^0)$) is added to C and the cycle repeats until all the fundamental circuits are added to C.

The planned testing procedure of the modified algorithm checks only the untested pairs leading to the minimum number of tests required. The procedure is explained well with a detailed flowchart in Chen (2006). Though this procedure has an obvious advantage of systematic testing leading to comparatively smaller computation time, still all the circuit pairs have to be determined in order not to miss the smallest cocircuit.

2.7.4 Bound and Decomposition

This algorithm is developed by Cho et al. (2007) to find the smallest cocircuit of a vectorial matroid and is more efficient when the matrix, over which the vectorial matroid is specified, is sparse meaning many of its entries are zeros. The strengths of this algorithm are in its exploitation of the underlying clustered structure in the design matrix, and utilization of the concept termed *connectivity* and its properties under the matroid theory. Brief outline of the algorithm: can be considered to have two steps, first step involves reshaping of the design matrix into Bordered Block Diagonal Form (BBDF) using the known methods developed in past and the second step identifies the smallest cocircuit by performing basic exhaustive search (*ES*) explained in section 2.7.1 but on smaller dimensional matrices.

The second step is directly related to connectivity of the matroid. In order to explain this we need to familiarize with another term called *restriction*. Let E be the ground set and I be the collection of independent sets and M = (E, I) be a matroid. Suppose there exists a set $X \subseteq E$ with collection of independent sets $I \subseteq X$ and $I \in I$, then (X, I | X) is a matroid called the restriction of M to X (Cho et al. 2007). It is denoted by M | X.

The following explanations for disconnected matroid are obtained from Oxley (1992). A matroid M is called disconnected if and only if, for some proper non-empty subset T of E(M),

$$\mathrm{I(M)} = \{I_1 \cup I_2 : I_1 \in I(\mathrm{M} \mid \mathrm{T}),\ I_2 \in I(\mathrm{M} \mid (E\text{-T}))\}$$

Generalizing the above condition for n matroids, if $M_1, M_2, \ldots, M_n$ are matroids on disjoint ground sets $E_1, E_2, \ldots, E_n$ with the respective collection of independent sets $I_1, I_2, \ldots, I_n$, then M(E, I) is a matroid. The individual matroids are called direct sum components of M. The matroid M is called the 1-sum or direct sum of the collection of individual matroids and is denoted by $M_1 \oplus M_2 \oplus \ldots \oplus M_n$.

The following properties are true for the disconnected matroids,

$$r(M_1 \oplus M_2 \oplus \ldots \oplus M_n) = r(M_1) + r(M_2) + \ldots r(M_n)$$

$$C(M_1 \oplus M_2 \oplus \ldots \oplus M_n) = C(M_1) \cup C(M_2) \cup \ldots \cup C(M_n)$$

$$(M_1 \oplus M_2 \oplus \ldots \oplus M_n)^* = M_1^* \oplus M_2^* \oplus \ldots \oplus M_n^*$$

$$C^*(M_1 \oplus M_2 \oplus \ldots \oplus M_n) = C^*(M_1) \cup C^*(M_2) \cup \ldots \cup C^*(M_n)$$

This algorithm directly makes use of the last property which states that cocircuits of the 1-sum matroid is union of all the cocircuits of smaller matroids. Thus finding the smallest cocircuit(s) of 1-sum matroid is equivalent to finding the smallest cocircuits of all the individual matroids and finally comparing them to choose the smallest.

The matrix structure of such a disconnected matroid could be represented as following,

$$\mathbf{A_{BDF}} = \begin{bmatrix} \mathbf{A_1} & & & \\ & \mathbf{A_2} & & \\ & & \ddots & \\ & & & \mathbf{A_n} \end{bmatrix}$$

where $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_n$ are sub-matrices and are called blocks. Each sub-matrix need not be of same size and the matrix $\mathbf{A_{BDF}}$ is said to be of block diagonal form.

Cho et al. (2008) mentioned that in general the design matrices in many engineering applications exhibiting some form of clustered structure may not be reshaped to the BDF form but can be reshaped to bordered block diagonal form (BBDF) as shown below.

$$\mathbf{A}_{\mathbf{BBDF}} = \begin{bmatrix} \mathbf{A}_1 & & & & \mathbf{B}_1 \\ & \mathbf{A}_2 & & & \mathbf{B}_2 \\ & & \ddots & & \vdots \\ & & & \mathbf{A}_n & \mathbf{B}_n \end{bmatrix}$$

where the sub-matrix $[\mathbf{B}_1^T\ \mathbf{B}_2^T \ldots \mathbf{B}_n^T]^T$ is called the border (**B**) of **A.**

There are a number of methods to permute the matrix **A** into the BBDF. Cho et al. (2007) adopted the hypergraph partitioning tools to obtain the BBDF. Another approach to find this is by using the bi-partite graph (Cho et al. 2004). Readers are advised to refer Cho et al. (2007) for information and references of such algorithms and tools which basically rely on graph theory and basic graph partitioning algorithms.

Let |B| denote the number of columns present in B and $[\mathbf{A}_i \cup \mathbf{B}_i]$ denote the matrix formed using the block $\mathbf{A}_i$ and its corresponding part $\mathbf{B}_i$ of the border matrix **B**.

Algorithm for Bound and Decomposition

1. Search for cogirth through *ES* until $d < 2|B| - 1$. If g^* still not found go to step 2
2. For $i = 1, 2, \ldots, n$, if there exists any $[\mathbf{A}_i \cup \mathbf{B}_i]_{(-d)}$ such that

   $$r([\mathbf{A}_i \cup \mathbf{B}_i])_{(-d)} < r([\mathbf{A}_i \cup \mathbf{B}_i])$$

   then stop, cogirth = d; else d = d + 1, repeat step 2.

where the value (2|B| - 1) is called as bound and the condition ($d < 2|B| - 1$) as bound condition. They prove such a condition must be satisfied in order for the procedure mentioned in step 2 to identify the cogirth accurately.

In the same work, they also presented another algorithm which can be considered as an extension of the above which has two modifications. First is that it uses lower bound value. As this reduces the number of bigger dimensional matrices ($\mathbf{A}_{\mathbf{BBDF}(-d)}$) that has to be generated and tested for rank deficiency, a lower bound is desirable. The second modification is that instead of performing *ES* on each of the blocks $[\mathbf{A}_i \cup \mathbf{B}_i]_{(-d)}$ individually for all $i = \{1, 2, \ldots, n\}$ for any specific value of d, a number (j) of blocks are combined together and *ES* is then performed. When j=n, the operation of this algorithm is exactly similar to basic exhaustive search and when j=1, then the operation is similar to the first algorithm. The authors also suggest a method to find an optimal value for j for a given value of d for the least computation time. Let $\mathbf{A}_j$ denote the collection of all matrices generated from the union of all possible j blocks (along with their border blocks) from $\{1, 2, \ldots, n\}$. The following algorithm summarizes the modified version of bound and decomposition

1. Search for cogirth through *ES* until $d < (n / (n\text{-}1))\ |\mathrm{B}|$. If g^* still not found go to step 2
2. Find optimal value for j
3. If there is any sub-matrix such that $r(\mathbf{A}_{j(-d)}) < r(\mathbf{A})$,
   then Cogirth of M = d, stop; else d=d+1, go to step 5

CHAPTER 3

PROPOSED ALGORITHM FOR FINDING SENSOR REDUNDANCY DEGREE

This section is divided into four subsections. Section 3.1 explains that the problem of determining the degree of sensor redundancy is equivalent to the problem of finding the cogirth (size of the smallest cocircuit) of a matroid specified over the transposed measurement matrix **H**. Section 3.2 describes the transformation of the problem of identifying the smallest cocircuit of a matroid into another problem of identifying the smallest circuit (or girth) of a different matroid. Section 3.3 details the method of determining the girth of the matroid by solving an optimization problem which is a ℓ^1-norm minimization problem. Section 3.4 summarizes the above mentioned steps and presents the final algorithm.

3.1 Degree of Sensor Redundancy - Cogirth Relation

The application of matroid theory to determine the degree of sensor redundancy was initially proposed by Cho et al. (2004) which used the bound and decomposition algorithm explained in chapter 2 previously. In order to understand the redundancy degree and cogirth relationship we shall first split the subsets of sensors into two groups – non-failure causing and failure causing subsets, and explain these two groups are the coindependent and codependent subsets respectively of a matroid obtained from $\mathbf{H}^T$. Then the relation between the degree of sensor redundancy and the cogirth of a matroid is presented.

3.1.1 Non-Failure Causing (NFC) and Failure Causing (FC) Subsets of Sensors

We shall first recap the association of degree of sensor redundancy with the rank of the measurement matrix explained in chapter 1. A sensor failure was represented by removal of its corresponding row from the measurement matrix **H**, whose original rank is

denoted by r(**H**). From (1.2) we know that the degree of sensor redundancy is equal to the size of the *smallest subset* of sensors or rows of **H** whose removal causes the rank of the resulting matrix reduced (i.e. $r(\mathbf{H}_{(-d)}) < r(\mathbf{H})$). Hence all possible subsets formed from the row set of **H** can be divided into two categories: one whose removal leads to a reduced matrix with rank unchanged and is called - *non-failure causing subsets* (NFC), and the other whose removal results in a reduced matrix with rank lower than r(**H**) and is called *failure causing subsets* (FC). From (1.2) the degree of sensor redundancy is equal to the size of the smallest FC subset minus one, i.e.

$$\text{Degree of Sensor Redundancy } \delta(\mathbf{H}) = |\text{ smallest FC subset }| - 1 \qquad (3.1)$$

3.1.2 The relation between NFC, FC subsets and Codependent sets/ Codependent sets of a Matroid

In this section we explain that the two types of subsets explained above: non-failure causing subsets and failure causing subsets are essentially the codependent sets and the coindependent sets respectively of the matroid M specified over $\mathbf{H}^T$, denoted by $M(\mathbf{H}^T)$. The subsequent lemmas provide the link between the states of the system (active, failure) with respect to the failures of sensors corresponding to the coindependent sets (or NFC sets) and codependent sets (or FC sets) of $M(\mathbf{H}^T)$.

Lemma 1. *Failure of sensors corresponding to the elements of any coindependent set of the matroid M does not cause failure of the system.*

Lemma 2. *Failure of sensors corresponding to the elements of any codependent set of the matroid M does cause failure of the system.*

Proof: Let M(**A**) be the vectorial matroid specified over a matrix **A** of p rows and n columns with $p < n$. Let E and I be the ground set and independent sets of M(**A**) respectively. In chapter 2, we saw that base of the matroid is the maximal independent set and its rank is equal to the rank of the matrix. It implies that the rank of any sub-matrix formed with only the columns corresponding to the elements of any base (b) is still equal

to the rank of original matrix. Another equivalent way of saying this is even on removal of columns corresponding to those elements not present in the base *b* (i.e. (*E* / *b*) or the corresponding cobase *b**) from the original matrix, the rank of the reduced matrix will still be equal to r($\mathbf{H}^T$).

Due to the same reason, even on removal of columns corresponding to any coindependent set (*i**), the resulting sub-matrix formed by columns corresponding to the elements present in *E* / *i** still has the rank equivalent to the rank of the full matrix. Conversely when the columns corresponding to a codependent set (*d**) are removed, the rank of the reduced matrix of *E* / *d** fall below the rank of full matrix.

Since the measurement matrix has redundant sensors defined on rows, while the linear dependency structure in matroid theory is expressed with the columns of a matrix, we need to work on the transposed **H** matrix. The following two statements which summarize the proof will be able to support the lemma 1 and 2 respectively.

(*S1*) *Since the compliment of any coindependent set is always either a base or a spanning set, removal of columns corresponding to a coindependent set from a matrix does not affect the rank of the reduced matrix.*

(*S2*) *Since the compliment of any codependent set is neither a base nor a spanning set, removal of columns corresponding to a codependent set from a matrix will affect the rank of the reduced matrix.*

From lemma 2, it is easy to see that removal of columns (sensors) corresponding to the *cocircuit*, which is a minimal codependent set, causes rank reduction (condition equivalent to the failure of the sensor network system). Combining this result with (3.1) it is clear that the smallest failure causing subset is the smallest cocircuit (SCC) of M($\mathbf{H}^T$) and,

$$\text{Degree of Sensor Redundancy } \delta(\mathbf{H}) = |\text{ SCC }| - 1 \qquad (3.2)$$

Now the problem of finding the degree of sensor redundancy is changed to identifying the smallest cocircuit of a matroid M. The basic exhaustive search method explained in chapter 2 is by far the only technique to determine the cocircuit of a general matroid. And if the matrix is sparse and clustered, bound and decomposition could be helpful. But as mentioned earlier, in this thesis, we transform this problem into finding the smallest circuit of another matroid and then develop an efficient algorithm for determining it. Details of these are explained in the sections henceforth.

3.2 Cogirth Problem into the Girth Problem

In order to understand the conversion of problem of determining the cocircuits ($\mathbb{C}^*$) of a matroid to the circuits ($\mathbb{C}$) of another matroid, we shall first recap that the cocircuits of the original matroid M are the circuits of its dual matroid M*. But in order to find the circuits of M* we need to have a matrix representing it. The following section describes the method of deriving this matrix so-called *standard representative matrix of the dual matroid*.

3.2.1 Dual Matroid Representation

Consider a matroid M defined over a matrix $\mathbf{B} = (\mathbf{H})^T$ of p rows and n columns with $p < n$. By performing Gauss-Jordan elimination on **B**, we can get the reduced row echelon form (rref) of **B** which have the following characteristics,

- All rows containing at least a single non-zero element are above any rows containing all zeroes.
- The pivot or the leading coefficient (the first non-zero number from the left) of a non-zero row is always strictly to the right of the leading coefficient of the row above it.

- The leading coefficient of each nonzero row is 1 and contains a zero both above and below it except for the first and last row which do not have zero above and below respectively.

By suitably shifting the columns in the RREF of **B,** it can be changed into $\mathbf{B}_{\boldsymbol{S}} = (\mathbf{I}_{\boldsymbol{r}} \mid \mathbf{D})$ where $\mathbf{I}_{\boldsymbol{r}}$ is the identity matrix of size r and r = rank(**B**). This matrix is called the standard representative matrix of M. Since the process of Gauss-Jordon elimination and subsequent shifting of columns won't affect the linear dependency in the matrix, M(**B**) = M($\mathbf{B}_{\boldsymbol{S}}$) = M($\mathbf{I}_{\boldsymbol{r}}$ | **D**). From matroid theory (Oxley 1992), the standard representative matrix for the dual matroid M*(**B**) is of the form ($-\mathbf{D}^{\mathrm{T}} \mid \mathbf{I}_{\boldsymbol{n-r}}$) (Cho et al. 2007). The procedure to obtain this matrix structure is presented below.

1. Perform Gauss-Jordon elimination to obtain rref of **B**.
2. If a row does not contain even a single non-zero number, eliminate that row.
3. Check if $\mathbf{I}_{\boldsymbol{r}}$ structure is present in the left side of the matrix. If not shift the columns appropriately until the structure $\mathbf{B}_{\boldsymbol{S}} = (\mathbf{I}_{\boldsymbol{r}} \mid \mathbf{D})$ is obtained. Keep track of the changes in the column numbers.
4. The negative of the transposed sub-matrix D is taken and concatenated with the identity matrix of size (n-r) to its right, forming the matrix $\mathbf{B}_{\boldsymbol{S}}^{*}$ of structure ($-\mathbf{D}^{\mathrm{T}} \mid \mathbf{I}_{\boldsymbol{n-r}}$).

The above mentioned procedure is illustrated as follows using the example matrix **A** introduced in chapter 2.

For our example matrix **A** (p=3, n=5),

$$\mathrm{RREF}(\mathbf{A}) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \text{let} \quad \mathrm{Col}\,(\mathbf{A})_{\mathrm{orig}} = \{1, 2, 3, 4, 5\}$$

By switching columns 3 and 4, we can get the desired structure of ($\mathbf{I}_r \mid \mathbf{D}$),

$$\mathbf{A}_{\boldsymbol{s}} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad \text{with} \quad \text{Col } (\mathbf{A})_{\text{new}} = \{1, 2, 4, 3, 5\}$$

r = rank(**A**) = 3, $n - r = 2$. Now we can get,

$$\mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{D} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$-\mathbf{D}^{\mathrm{T}} = \begin{bmatrix} -1 & -1 & 0 \\ 0 & -1 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Then the standard representative matrix of M* is given by,

$$\mathbf{A}_{\boldsymbol{s}}^{*} = \begin{bmatrix} -1 & -1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \end{bmatrix}$$

3.3 Optimization Problems to Determine Girth of a Matroid

In this section we first present the mathematical formulation for determining the girth of a matroid. The objective of this formulation is to find a solution vector with minimal ℓ^0-norm, in other words the sparsest solution (solution with fewest number of non-zero elements) of a system of underdetermined linear equations. Since this problem is NP-hard, alternative approaches that can provide close approximations to the sparsest solution are generally adopted. Solving the problem for solution with the minimal ℓ^1-norm is one such approach and is the one used in the proposed algorithm. The details of these two problems, brief description of the available methods to solve these two problems are presented.

3.3.1 Determining Girth by Solving ℓ^0-norm Minimization Problem

By definition, a circuit of a matroid is a minimal dependent set. Hence the problem of finding a smallest circuit can be reframed as a problem of identifying the fewest number of columns that are linearly dependent. Now consider the matrix $\mathbf{B}_{\boldsymbol{s}}^{*}$

representing the dual matroid whose circuit is to be determined. The columns corresponding to a smallest circuit of $M(\mathbf{B}_s^*)$ can be identified from the indices of non-zero elements of the vector $\mathbf{x} \in R^n$ found as a solution of the following optimization problem,

$$min \ \|\mathbf{x}\|_0 \tag{3.3}$$

$$s.t. \ \mathbf{B}_s^* \mathbf{x} = 0 \tag{3.4}$$

$$\mathbf{x} \neq 0 \tag{3.5}$$

where $\|\mathbf{x}\|_0 = \sum_{i=1}^{n} x_i^0$ is called the zero norm (or ℓ^0-norm) of **x** which equals the number of non-zero elements in **x**. Since the sparsest solution of **x** is equivalent to the smallest circuit, these two terms are used interchangeably hereafter.

Finding such sparsest solutions while satisfying certain linear constraints is a widely discussed problem in recent years in signal processing community. Their problem is very similar to (3.3)-(3.5) and can be written as,

$$min \ \|\mathbf{x}\|_0 \tag{3.6}$$

$$s.t. \ \mathbf{S}\,\mathbf{x} = \mathbf{t} \tag{3.7}$$

Their objective is to identify the signals (represented by vector **x**) responsible for producing the measurement vector **t** with the knowledge of a matrix **S** containing the linear transformation coefficients for each of the signal. When the number of linear equations is lesser than the number of signals ($r < n$), then the system is said to be underdetermined while the opposite ($r > n$) is called overdetermined. Our problem falls under the first category.

This problem is in general considered NP-hard, the main reason due to the non-convex objective function resulting in large number of local optima (Donoho and Elad 2006). The combinatorial increase in number of local minima with increase in the length of **x** causes the above optimization problem intractable. This makes it very difficult to

frame an algorithm that could provide direct solution, except for the exhaustive search which is impossible for even moderate length of **x**.

3.3.2 Alternate methods to approximate minimal ℓ^0-norm solution

The wide range of applications and fields that requires the identification of the sparsest solution has motivated researchers to develop alternate techniques that could either get the exact or close approximate solutions of minimal ℓ^0-norm. Some of the applications are overcomplete signal representation (Elad and Bruckstein 2002), compressed sensing (Donoho 2006c), imaging application: Spectroscopy, MRI (Donoho and Tsiag 2008), removal of impulsive noise (Chen et al. 1998).

The alternate methods include greedy and convex programming approaches (Donoho and Elad 2006). Though none of the techniques are able to completely surrogate the ℓ^0-norm minimization problem, under some specific conditions certain methods are able to find the exact minimal ℓ^0-norm solutions. We use the convex programming approach to find the sparsest solution in our proposed algorithm. This method is explained in detail in this subsection after a brief description of the two greedy approaches.

3.3.2.1 Greedy Algorithms

This algorithm is a heuristic greedy approach to find the sparsest solution for an underdetermined system of linear equations. It was developed by Pati et al. (1993) as a modification to another algorithm called matching pursuit. Tropp and Gilbert (2007) used this algorithm for signal recovery in random linear measurements which is equivalent to finding the sparsest solution in the problem (3.6)-(3.7). The underlying idea is to approximate **t** with fewest numbers of columns of **S** (through linear combinations) by greedy selection of one column per iteration that correlates best with the unexplained part of **t**.

Refer to (3.6)-(3.7) for notations that are used in the below description of this algorithm. The sparse solution is developed iteratively with the help of so called active set which maintains the list of nonzero elements of the sparse solution. Initially the active set is empty and the residual $\mathbf{r_0}$ is set equal to $\mathbf{t}$. During the j^{th} iteration, one column of $\mathbf{S}$ (not present in the active set) that is strongly correlated with the residual $\mathbf{r}_{j-1}$ is selected. This is done by determining the column that has the maximum absolute value for the inner product with the residual $\mathbf{r}_{j-1}$. Its column number is added to the active set. The coefficients for the active set columns (and hence the sparse solution $\mathbf{x}_j$ for j^{th} iteration) are determined by solving for minimal ℓ^2 norm of $\mathbf{S}\ \mathbf{x} - \mathbf{t}$, where coefficients of elements not present in the active set are made equal to 0. Denoting the contribution by the active set columns by $\mathbf{t}_j = \mathbf{S}\ \mathbf{x}_j$, the residual is updated as $\mathbf{r}_j = \mathbf{t} - \mathbf{t}_j$. This completes a single iteration. The procedure continues with the selection of one new column that best correlates the residual $\mathbf{r}_j$, adding it to the active set and updating of residual until the number of iterations equals to the certain user specified number.

Another greedy algorithm developed by Donoho et al. (2007) is called *stagewiseOMP*. In this method, unlike the OMP, more than one column can enter the active set at a given iteration. Similar to OMP, this algorithm also finds the correlations of each of the columns with the residual by finding the absolute values for their inner product. The heuristic method then adopted to find the set of columns that best correlates the residual, which is then added to the active set. This is done by determining the columns whose inner product with the residual is greater than a threshold. This process is called hard thresholding and the values are derived in their work. Then the vector $\mathbf{t}$ is projected on the columns present in the active set and its coefficients are determined. Then similar to OMP, the new residual is found by subtracting the contribution from the active set columns. The authors recommended number of iterations is 10.

3.3.2.2 Convex Programming Approaches

Another approach that could approximate the results of the problem (3.6)-(3.7) is by solving it with the objective function of minimizing the ℓ^1-norm of **x**, instead of the ℓ^0-norm. Now the optimization problem looks as below,

$$min \ \|\mathbf{x}\|_1 \tag{3.8}$$

$$s.t. \ \mathbf{S}\,\mathbf{x} = \mathbf{t} \tag{3.9}$$

where $\|\mathbf{x}\|_1$ is called the one norm (or ℓ^1-norm) of **x** and is equal to sum of absolute values of element of **x**,

$$\|\mathbf{x}\|_1 \ = \ |x_1| + \ |x_2| + \ . \ . \ . \ + |x_n| \tag{3.10}$$

The above optimization problem (3.8)-(3.9) is convex and is called as Basis Pursuit method in signal processing literature. This convex optimization problem can be formulated as a linear programming problem and can be solved by recognized methods like simplex method or interior point methods.

The linear programming problem equivalent to ℓ^1-norm minimization problem (3.8)-(3.9) can be obtained by suitable transformation of its variables. This is explained below. The problem (3.8)-(3.9) can be rewritten as,

$$min \ \|\mathbf{x}\|_1 \ = \ |x_1| + \ |x_2| + \ . \ . \ . \ + |x_n| \tag{3.11}$$

$$s.t. \ \mathbf{S}\,\mathbf{x} = \mathbf{t} \tag{3.12}$$

Changing the variable $x_i = \ x_i^+ - \ x_i^- \ \forall \ i \in \{1, 2, \ldots, n\}$, where both x_i^+ and $x_i^- \geq 0$, we have

$$min \ |x_1| + \ |x_2| + \ . \ . \ . \ + |x_n| \tag{3.13}$$

$$s.t. \mathbf{A}\,\mathbf{x}^* = \mathbf{b} \tag{3.14}$$

$$\mathbf{x}^* \geq 0 \text{ or } x_i^+ \ , x_i^- \ \geq \ 0 \ \forall \ i \in \{1, 2, \ldots, n\} \tag{3.15}$$

where $\mathbf{x}^* = \ [x_1^+ \quad x_2^+ \quad . \quad . \quad x_n^+ \quad | \quad x_1^- \quad x_2^- \quad . \quad . \quad x_n^-]_{2n}^{\mathbf{T}}$, $\mathbf{A} = (\mathbf{S} \mid -\mathbf{S})$ and $\mathbf{b} = \mathbf{t}$. The –**S** in $\mathbf{A} = (\mathbf{S} \mid -\mathbf{S})$ accommodates the negative values of **x** through $\mathbf{x}^-$ which is the second half of $\mathbf{x}^*$.

We then consider the following linear programming problem:

$$min\ x_1^+ + x_1^- + x_2^+ + x_2^- + \ldots + x_n^+ + x_n^- \tag{3.16}$$

$$s.t.\ \mathbf{A}\,\mathbf{x}^* = \mathbf{b} \tag{3.17}$$

$$\mathbf{x}^* \geq 0 \tag{3.18}$$

Problem (3.16)-(3.18) is equivalent to problem (3.13)-(3.15) because in the optimal solution of (3.16)-(3.18) either $x_i^+ = 0$ or $x_i^- = 0$ or both equal to $0\ \forall\ i \in \{1, 2, \ldots, n\}$. Therefore we can say that $x_i^+ + x_i^- = |x_i^+ - x_i^-| = |x_i|$ so that at the optimal solution of the objective function (3.16) is equivalent to (3.13).

Equation (3.16) is equal to the sum of all the elements of $\mathbf{x}^*$and can be rewritten as shown below, which is the linear programming form of the ℓ^1-norm minimization problem:

$$min \quad \mathbf{1}^T\,\mathbf{x}^* \tag{3.19}$$

$$s.t.\ \mathbf{A}\,\mathbf{x}^* = \mathbf{b} \tag{3.20}$$

$$\mathbf{x}^* \geq 0 \tag{3.21}$$

The above linear programming problem (3.19)-(3.21) can be solved using the general purpose LP-solvers using simplex method or interior point methods. Consequently, the problem has become tractable and solution is attainable in most cases.

3.3.2.3 Conditions for Good Approximation of minimal ℓ^1-norm solution to sparsest solution

Several researchers have worked on determining the conditions under which the sparsest solution (ℓ^0-norm) and the minimal ℓ^1-norm solution are same for a given problem. The general requirement is that the ℓ^0-norm solution should be sufficiently sparse. Tsaig and Donoho (2006) used the Equivalence Breakdown Point (EBP) to quantify this sufficient sparsity in the ℓ^0-norm solution. The EBP is the number of non-zero elements that a sparsest solution can have to guarantee it is also the minimal ℓ^1-norm solution. The threshold EBP depends on the size and type of the matrix. They developed

heuristic method to determine the upper and lower bounds and hence an approximate formula for EBP values for different kind of matrices. The accuracy was found to be very high for the matrices from certain ensembles they had considered. Such bounds for any matrix from any ensemble can be obtained by conducting only few experiments following the heuristic approach. The reported approximate EBP for matrices from uniform spherical ensemble is equal to 0.44 p/ log (2n/p) and for matrices from random sign, partial Hadamard, partial Fourier ensembles is equal to 0.65 p / log (1 + 10n/p). Note n and p are number of columns and rows in a matrix.

Donoho (2006b) reported that the minimal ℓ^1-norm solution for a large underdetermined system of linear equations is unique and equal to the sparsest solution, provided the solution is sufficiently sparse. He also showed that there cannot be a sparsest solution with fewer than EBP non-zeros if the minimal ℓ^1-norm solution has EBP or greater number of non-zeroes. The above statement holds true for most of the problems and only a negligible fraction does not fall in this category. More information related to the necessary conditions for these two solutions to be equal can be found in Donoho (2006a, 2006b), Candes, Romberg and Tao (2005b).

The proposed algorithm uses the above mentioned convex approximation: ℓ^1-norm minimization in order to solve the ℓ^0-norm minimization problem defined earlier by (3.3)-(3.5) in order to determine the girth of the dual matroid, equivalently the cogirth of the original matroid. This is explained in detail with an example in the subsequent section. Even though this method is computationally expensive compared to the greedy heuristic algorithms such as OMP and StOMP, it is chosen after reviewing the literature of superior theoretical and empirical results supporting determination of exact sparsest solution in wide variety of problems. This was also noticed in our work when we used and compared OMP, StOMP with the minimal ℓ^1-norm approximation method on selected problem instances.

3.3.3 Cogirth by Solving ℓ^1-norm Minimization Problem

We saw that the problem of finding cogirth can be formulated as in (3.3)-(3.5). As discussed in the preceding section, we will apply ℓ^1-norm approximation method to solve (3.3)-(3.5). Applying ℓ^1-norm approximation to the objective function in (3.3), we get the following optimization problem,

$$min \ \|\mathbf{x}\|_1 \tag{3.22}$$

$$s.t. \ \ \mathbf{B}_{\boldsymbol{S}}^{*}\,\mathbf{x} = 0 \tag{3.23}$$

$$\mathbf{x} \neq 0 \tag{3.24}$$

The constraint (3.24) can be enforced by solving a series of convex optimization problems as follows for $i = 1, \ldots, n$:

$$min \ \|\mathbf{x}\|_1 \tag{3.25}$$

$$s.\mathbf{t}. \ \ \mathbf{B}_{\boldsymbol{S}}^{*}\,\mathbf{x} = 0 \qquad i = 1, \ldots, n \tag{3.26}$$

$$x_i = -1 \tag{3.27}$$

where x_i denotes the i^{th} element of the vector $\mathbf{x}$. Let $\mathbf{x}_{(-\boldsymbol{i})}$ denote the vector without the i^{th} element. Similarly let $[\mathbf{B}_{\boldsymbol{S}}^{*}]_{(\boldsymbol{i})}$ denote its i^{th} column of the matrix $\mathbf{B}_{\boldsymbol{S}}^{*}$, and $[\mathbf{B}_{\boldsymbol{S}}^{*}]_{(-\boldsymbol{i})}$ denote the matrix without the i^{th} column. Now the problem (3.25)-(3.27) can be re-written as,

for $i = 1, \ldots, n$:

$$min \left\|\mathbf{x}_{(-\boldsymbol{i})}\right\|_1 \tag{3.28}$$

$$s.t. \ [\mathbf{B}_{\boldsymbol{S}}^{*}]_{(-\boldsymbol{i})} \ \mathbf{x}_{(-\boldsymbol{i})} = [\mathbf{B}_{\boldsymbol{S}}^{*}]_{(\boldsymbol{i})} \tag{3.29}$$

The problem is converted to n ℓ^1-norm minimization problems, where n is the number of columns in matrix $\mathbf{B}_{\boldsymbol{S}}^{*}$ (or number of sensors used in the sensor network). Each of these problem is a ℓ^1-norm minimization problem described by (3.8)-(3.9) with the vector $\mathbf{t}$ being replaced by the i^{th} column of $\mathbf{B}_{\boldsymbol{S}}^{*}$.

For the optimal solution of problem i above, i = 1, . . , n, the column set corresponding to the non-zero elements of $\mathbf{x}$ (including x_i) is expected to be the smallest circuit of M($\mathbf{B}_{\boldsymbol{S}}^{*}$) containing the i^{th} column (since we assigned x_i to a nonzero value -1).

By determining such sets for all values of i, the smallest circuit of M($\mathbf{B}_s^*$) can be determined. Then from section 3.1 and 3.2, the sensor redundancy degree can be found by:

$$\text{Degree of Sensor Redundancy } \delta(\mathbf{H}) = |\text{ Smallest Circuit of M}(\mathbf{B}_s^*)| - 1 \quad (3.30)$$

where $\mathbf{B}_s^*$ is the standard representative matrix of M*($\mathbf{H}^T$).

We will use the example matrix

$$\mathbf{A}_s^* = \begin{bmatrix} -1 & -1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 & 1 \end{bmatrix}$$

derived in section 3.2 to illustrate the use of ℓ^1-norm approximation to find the cogirth of M($\mathbf{A}$). We will use the ℓ^1-norm minimization problem (3.16)-(3.18) with i=5 by setting x_5 = -1 .

Let $[\mathbf{A}_s^*]_{(-5)} = \begin{bmatrix} -1 & -1 & 0 & 1 \\ 0 & -1 & -1 & 0 \end{bmatrix}$ and $[\mathbf{A}_s^*]_{(5)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$$\mathbf{x}_{(-5)} = (x_1\ x_2\ x_3\ x_4)^{\mathrm{T}}$$

Problem (3.28)-(3.29) becomes

$$min\ \|\mathbf{x}_{(-i)}\|_1$$
$$s.t.\ [\mathbf{A}_s^*]_{(-5)}\ \mathbf{x}_{(-5)} = [\mathbf{A}_s^*]_{(5)}$$

Solving the problem above, the optimal solution is $(0, 0, -1, 0)^{\mathrm{T}}$. Based on the indices of the non-zero coefficients of $\mathbf{x}$, the column set $\{[\mathbf{A}_s^*]_3$, and $[\mathbf{A}_s^*]_5\}$ is the smallest circuit containing the 5th column of $\mathbf{A}_s^*$. Similarly we can solve the corresponding ℓ^1-norm minimization problems corresponding to i = 1, 2, 3, 4. None of those problems generate a circuit smaller than {3, 5}.

Note that even though the circuits of M* are cocircuits of M, this set {3, 5} is not a cocircuit of M because of the rearrangement of the columns performed to obtain $\mathbf{A}_s$ and subsequently $\mathbf{A}_s^*$, which were explained in section 3.2.1. By keeping track the column indices in these re-arrangements, the actual column set can be identified as {4, 5}. This

can also be confirmed a smallest cocircuit by checking the list of all cocircuits of M(**A**) found in the section 2.1.1.

3.4 Summary of the Proposed Algorithm

The structure of our algorithm is shown in the Figure 1. The following algorithm summarizes the methodology explained in this chapter to determine the redundancy degree of the linear sensor system with the knowledge of the measurement matrix. In this algorithm, Col($\mathbf{y}_i$) denotes the column set (in $\mathbf{B}_s^*$) corresponding to the nonzero elements in $\mathbf{y}_i$.

Algorithm to find the redundancy degree of a linear sensor system

1. Let **H** be the Measurement matrix. Find $\mathbf{H}^T$. Orig_Col = column arrangement of $\mathbf{H}^T$
2. Using the *Procedure* explained in section 3.2, determine the standard representative matrix $\mathbf{B}_s^*$. New_Cols = column arrangement of $\mathbf{B}_s^*$. Let n = number of columns in $\mathbf{B}_s^*$.
3. For $i = 1$ to n, let $x_i = -1$

   find $\mathbf{y}_i = argmin \left\|\mathbf{x}_{(-i)}\right\|_1$

   $s.t.\ \ [\mathbf{B}_s^*]_{(-i)}\ \mathbf{x}_{(-i)} = [\mathbf{B}_s^*]_{(i)}$

   find $z_i = ||\mathbf{y}_i||_0$
4. Determine $z_{\min} = \min(z_i, i \in \{1, 2, \ldots, n\})$
5. $j = 0$. For $i = 1$ to n, If $z_i = z_{\min}$ then $j=j+1$, smallest circuit SC_j = Col($\mathbf{y}_i$)
6. Identify the corresponding smallest cocircuit SCC for each of SC found in step 5 by matching its columns with Orig_Cols
7. Sensor redundancy degree $\delta(\mathbf{H})$ = Size of SCC – 1 (*or*) $\delta(\mathbf{H}) = z_{\min} - 1$.

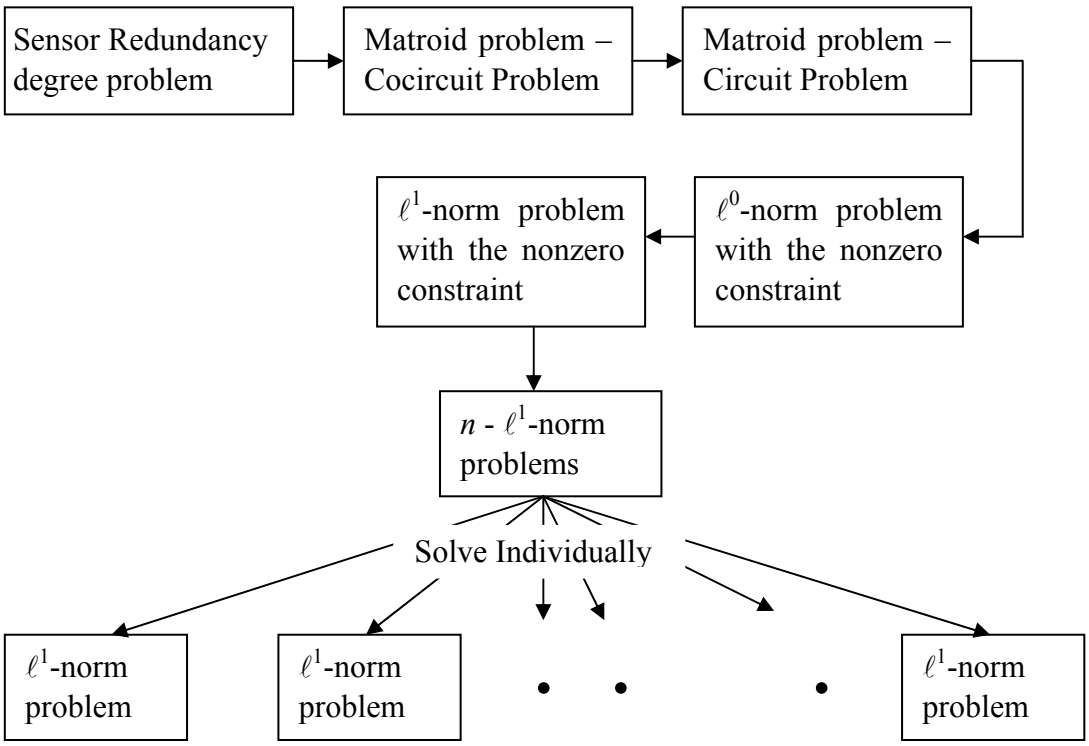

Figure 1 Structure of the proposed algorithm

CHAPTER 4

RESULTS AND DISCUSSION

This chapter is divided into three sections. Section 4.1 provides the relevant details about the software and the computer system used. Section 4.2 shows the results and discusses the performance of the proposed algorithm in terms of accuracy and computation time on 82 problem instances that were tested upon. Also details of the problem instances are given. The chapter is then concluded with a brief summary of the results obtained and inference on the performance of the algorithm.

4.1 Software and Machine details

The algorithm was written in Matlab 7.4.0 (R2007 a) and is given in Appendix B. To find the minimal ℓ^1-norm solution for an underdetermined system of linear equations, a function named SolveBP was used. It is one of the several functions of the Matlab software package called SparseLab available at http://sparselab.stanford.edu/. The library routines are free of charge and are mainly developed to find sparse solutions to underdetermined system of linear equations.

The function SolveBP intakes the parameters: the matrix $\mathbf{B}^*_{\boldsymbol{s}_{(-i)}}$, scalar value equal to the length of $\mathbf{x}_{(-i)}$, and the vector $\mathbf{B}^*_{\boldsymbol{s}_{(i)}}$ (Refer (3.28)-(3.29)). After converting the problem into a linear programming problem as described in section 3.3.2.2, it calls another function called pdco which is abbreviation for Primal Dual barrier method for Convex Optimization problem with linear constraints. This function uses log-barrier method for interior point search and determines the solution.

The computer system used to test the algorithm and analyze its performance had processing speed of 2.83 GHz and a RAM of 3GB.

4.2 Results and Discussion

The proposed algorithm was tested on 88 problem instances (or measurement matrices); 82 of which are based on random matrices and the remaining 6 have special structures. This section is divided into two subsections, one each providing short description of the problem instances followed by the discussion of the performance of the proposed algorithm for two different kinds of matrices studied in this work.

We also tried to confirm the findings reported in the literature that the ability of ℓ^1-norm minimization method to find / approximate the minimal ℓ^0-norm solution is better than the greedy approximations such as OMP and StOMP. Function SolveOMP and SolveStOMP available from the SparseLab package were used for this. Unless otherwise mentioned, the 'proposed algorithm' refers to the algorithm explained in section 3.4 using the ℓ^1-norm minimization technique. The algorithms using the OMP or StOMP techniques are referred as the 'greedy algorithms'.

4.2.1 Random Measurement Matrices

These problem instances are called random as most of its entries are obtained randomly from a continuous uniform distribution over the interval [0, 100]. Appendix A describes in detail the method adopted in generation of these data sets. Note that the cocircuits of a matroid specified over the transposed measurement matrix is equivalent to the list of failure causing subsets of sensors. And the redundancy degree is one less than the size of the smallest cocircuit, namely the cogirth.

These problem instances were simulated and can be classified based on the size of the measurement matrix and the cocircuits present in it. We classify these problem instances into four categories. Category 1 has matrices containing only one cocircuit. Category 2 has matrices containing cocircuits of multiple sizes but only one smallest cocircuit and overlaps between two cocircuits are possible. Category 3 has matrices with several cocircuits of varying sizes and no overlap between any two of them. And

category 4 is similar to category 3 except that overlap may exist between any two cocircuits. In each of the four categories specified above, seven different sizes of matrices were tested. This leads to 4 x 7 = 28 different combinations of data sets. Most of these combinations are tested with more than one size of the cogirth. For example, in category 3 (see Table C.3), for the matrix of size 150 x 60, three different data sets are created corresponding to the cogirth values of 5, 8 and 12, respectively. More details including the category, size of the matrix, size of the cogirth, number of smallest cocircuits, and the list of cocircuits present in each matrix are shown in Appendix C. The Matlab code used to generate the data sets is given in Appendix B.

Our proposed algorithm is applied for all problem instances. Greedy algorithms were tested and compared using the problem instances under category 3 and 4. Tables 1 and 2 shows the performance of the proposed algorithm on the problem instances from category 1 and 2 respectively. It was able to identify the unique and the smallest cocircuit for 17 of 20 problem instances accurately in category 1 and 12 of 15 problem instances in category 2. The computation times mentioned in this work are generally in terms of seconds unless otherwise mentioned explicitly as minutes. The last column in Table 1 shows the results based on the exhaustive search method for the category 1 problem instances. It can be seen that, while our proposed algorithm can converge fairly quickly for all problem instances, the exhaustive search method is not able to find a cocircuit within 12 hours for most problems.

Table 1. Performance of Proposed Method on the Category 1 Random Problem Instances

No.	Size	CoGirth Size	# Smallest Cocircuits	# Smallest Cocircuits determined	Computation Time (Sec)	Exhaustive Search (Time)
1	25 x 12	5	1	1	0.5378	4.22
2		7	1	*S14/ 25	0.5571	66.68
3		8	1	*S13/ 2	0.5802	146.66
3	70 x 30	5	1	1	1.9716	140 Mins
4		8	1	1	1.9353	> 12 Hrs
5		12	1	*S39/ 1	1.6527	
6	150 x 60	7	1	1	8.03	
7		11	1	1	8.56	
8		15	1	1	6.79	
9	220 x 60	7	1	1	27.85	
10		12	1	1	29.08	
11		16	1	1	28.87	
12	325 x 150	7	1	1	69.73	
13		13	1	1	71.69	
14		17	1	1	72.34	
15	500 x 200	8	1	1	325.78	
16		16	1	1	323.23	
17		21	1	1	324.82	
18	1000 x 250	10	1	1	96.22 Mins	
19		20	1	1	96.10 Mins	
20		30	1	1	95.50 Mins	

* SΦ/ N -Size of the smallest cocircuit is Φ and N is the number of the smallest cocircuits determined.

Table 2. Performance on the Category 2 Random Problem Instances

No	Size	CoGirth Size	# Smallest Cocircuits	Other Cocircuit Size (nos)	# Smallest Cocircuits Determined	Computation Time (sec)
1	25 x 12	5	1	6(2)	1	0.5363
2		7	1	8(2)	*S14 / 25	0.7498
3	70 x 30	7	1	8(2), 9(1)	1	1.95
4		10	1	11(2), 12(1)	*S39 / 1	2.08
5	150 x 60	8	1	9(2), 10(1)	1	8.99
6		12	1	13(2), 14(1)	1	9.38
7	220 x 60	9	1	10(3)	1	30.87
8		14	1	15(1), 16(2)	1	30.68
9	325	9	1	10(2), 11(1)	1	70.99
10	x	14	1	15(1), 16(2)	1	69.51
11†	150	17	1	18(2), 19(1)	*S17† / 2	71.83
12	500 x	12	1	13(3), 14(2)	1	318.49
13	200	18	1	19(3), 20(2)	1	322.03
14	1000 x	15	1	16(3), 17(2)	1	94.1 Mins
15	250	25	1	26(3), 27(1)	1	77.05 Mins

* SΦ/ N: Φ is the size of the smallest cocircuit and N is the number of the smallest cocircuits determined.

† Both of the determined cocircuits were incorrect although the size is equal to the true size.

As mentioned earlier we use the problem instances in categories 3 and 4 to evaluate and compare the performances of the three methods – ℓ^1-norm minimization, OMP, and StOMP in finding the minimal ℓ^0-norm solution and consequently the cogirth (g^*) and the redundancy degree (δ). Table 3 presents the results obtained upon using these methods for problem instances in category 3. Note that for certain problem instances (no. 13 and 16) in Table 3 multiple trials were performed. Each trial had the same cocircuits but the data sets (individual entries) were different. This was performed after the initial observation of variations in the number of smallest cocircuits being determined by the algorithm.

From the results in Table 3, it can be seen that out of the total 22 problem instances in this category of random matrices the proposed algorithm found the cogirth accurately for 21 of them. Meanwhile the greedy algorithms - OMP and StOMP found the cogirth accurately for only 15 and 20 problems respectively. This shows the superior performance of ℓ^1-norm minimization technique over OMP and StOMP to determine the sparsest solution. The comparison in terms of accuracy of cogirth is shown in Table 4.a. In terms of computation times both the heuristic algorithms are faster than the convex optimization method which is similar to the results reported in the literature.

Table 3. Performance on the Category 3 Random Problem Instances

No	Size	Co Girth Size	# Small-est Co-circuits	ℓ^1-norm minimization		OMP		StOMP	
				g	no. found / time	*g*	no. found / time	*g*	no. found / time
1	25 x 12	5	3	3	3 / 0.5185	5	2 / 0.0508	5	2 / 0.032
2		8	2	13*	2 / 0.5846	7*	1 / 0.0509	7*	1 / 0.03
3	70 x 30	5	4	5	4 / 1.61	5	4 / 0.2944	5	1 / 0.1576
4		8	4	8	4 / 1.65	8	3 / 0.2914	8	1 / 0.1478
5	150 x 60	5	4	5	4 / 7.75	5	4 / 2.123	5	3 / 1.113
6		8	4	8	4 / 8.03	8	4 / 2.178	8	1 / 0.6265
7		12	4	12	3 / 7.26	5*	1/ 2.12	12	1 / 0.5295
8	220 x 60	5	7	5	7 / 25.76	5	7 / 8.127	5	4 / 4.765
9		10	10	10	10 / 22.44	10	10+<u>2</u> / 3.64	10	8 / 4.44
10		15	4	15	4 / 25.19	15	4 / 4.25	15	1 / 1.948
11	325 x 150	5	7	5	7 / 62.88	5	7 / 14.78	5	1 / 5.86
12		10	9	10	9 / 59.43	10	9 / 14.37	10	2 / 5.952
13		15	7	15	7 / 62.41	14*	1 / 15.95	15	3 / 6.16
			7	15	3 / 49.38	6*	1 / 9.85	15	4 / 5.174
			7	15	4 / 56.38	15	4+<u>2</u> / 11.01	15	4 / 4.778
			7	15	5 / 61.47	15	7 / 11.98	15	3 / 4.97
14	500 x 200	10	7	10	7 / 288.96	10	7 / 100.47	10	4 / 18.056
15		15	7	15	7 / 284.5	15	7 / 92.36	15	2 / 17.014
16		20	7	20	2 / 269.92	19*	1 / 102.08	20	3 / 17.19
			7	20	3 / 196.45	19*	1 / 103	19*	1 / 17.025
			7	20	3 / 201.39	7*	2 / 21.92	20	2 / 18.86
17	1000 x 250	17	10	17	10 / <u>80.1Min</u>	16*	1 / <u>77 Mins</u>	17	5 / <u>8.11Min</u>

* Cogirth determined is incorrect

Underlined number is the number of incorrect smallest cocircuits determined

Another parameter that can be used to compare the accuracies of the methods is the number of true smallest cocircuits being found. Consider a problem instance having matrix $(\mathbf{H})^T$ with n columns and k smallest cocircuits of size g^*. Let k' be the number of smallest cocircuits found by a given method. Table 5 compares the three methods in terms of $\sum k'$ which is the sum of all k' for all the problem instances.

Table 4. Comparison of ℓ^1-norm, OMP, StOMP with respect to Accuracy of δ

	ℓ^1-norm	OMP	StOMP
Total no. of problems tested	22	22	22
No. of problems δ determined accurately	21	15	20
True percentage	95.45%	68.18%	90.9%

Table 5. Comparison of ℓ^1-norm, OMP, StOMP with respect to Accuracy of k'

	True	ℓ^1-norm	OMP	StOMP
$\sum k'$	135	109	79	55

From Table 5, it is clear that $\sum k'$ for our proposed method is much larger than that of the greedy algorithms implying it can determine comparatively larger number of smallest cocircuits which is desirable.

The following discussion is only on the results of the proposed algorithm which uses the ℓ^1-norm minimization method. The proposed algorithm determined the g^* accurately for all but one problem instances (no. 2). Excluding problem number 2, it also determined all of the smallest cocircuits for all the problems except instance numbers 7, 13 and 16. Results from Table 1, 2, 3 indicate that when the cogirth is small (with respect to the size of the matrix), the algorithm can identify the cogirth and all the smallest cocircuits correctly. From Table 3 (especially problem no. 13, 16), we can see that when the cogirth becomes larger, the algorithm may be able to identify only some of the smallest cocircuits. At the same time when the cogirth is too large, the algorithm may fail to find any of the smallest cocircuits so that the cogirth determined would be incorrect.

For the girth problem with a random matrix, Tsaig and Donoho (2006) showed that there exists a so-called Equivalence Breakdown Point (EBP) such that when the girth is smaller than EBP, the ℓ^1-norm minimization method can always find the exact solution and hence the correct girth. Similarly, we conjecture that, for the cogirth problem, when the cogirth is smaller than a threshold l^*, the ℓ^1-norm minimization method can always find the correct cogirth. Further, when there are multiple smallest cocircuits, there exists a threshold l^* such that the ℓ^1-norm minimization method can find *all* the smallest cocircuits, and a threshold $u^*>l^*$ such that the ℓ^1-norm minimization method can always find the cogirth correctly and at least one smallest cocircuits when the cogirth is smaller than u^*. The results from all the problems we tested support this conjecture. Further study on this conjecture and how to find/approximate u^* and l^* is left for future work.

Next we present the results from testing of problem instances belonging to the category 4 - cocircuits of multiple sizes with overlapping. These problems were also tested with the greedy algorithms and the results are shown in Table 6. As we can see from the table three trials were performed for each pair of matrix dimensions and cogirth size. All the three trails of a specific pair had measurement matrices of same dimensions with same set of cocircuits but different random entries.

Table 6. Performance on the Category 4 Random Problem Instances

No	Size	Co Girth	# Smallest Cocircuit	Method	Incorrect Cogirth{*if any*}* / (# Smallest Cocircuits determined) Computation time in Sec		
					Trial 1	*Trial 2*	*Trial 3*
1	25 x 12	5	3	ℓ^1-norm	(3) 0.5460	(3) 0.5832	(3) 0.5991
				OMP	(2) 0.05	14*/ (22) 0.053	(2) 0.05
				StOMP	(2) 0.0348	(3) 0.0366	(3) 0.0346
2	70 x 30	5	6	ℓ^1-norm	(4) 1.755	(4) 1.7765	(4) 1.8981
				OMP	(4) 0.2803	(4) 0.2942	(3) 0.319
				StOMP	(1) 0.1346	(3) 0.1414	(3) 0.1456
3		8	4	ℓ^1-norm	(4) 1.8975	(3) 1.85	(2) 1.92
				OMP	(2) 0.3552	(3) 0.337	(3) 0.35
				StOMP	39*/ (1) 0.539	(1) 0.1344	(1) 0.1396
4	150 x 60	5	9	ℓ^1-norm	(7) 7.1105	(6) 7.97	(5) 7.4271
				OMP	(6) 1.55†	(4) 1.771	(3) 1.83
				StOMP	(7) 1.34	(2) 1.05	(3) 1.04
5		8	7	ℓ^1-norm	(6) 8.1026	(5) 8.1785	(5) 8.034
				OMP	(4) 2	(4) 1.85	7* / (1) 1.77
				StOMP	(3) 1.286	(1) 1.01	(2) 1.217
6	220 x 60	8	7	ℓ^1-norm	(4) 27.713	(5) 30.5	(6) 25.1
				OMP	(2) 7.84	(4) 8.58	(4) 7.285
				StOMP	(3) 5.214	(2) 3.0312	(3) 3.1405
7	325 x 150	14	7	ℓ^1-norm	(7) 67.31	(5) 53.64	(7) 68.37
				OMP	(7) 17.08	(6) 10.15†	(7) 18.16
				StOMP	(2) 5.69	(3) 6.4317	173* / (1) 4.6
8	500 x 200	10	8	ℓ^1-norm	(7) 314.2	(7) 315.25	(8) 326.27
				OMP	(8) 137.60	(8) 135.16	(8) 124.26
				StOMP	296*/ (1)17.17	296* / (1) 17	295* / (1) 17.55
9	1000 x 250	17	10	ℓ^1-norm	(10) 82.41 Mins		
				OMP	(10) 40 Mins		
				StOMP	(4) 8.96Mins		

* Incorrect cogirth found

First we shall compare the performances of the three methods and then discuss the results of the proposed algorithm only. Each trial can be considered as an individual problem which would result in a total of 27 problem instances. Again in terms of accuracy of δ and $\sum k'$, the results were better and favorable for the proposed algorithm using the ℓ^1-norm minimization method over the greedy algorithms.

Table 7. Comparison of ℓ^1-norm, OMP, StOMP with respect to Accuracy of δ

	ℓ^1-norm	OMP	StOMP
Total no. of problems tested	25	25	25
No. of problems δ determined accurately	25	23	20
True percentage	100%	92%	80%

Table 8. Comparison of ℓ^1-norm vs. OMP / StOMP with respect to Accuracy of k'

	True	ℓ^1-norm	OMP	StOMP
$\sum k'$	183	127	108	51

The following discussion is concerned with the performance of the proposed algorithm which uses the ℓ^1-norm minimization method. Though it determined the redundancy degree (cogirth) for all the problem instances exactly, the algorithm was unable to detect all the cocircuits uniquely for most problems. This may be because the cogirth could be in the range between l^* and u^*. But on comparing the results of similar problem instances from category 3 and 4 we can notice that comparatively fewer cocircuits were determined for the instance from category 4. By similar problem instances we meant instances with similar dimensions for the measurement matrix and cogirth. This suggests that overlapping of cocircuits has some negative effect in determining all of them uniquely.

4.2.2 Measurement Matrices with Special Structure

This section presents the results of the proposed algorithm on problem instances which have measurement matrices of certain special structure – sparse and clustered. Let **H** denote the measurement matrix of this class of problems, then its special structure formally called as bordered block diagonal form (BBDF) is as following,

$$\mathbf{H} = \begin{bmatrix} \mathbf{A}_1 & & & & & \\ & \mathbf{A}_2 & & & & \\ & & . & & & \\ & & & . & & \\ & & & & . & \\ & & & & & \mathbf{A}_n \\ \mathbf{B}_1 & \mathbf{B}_2 & . & . & . & \mathbf{B}_n \end{bmatrix}$$

where $\mathbf{A_1}$ - $\mathbf{A_n}$ are called blocks (sub-matrices) along the diagonal and $\mathbf{B_1}$ - $\mathbf{B_n}$ are their corresponding sub-matrix in the border **B** of the **H**. Problems of this nature can be found in multi-station assembly processes (Cho et al. 2009a, Ding et al. 2002, Kim and Ding 2004) and wireless sensor networks (Cho at al. 2009b).

The general form of **H** for these problems need not have the above structure explicitly. The design matrices of many engineering problems can be converted to the above structure by proper rearrangement of rows and columns. Cho et al. (2009a) developed an enhanced procedure, based on the Aykanat's algorithm, for transforming a matrix to BBDF by modeling a matrix using a bipartite graph and using Even's algorithm. The matrix **H** of one such simple problem instance found in multi-station assembly process reported in literature is shown in Figure 2a and its BBDF is shown in Figure 2b (Cho et al. 2007).

$$\mathbf{H} = \begin{bmatrix}
1 & 1.8 & -1.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1.8 & -1.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & .2857 & -.2857 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & .2857 & -.2857 & 0 & 0 & 0 \\
1 & -.9 & .9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1.26 & -.26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & .56 & .44 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & .55 & -.55 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -.35 & .35 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -.3313 & 0 & 1 & .55 & -.2186 & -1.0041 & -.0297 & 0 & .0041 & 0 & .0297 \\
0 & 0 & 0 & 0 & 1.09 & -.09 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .5 & .5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -.375 & 1.375 \\
-1 & .2108 & 0 & 1 & -.35 & .1392 & -.9675 & .2380 & 0 & -.0325 & 0 & -.2380 \\
0 & 0 & 0 & 0 & -.16 & 1.16 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -.2857 & .2857 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.2857 & -.2857 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .5714 & .4286 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -.1429 & 1.1429 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2.25 & -2.25 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & -2 \\
0 & 0 & 0 & 0 & .465 & .535 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -.14 & 1.14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & .55 & -.55 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1.125 & 1.125 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.375 & -.375
\end{bmatrix}$$

a. General form of the design matrix

$$\mathbf{H}_{\mathrm{BBDF}} = \begin{bmatrix}
1 & 1.8 & -1.8 & & & & & & & & & \\
1 & 1.8 & -1.8 & & & & & & & & & \\
1 & -.9 & .9 & & & & & & & & & \\
0 & 1.26 & -.26 & & & & & & & & & \\
0 & .56 & .44 & & & & & & & & & \\
0 & -.14 & 1.14 & & & & & & & & & \\
 & & & 1 & .55 & -.55 & & & & & & \\
 & & & 1 & .55 & -.55 & & & & & & \\
 & & & 1 & -.35 & .35 & & & & & & \\
 & & & 0 & 1.09 & -.09 & & & & & & \\
 & & & 0 & .47 & .54 & & & & & & \\
 & & & 0 & -.16 & 1.16 & & & & & & \\
 & & & & & & 1 & .29 & -.29 & & & \\
 & & & & & & 1 & .29 & -.29 & & & \\
 & & & & & & 1 & -.29 & .29 & & & \\
 & & & & & & 0 & 1.29 & -.29 & & & \\
 & & & & & & 0 & .57 & .43 & & & \\
 & & & & & & 0 & -.14 & 1.14 & & & \\
 & & & & & & & & & 1 & 2.25 & -2.25 \\
 & & & & & & & & & 1 & 2 & -2 \\
 & & & & & & & & & 1 & -1.13 & 1.13 \\
 & & & & & & & & & 0 & 1.38 & -.38 \\
 & & & & & & & & & 0 & .5 & .5 \\
 & & & & & & & & & 0 & -.38 & 1.38 \\
-1 & -.33 & 0 & 1 & .55 & -.22 & -1 & -.03 & 0 & 0 & 0 & .03 \\
-1 & .21 & 0 & 1 & -.35 & .14 & -.97 & .24 & 0 & -.03 & 0 & -.24
\end{bmatrix}$$

b. The permuted measurement matrix of form BBDF

Figure 2. The Design Matrix of the multi-station assembly process (Prob 1 in Table 6)

Source: Cho, J.J., Chen, Y. and Ding, Y., Journal, *Discrete Applied Mathematics*, 2007, Vol. 155, pp. 2456-2470.

Six problem instances of this type have been tested and the results are presented in Table 9. In general the cocircuits are overlapping in all the problems. The table also provides the computation times by exhaustive search and bound and decomposition algorithm (explained in section 2.7), which is a method designed specifically for problems with clustered structures.

Table 9. Performance Comparison on matrices with Special Structure

No	Size	Original cogirth Size	Proposed Algorithm		Computation Time		
			g found	no. found	Proposed Algorithm	Exhaustive Search	Bound and Decomposition
1	26 x 12	5	5	10	0.57 sec	8 sec	0.1 sec
2*	66 x 27	8	7	1	1.93 sec	>120 h	6.1 Min
3	154 x 72	5	5	38	6.51 sec	>120 h	120 Min
4*	221 x 55	14	16	7	35.25 sec	>120 h	16.5 Min
5	318 x 144	5	5	36	52.17 sec	>120 h	>120 h
6*	1009 x 252	16	18	1	102 Min	>120 h	38.2 Min

* Incorrect cogirth determined by the proposed algorithm

h - Hours

Instance no. 1 and 2 are from multi-station assembly, no 3, 4, 5 are from sensor networks and no. 6 is hypothetical.

The proposed algorithm determined cogirth incorrectly for three out of six problem instances. As expected the exhaustive search method runs into heavy computation even for moderate sizes of **H** and g^*, making it not a practical method. When compared to bound and decomposition algorithm, the computation time for the proposed algorithm is much shorter duration for four problem instances. And the bound and decomposition algorithm fails to converge on one problem. In general, from the six

problem instances considered, the overall accuracy of the proposed algorithm for clustered and sparse systems is not as high as compared to the problems based on random matrices. But it can still be considered as a worthwhile alternative method for such special problems with clustered structures. For example, problem No. 5 is solved correctly and quickly by our method, but cannot be solved by the exhaustive search or bound and decomposition algorithm.

The following Table 10 summarizes the major results related to the performance in terms of accuracy of the proposed algorithm.

Table 10. Accuracy of the proposed algorithm

Problem Type	No. of problems tested	No. of problems g^* found accurately	Accuracy (%)
Random matrices with Non-overlapping Cocircuits (Category 1 and 3)	42	38	90.47
Random matrices with Overlapping Cocircuits (Category 2 and 4)	40	37	92.5
Clustered and Sparse matrices with overlapping cocircuits	6	3	50

The proposed algorithm using the ℓ^1-norm minimization finds the cogirth accurately for most of the random matrices with not very large cogirth. The maximum cogirth that can be accurately determined depends on the size of the matrix. From the results we speculate that overlapping cocircuits have an effect (negative) in determining all of the smallest cocircuits uniquely and suggest further research to analyze this cause. The proposed algorithm is less accurate for matrices with strong deterministic structures

such as a sparse and clustered structure. But it is still a worthwhile alternative even for these special matrices.

CHAPTER V

CONCLUSION

5.1 Conclusions

The aim of the thesis is to develop an algorithm to calculate the redundancy degree of the linear sensor systems. This type of sensor systems has been used in variety of applications including manufacturing, array signal processing, and power plant. Redundancy in sensor system is included to overcome the troublesome of sensor failures and sensor malfunctions to some extent. Redundancy degree is equal to the maximum number of any sensor failures that a system can tolerate and hence recognized as a parameter for measuring the robustness of the sensor system against the complete sensor failures. The proposed algorithm determines the redundancy degree by working on the design matrix of the system with a linear model. The foundation of the algorithm rests on two major techniques – matroid theory and ℓ^0 and ℓ^1-norm minimization problems. First, using the matroid theory the redundancy structure is captured via the vectorial matroid (from the design matrix). The matroid problem equivalent to the measure of redundancy degree is developed. Second, the mathematical formulation for solving this problem is found to be equivalent to a ℓ^0-norm minimization problem. After reviewing the literature reporting closeness between the minimal ℓ^0 and ℓ^1-norm solutions under specific conditions the problem is converted approximately to a ℓ^1-norm minimization problem. The proposed algorithm was written in Matlab. The code used a function from Sparselab package for determining the minimal ℓ^1-norm solution for an underdetermined system of linear equations.

The proposed algorithm was tested on two types of design matrices. First type had sixty-one simulated instances whose design matrices are random and dense. The second type had six instances whose design matrices were sparse and had cluster structure. Superior performance in term of accuracy by the proposed algorithm (ℓ^1-norm

minimization) over the greedy heuristic methods (OMP and StOMP) was evident from the results. This justifies the usage of ℓ^1-norm minimization method even though the computation time was higher. The accuracy of the proposed approximation method was convincingly high with the correct calculation of the redundancy degree for 75 of 82 problems in this type. The largest value of redundancy degree that can be found accurately depends on the number of sensors used in the system. For sparse and clustered matrices, the computation times for the proposed algorithm are much lower than the exhaustive search and bound and decomposition algorithms for all but one matrix. The algorithm was less accurate for the problems containing matrices with sparse and clustered structure. It determined the correct redundancy degree for 3 of 6 problems.

The degree of sensor redundancy can be used to check the failure tolerance capability of the sensor system, compare different network designs quickly, and build a robust system of desired reliability under the given constraints.

5.2 Future Research

There are two main areas which can be considered for the future research. One is related with the conjecture offered earlier about the two thresholds l^* and u^*. Though this seems to be supported by the results obtained from the problems considered in this work, more research has to be done to prove the correctness of this conjecture and if proven the subsequent step could be approximate estimation of them. Another area for the future research could be the study of the performance of the proposed algorithm on design matrices of other types. In this thesis work only two types of design matrices - random and sparse & clustered structures are tested. The determination of the possible reasons for comparatively lower performance, in terms of accuracy of δ, of the proposed algorithm on the design matrices with sparse & clustered and other special structures could be another interesting area for the future research.

APPENDIX A

SIMULATED DATA SETS GENERATION METHOD

As mentioned earlier four different categories of problem instances are generated to control the problem characteristics in our testing process. The procedure adopted for the generation is quite similar for all of them. We first generate a matrix, which can be considered equivalent to the transposed measurement matrix of required size (say p x n with $p < n$) with elements coming (independently) from a uniform distribution (0,100). Then the predetermined list of cocircuits can be implanted in it by systematically modifying some elements of the matrix. This procedure is explained below. The cocircuits list is recorded for the analyzing the accuracy of the proposed algorithm. For simulated data sets used in this work the cocircuit list is presented in Appendix C.

Procedure for implanting cocircuits into a matrix of r rows and n columns

The first cocircuit in the list is selected and the following steps are repeated until we reach the end of the list:

1. Let row set = (1, 2, . . , r)
2. The 'compliment set' of the current cocircuit is determined
3. One of the two options is selected randomly and executed:
    i) Option1:
        (a) A row is picked from row set randomly
        (b) Make all the elements of H^T located in the selected row and the compliment set's columns to "zero"
    ii) Option2:
        (a) A row (other than the first or last element) is picked randomly from the row set.

(b) Make all the elements located in the above picked row and in the compliment set columns to be twice the sum of the elements located exactly one row above and below.

4. Next the rows (one row in option 1 and three rows in option 2) that were involved in the above operation is/ are removed from row set. This is done to ensure the final matrix has all the cocircuits implanted correctly.
5. If all the cocircuits are inserted, Stop; else select the next cocircuit in the list and go to Step2.

APPENDIX B

MATLAB CODE

This section contains the Matlab code for the proposed algorithm and the code used to generate problem instances random category. The code for the proposed algorithm is broken into 5 functions/ Matlab files. Their names and purpose is as follows,

B1. SolveCogirth.m - Main file to be executed; reads data, calls other files and displays results.

B2. DualMatroidRepMatrix.m - Generates standard representative matrix of the dual matroid

B3. circuitsByL1Norm.m - Determines the circuit of the dual matroid

B4. cocircuitByColumnMatch - Identifies the cocircuits of the original matroid

B5. MatrixCreation.m - For generating the problem instances of random category

B1. File Name: *SolveCogirth.m* (Main File)

```
clc
clear

%  Reading Data: Matrix must have no. of rows >= no. of columns
Mat = Read Data File from Source

tic                                    % Start Timer
Matrix = rref(transp(Mat));
[Matrix_rows,Matrix_cols]=size(Matrix);

 %  Standard Representative Matrix of the Dual Matroid and the original Columns
[M, Actual_Cols] = DualMatroidRepMatrix(Matrix);

%% Function finds the Smallest Circuits of the above matrix M
circuits_DualMatroid = circuitsByL1Norm(M);
```

```
 %  Column matching to find the Smallest Cocircuits of the original Matroid
cociruits_OrigMatroid = cocircuitByColumnMatch(circuits_DualMatroid, Actual_Cols);

%  Print the Cocircuits, Size and Computation Time
 cociruits_OrigMatroid
[No_SmallestCocircuits, Cogirth_Size] = size(cociruits_OrigMatroid)
Computation_Time = toc
```

B2. File name: *DualMatroidRepMatrix.m*

```
function [M, New_Cols] = DualMatroidRepMatrix(Matrix)

%  DualMatroidRepMatrix : Determines the Standard Representative Matrix of the Dual
%                                 Matroid
%
%  Usage
%    [sol_Matrix, sol_actualColumns] = DualMatroidRepMatrix(M)
%  Input
%    Matrix         Standard Representative Matrix of the Original
%                   Matroid
%  Output
%     M             Standard Representative Matrix of the Dual
%                   Matroid
%    New_Cols      Vector containing the columns arrangement after
%                   the transformation
%
%  The function first determines the Standard Representative Matrix of the Original
%  Matroid by shifting the columns accordingly such that the matrix is of form
%                [I_(r) | D]
%  Next the Std. Rep Matrix of the Dual Matroid is determined by transforming the
%  matrix to form
%               [-transp(D) | I_(n-r)]
%    where I is an identity matrix, r - no. of rows, n - no. of columns


[Matrix_rows,Matrix_cols] = size(Matrix);

New_Cols = 1 : Matrix_cols;
```

```
%  |----< Rearranging columns to get matrix of form I|D >-----|

I_track = 1;                    % Keep track of the columns of Identity matrix of I|D

for r = 1 : Matrix_rows
   r;
   I_track;
   New_Cols;

   [I_track_rows, I_track_cols] = size(I_track);

   if r == 1
      I  = Matrix(r,r);
   else
      next_col = I_track_cols + 1;
      for j = 1 : I_track_cols
         I_part1(1:r,j) = Matrix(1:r,I_track(1,j));
      end;

      I_part2 = Matrix(1:r,New_Cols(1,r));
      I = horzcat(I_part1, I_part2);
   end;

   if I == eye(r)
      I_track(1,r) = r;
   else
      initial = r;
      for i = initial+1 : Matrix_cols
         [I_track_rows, I_track_cols] = size(I_track);

         for j = 1 : I_track_cols
            I_part1(:,j) = Matrix(1:r,I_track(1,j));
         end;

         I_part2 = Matrix(1:r,i);

         I = horzcat(I_part1, I_part2);

         if I == eye(r)
            I_track(1,r) = i;

            new_temp = New_Cols(1,r);
            New_Cols(1,r) = i;
            New_Cols(1,i) = new_temp;
```

```
            break;
          end;
        end;
    end;
end;
Matrix_rank = rank(Matrix);

for i = Matrix_rows : -1: Matrix_rank
    if sum(Matrix(i,:)) == 0
        Matrix(i,:) = [];
    end;
end;
Matrix;
[Matrix_rows, Matrix_cols] = size(Matrix);

n = 1;
for i = (Matrix_rows + 1) : Matrix_cols
    part_D(:,n) = Matrix(:,New_Cols(1,i));
    n = n + 1;
end;
part_D;

n = 1;
for i = 1 : Matrix_cols
    ID_Mat(:,n) = Matrix(:,New_Cols(1,i));
    n = n + 1;
end;
ID_Mat;

%  |---< Transform to get matrix of form [-transp(D) | I_(n-r)]>---|

part_I = eye(Matrix_cols - Matrix_rows);
trans_D=transp(part_D);

M = horzcat(-1*trans_D , part_I);
New_Cols;
```

B3. File Name: *circuitsByL1Norm.m*

```
function [Circuits] = circuitsByL1Norm(M)

%  circuitsByL1Norm: Finds the smallest Circuits of a Dual
%                Matroid
%  Usage
%    sol = circuitsByL1Norm(M)
%  Input
%    M             Standard Representative Matrix of the Dual Matroid
%  Output
%    Circuits     Smallest Circuits of the Dual Matroid
%
%  Actual Problem: min ||x||_0 s.t. A*x = 0
%  Approach: L0_Norm -> L1_Norm -> 'n' L1_Norm problems
%
%  Details: Problem converted into
%          min ||x_(-i)||_1
%          s.t. A(-i) * x(-i) = A(i)  for i = 1 to n
%      where 'n' is the number of columns in M and 'i' is the
%  index of a single column
%
%  Chooses the values of i's if x(-i) has least L0_norm - from
%  which the Circuits of the Dual Matroid is determined

[M_Rows, M_Cols] = size(M);

for i = 1 : M_Cols
   b = M(:,i);
   A_firstPart = M(:,1:i-1);
   A_secondPart = M(:,i+1:M_Cols);
   A = horzcat(A_firstPart, A_secondPart);

   x_i = SolveBP(A,b,(M_Cols-1),20,0,0);
   % Temporary x_i holds the the l1-norm solution for when co-eff of i-th column = -1

   abs_xi = abs(x_i);
   X{i,1}(:,1) = abs_xi;       Cell 'X' hold all the 'x' values
   l0Norm_X(i,1) = length(find(abs_xi > .001));
   % Number of non-zero elements in each 'x' is recorded
end;

min_l0Norm = min(l0Norm_X);                  % Minimum size is found
```

```
 n = 1;
   for i = 1 : M_Cols                          % Column number corresponding to the least
      if l0Norm_X(i,1) == min_l0Norm           %   circuit is found and added later
         min_l0Norm_loc(n,1) = i;
         n = n + 1;
      end;
   end;
[Circuits_Nos, i_col] = size(min_l0Norm_loc);

n = 1;
for j = 1 : Circuits_Nos
    temp_X{j,1} = X{(min_l0Norm_loc(j,1)),1};
    CV(n,:) = transp(find(temp_X{j,1} > .001));
    n = n + 1;
end;

[CV_Rows, CV_Cols] = size(CV);

%  Adjusting the column numbers of the Circuits w.r.t. to the identifying column number
n = 1;
for i = 1 : CV_Rows
  for j = 1 : CV_Cols
   if (CV(i,j) < min_l0Norm_loc(i,1))
         circuit(n,j) = CV(i,j);

   else if (CV(i,j) >= min_l0Norm_loc(i,1))
         circuit(n,j) = (CV(i,j) + 1);
       end;
   end;
  end;
  n = n + 1;
end;

%  Complete Circuits after adding the column that generated a solution in X with
%  minimum l0-norm
circuit(:,j+1) = min_l0Norm_loc(:,1);

% Remove duplicates: First sort each row, then select unique rows
for i = 1 : Circuits_Nos
   sorted_circuit(i,:) = sort(circuit(i,:));
end;

Circuits = unique(sorted_circuit, 'rows');
[Circuits_Rows, Circuits_Cols] = size(Circuits);
```

B4. File Name: *cocircuitByColumnMatch.m*

```
function [Cocircuits] = cocircuitByColumnMatch(circuits_DualMatroid, Actual_Cols)

%  Matching the columns in the Circuits with New_Cols to determine the
%  required Cocircuits of the original Matroid

[Circuits_Nos, Circuits_Size] = size(circuits_DualMatroid)
for i = 1 : Circuits_Nos
   for j = 1 : Circuits_Size
      CoCir(i,j) = Actual_Cols(1,circuits_DualMatroid(i,j));
   end
end;

for i = 1 : Circuits_Nos
   Cocircuits(i,:) = sort(CoCir(i,:));
end;
```

B5. File Name: *MatrixCreation.m*

```
clc
clear
no_rows = {enter number of rows of H – variables in the sensor system}
no_cols = {enter number of columns of H – sensors in the system}

Mat = unifrnd(0,100,no_rows,no_cols);
[Mat_rows, Mat_cols] = size(Mat);
 A = 1 : Mat_cols;
B = 1 : Mat_rows;

% Replace the vector c1, c2, . . by cocircuits that are to be inserted transpose(H)
CoCir_list{1,1} = c1;
CoCir_list{2,1} = c2;
CoCir_list{3,1} = c3;
% ..
% …. continue the list for the required number of cocircuits
 [CoCir_list_rows, CoCir_list_cols] = size(CoCir_list)
```

```
for i = 1 : CoCir_list_rows
        current_cocir_size = length(CoCir_list{i,1})

         [B_rows, B_cols] = size(B)
        option = unidrnd(2)             % Equal chance given two options of implanting a
                                        % cocircuit explained in Appendix A. #of rows
                                        % should be greater than or equal to number of co-
                                        % circuits put in.

        if option == 1
                pres_cir = CoCir_list{i,1};
                compl_pres_cir = setdiff(A,pres_cir);
                row_index = unidrnd(B_cols);
                row = B(row_index)
                Mat(row,compl_pres_cir) = 0;

        else if option == 2
                pres_cir = CoCir_list{i,1};
                compl_pres_cir = setdiff(A,pres_cir);
                row_index = unidrnd(B_cols-2) + 1;
                row = B(row_index)
                Mat(row,compl_pres_cir) = 2*Mat(B(row_index+1),compl_pres_cir) +
                                        2*Mat(B(row_index-1),compl_pres_cir);
                row = [B(row_index-1) row B(row_index+1)];
             end;
        end;

        B = setdiff(B,row);             %  Banning the selected rows from future selection
end;

Mat
csvwrite("Path specifying the location in System to write the file +  file_name");
```

APPENDIX C
SIMULATED DATA SETS DETAILS

Details of the data sets of four categories namely 1) unique and single cocircuit, 2) single smallest cocircuit, 3) multiple but non-overlapping cocircuits and 4) over-lapping cocircuits are shown in tables C1 – C4 respectively. For the problem instances in category 1 and 3, only the smallest cocircuits present in the matrix are shown, while for those in category 2 and 4 all the cocircuits are shown. The cocircuits/ coindependent sets that are not equivalent to the smallest size are preceded with a '-' symbol. SCC stands for smallest cocircuits.

Table C1. Category 1: Single and Unique Cocircuit Problem Instances Details

No	Matrix Size	Co-Girth Size	# SCC	Cocircuits
1	12 x 25	5	1	[2 4 6 8 10]
		7	1	[2 4 6 8 10 12 14]
		8	1	[2 4 6 8 10 12 14]
2	30 x 70	5	1	[31 33 35 37 39]
		8	1	[31 33 35 37 39 41 43 45]
		12	1	[31 33 35 37 39 41 43 45 47 49 51 53]
3	60 x 150	7	1	[91 92 93 94 95 96 97]
		11	1	[91 92 93 94 95 96 97 99 101 103 105]
		15	1	[91 92 93 94 95 96 97 99 101 103 105 107 108 109 110]
4	60 x 220	7	1	[1 11 21 31 41 51 61]
		12	1	[1 11 21 31 41 51 61 71 81 91 101 111]
		16	1	[1 11 21 31 41 51 61 71 81 91 101 111 121 131 141 151]
5	150 x 325	7	1	[1 51 101 151 201 251 301]
		13	1	[1 51 101 151 201 251 301 303 305 307 309 311 313]
		17	1	[1 51 101 151 201 251 301 303 305 307 309 311 313 315 317 319 321]
6	200 x 500	8	1	[71 73 75 171 173 175 271 273]
		16	1	[71 73 75 171 173 175 271 273 275 371 373 375 471 473 475 500]
		21	1	[71 73 75 171 173 175 271 273 275 371 373 375 471 473 475 500 51 52 53 54 55]
7	250 x 1000	10	1	[10 20 30 40 50 110 120 130 140 150]
		20	1	[10 20 30 40 50 110 120 130 140 150 310 320 330 340 350 410 420 430 440 450]
		30	1	[10 20 30 40 50 110 120 130 140 150 310 320 330 340 350 410 420 430 440 450 610 620 710 720 810 820 910 920 950 999]

Table C2. Category2: Single Smallest Cocircuit Problem Instances Details

No	Matrix Size	Co-Girth	# SCC	Cocircuits
1	12 x 25	5	1	[2 4 6 8 10] -[1 3 5 7 9 11] -[13 15 17 19 21]
		7	1	[2 4 6 8 10 12 14] -[1 2 3 4 5 6 7 8] -[10 11 12 13 14 15 16 17]
2	30 x 70	7	1	[31 33 35 37 39 41 43] -[1 2 3 4 5 6 7 8] -[1 2 3 31 33 35 37 39] -[51 52 53 54 55 37 39 41 43]
		10	1	[31 33 35 37 39 41 43 45 47 49]; -[1 2 3 4 5 6 7 8 9 10 11] -[1 2 3 4 5 31 33 35 37 39 51 52] -[51 52 53 54 55 37 39 41 43 1 2]
3	60 x 150	8	1	[1 5 10 15 20 25 30 35] -[1 5 10 15 101 102 103 104 105] -[15 20 61 62 63 64 65 103 104 105] -[51 52 53 54 20 25 61 62 105]
		12	1	[1 5 10 15 20 25 30 35 40 45 50 55] -[1 5 10 15 101 102 103 104 105 141 142 143 144] -[15 20 61 62 63 64 65 103 104 105 50 55 70 71] -[51 52 53 54 20 25 61 62 105 70 50 55 149]
4	60 x 220	9	1	[1 11 21 31 41 51 61 71 81] -[1 11 21 31 151 152 153 154 155 156] -[31 41 51 151 152 201 202 203 204 205] -[1 31 51 71 151 154 156 200 202 204 205]

Table C2. Continued

4	60 x 220	14	1	[1 11 21 31 41 51 61 71 81 91 101 111 121 131] [1 11 21 31 41 151 152 153 154 155 156 157 158 159 160] [31 41 51 61 151 152 153 201 202 203 204 205 206 207 159 160] [1 31 51 71 151 154 156 200 202 204 205 158 159 81 200 201]
5	150 x 325	9	1	[1 11 21 31 41 51 61 71 81] [1 11 21 31 151 152 153 154 155 156] [31 41 51 151 152 201 202 203 204 205] [1 31 51 71 151 154 156 200 202 204 205]
		14	1	[1 11 21 31 41 51 61 71 81 91 101 111 121 131] [1 11 21 31 41 151 152 153 154 155 156 157 158 159 60] [31 41 51 61 151 152 153 201 202 203 204 205 206 207 159 160] [1 31 51 71 151 154 156 200 202 204 205 158 159 81 200 201]
		17	1	[1 51 101 151 201 251 301 303 305 307 309 311 313 315 317 319 321] [1 51 101 151 315 319 141 142 143 144 145 146 147 148 149 150 161 162] [210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228] [301 303 305 307 210 211 212 213 214 144 145 148 227 228 271 272 273 51]
6	200 x 500	12	1	[21 22 23 24 25 26 27 28 29 30 31 32] [51 52 53 54 55 56 57 58 59 60 61 62 63] [71 72 73 74 75 22 23 24 25 26 151 152 153] [200 201 202 203 31 32 62 63 64 65 66 152 153 154] [300 301 302 303 71 72 73 74 75 64 65 66 315 316] [451 452 453 454 22 23 24 200 201 300 301 315 316]
		18	1	[21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38] [51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69] [71 72 73 74 75 22 23 24 25 26 28 29 38 151 152 153 154 155 156] [200 201 202 203 204 205 305 31 32 33 34 62 63 64 65 66 152 153 154 155] [300 301 302 303 71 72 73 74 75 64 65 66 67 18 21 22 23 315 316] [451 452 453 454 22 23 24 31 32 33 200 201 202 300 301 302 315 316 64 65]

Table C2. Continued

7	250 x 1000	15	1	[10 20 30 40 50 110 120 130 140 150 210 220 230 240 250] [10 20 30 40 50 310 320 330 340 350 410 420 430 440 450 460] [110 120 130 140 350 410 420 430 440 510 520 530 540 550 560 570] [600 610 620 630 640 650 660 670 680 690 700 710 720 730 740 750 760] [600 610 620 810 820 830 840 850 860 870 880 110 120 130 140 510 520] [10 30 110 130 210 230 710 720 800 810 900 910 920 930 350 450]
		25	1	[10 20 30 40 50 110 120 130 140 150 310 320 330 340 350 410 420 430 440 450 610 620 710 720 810] [10 20 30 40 50 310 320 321 322 323 324 325 326 327 328 329 421 422 423 424 425 426 427 428 429 430] [110 120 130 140 150 610 620 421 422 423 424 500 501 502 503 504 511 512 513 514 515 611 612 613 614 615] [410 420 430 440 450 710 810 910 911 912 913 914 915 916 321 322 323 324 325 421 422 423 612 613 614 615 616] [710 720 810 501 502 503 504 505 613 614 615 616 911 912 913 914 915 916 410 420 430 440 441 442 443 444]

Table C3. Category3: Non-Overlapping Cocircuits Problem Instances Details

No	Matrix Size	Co-Girth	# SCC	Cocircuits
1	12 x 25	5	3	[1, 3, 5, 7, 9] [11, 12, 13, 14, 15] [2, 8, 16, 24, 25]
		8	2	[1, 3, 5, 7, 9, 11, 13, 15] [17, 18, 19, 20, 21, 22, 23, 24]

Table C3. Continued

2	30 x 70	5	4	[1, 11, 21, 31, 41] [5, 15, 25, 35, 45] [7, 17, 27, 37, 47] [9, 19, 29, 39, 49]
		8	4	[1, 11, 21, 31, 41, 51, 61, 62] [5, 15, 25, 35, 45, 55, 65, 66] [7, 17, 27, 37, 47, 57, 67, 68] [9, 19, 29, 39, 49, 59, 69, 70]
3	60 x 150	5	4	[1, 11, 21, 31, 41] [5, 15, 25, 35, 45] [7, 17, 27, 37, 47] [9, 19, 29, 39, 49]
		8	4	[1 11 21 31 41 51 61 71] [5 15 25 35 45 55 65 75] [7 17 27 37 47 57 67 77] [9 19 29 39 49 59 69 79] [101 102 103 104 105 106 107 108 109 110] [111 112 113 114 115 116 117 118 119 120] [122 123 124 125 126 127 128 129 130 131 132]
4	60 x 220	5	7	[1 11 21 31 41] [5 15 25 35 45] [7 17 27 37 47] [9 19 29 39 49] [101 102 103 104 105] [111 121 131 141 151] [156 157 158 159 160]

Table C3. Continued

4	60 x 220	10	10	[1 11 21 31 41 51 61 71 81 91] [5 15 25 35 45 55 65 75 85 95] [7 17 27 37 47 57 67 77 87 97] [9 19 29 39 49 59 69 79 89 99] [101 102 103 104 105 106 107 108 109 110] [111 121 131 141 151 161 171 181 191 192] [132 133 134 135 136 137 138 139 140 142] [201 202 203 204 205 206 207 208 209 210] [193 194 195 196 197 198 199 200 112 122] [3 13 23 33 43 53 63 93 113 123]
		15	4	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15] [21 22 23 24 25 26 27 28 29 30 31 32 33 34 35] [71 75 73 74 75 76 77 78 79 80 81 82 83 84 85] [101 102 103 104 105 106 107 108 109 110 111 112 113 114 115]
5	150 x 325	5	7	[1 11 21 31 41] [5 15 25 35 45] [7 17 27 37 47] [9 19 29 39 49] [101 102 103 104 105] [111 121 131 141 151] [156 157 158 159 160]
		10	9	[1 11 21 31 41 51 61 71 81 91] [5 15 25 35 45 55 65 75 85 95] [7 17 27 37 47 57 67 77 87 97] [9 19 29 39 49 59 69 79 89 99] [101 102 103 104 105 106 107 108 109 110] [111 121 131 141 151 161 171 181 191 192] [132 133 134 135 136 137 138 139 140 142] [201 202 203 204 205 206 207 208 209 210] [211 221 231 241 251 301 311 321 322 323]

Table C.3 Continued

5	150 x 325	15	7	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15] [21 22 23 24 25 26 27 28 29 30 31 32 33 34 35] [71 72 73 74 75 76 77 78 79 80 81 82 83 84 85] [101 102 103 104 105 106 107 108 109 110 111 112 113 114 115] [221 222 223 224 225 226 227 228 229 230 231 232 233 234 235] [251 252 253 254 255 256 257 258 259 260 261 262 263 264 265] [301 302 303 304 305 306 307 308 309 310 311 312 313 314 315] [51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66] [150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166] [200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218]
6	200 x 500	10	7	[1 2 3 4 5 6 7 8 9 10] [21 22 23 24 25 26 27 28 29 30] [71 72 73 74 75 76 77 78 79 80] [101 102 103 104 105 106 107 108 109 110] [221 222 223 224 225 226 227 228 229 230] [251 252 253 254 255 256 257 258 259 260] [301 302 303 304 305 306 307 308 309 310]
		15	7	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15] [21 22 23 24 25 26 27 28 29 30 31 32 33 34 35] [71 72 73 74 75 76 77 78 79 80 81 82 83 84 85] [101 102 103 104 105 106 107 108 109 110 111 112 113 114 115] [221 222 223 224 225 226 227 228 229 230 231 232 233 234 235] [251 252 253 254 255 256 257 258 259 260 261 262 263 264 265] [301 302 303 304 305 306 307 308 309 310 311 312 313 314 315]
		20	7	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20] [21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40] [71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90] [101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120] [221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240] [251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270] [301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320]

Table C.3 Continued

7		250 x 1000	17	10	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17] [51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67] [101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117] [151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167] [200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216] [251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267] [301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317] [351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367] [400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416] [451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467] -[501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518] -[651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668] -[700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718] -[851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868] -[901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919]

Table C4. Category4: Overlap Cocircuit Problem Instances Details

No	Matrix Size	Co-Girth	# SCC	Cocircuits
1	12 x 25	5	4	[1 2 3 4 5] [1 2 8 9 10] [3 4 11 12 13] [1 11 12 16 25] -[12 13 14 15 16 17] -[9 10 13 16 21 22]

Table C4. Continued

2	30 x 70	5	4	[1 2 3 4 5] [51 52 53 54 55] [1 2 3 31 32] [31 32 33 53 54] [18 19 20 31 32] [25 26 27 54 55] -[25 26 27 28 29 30 31] -[41 42 43 51 52 27 28] -[36 37 38 54 55 56] -[41 42 1 2 9 10]
		8	4	[1 2 3 4 5 6 7 8] [1 2 21 22 23 24 25 26] [4 5 24 25 54 55 56 57] [7 8 21 22 54 55 61 62] -[11 12 13 14 15 16 17 18 19 20] -[17 18 31 32 33 34 35 36 54 55] -[24 25 11 12 17 34 35 65 66 67]
3	60 x 150	5	9	[1 2 3 4 5] [1 2 51 52 53] [21 22 23 24 25] [23 24 25 71 72] [1 2 51 52 72] [81 82 83 101 102] [120 121 122 123 124] [101 102 121 122 141] [122 141 142 83 124] -[23 24 91 92 122 141] -[120 121 95 96 97 141 142] -[4 5 12 13 14 15]

Table C4. Continued

3	60 x 150	8	7	[1 2 3 4 5 6 7 8] [1 2 51 52 53 54 55 56] [21 22 23 24 25 26 27 28] [23 24 25 71 72 73 74 75] [1 2 51 52 72 5 54 73] [81 82 83 101 102 84 52 104] [120 121 122 123 124 125 23 24] -[101 102 121 122 141 142 98 99 100 105] -[122 141 142 83 124 102 105 106 107 108] -[23 24 91 92 122 141 95 96 124 13]
4	60 x 220	8	7	[1 2 3 4 5 6 7 8] [1 2 3 4 51 52 53 54] [1 6 7 53 101 102 103 104] [121 122 123 124 125 126 127 128] [4 5 52 53 102 103 125 126] [160 161 162 163 53 102 6 7] [170 171 172 173 123 124 21 22] -[180 181 182 31 32 21 161 162 128] -[190 191 170 171 180 181 200 201 220] -[140 141 4 5 104 161 162 200 102 110] -[141 142 143 144 145 146 147 148 149 150] -[145 147 150 152 153 154 155 156 157 1]
5	150 x 325	14	7	[1 2 3 4 5 6 7 8 9 10 11 12 13 14] [1 2 3 4 31 32 33 34 35 101 102 103 104 105] [5 6 7 8 51 52 53 54 55 111 112 113 114 115] [31 33 104 53 54 113 151 152 153 154 155 200 201 202] [70 71 72 73 74 75 151 152 153 154 80 81 82 83] [170 171 172 173 174 111 112 113 114 80 81 82 1 2] [300 301 302 303 304 121 122 123 73 74 75 81 13 14] -[300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315] -[145 146 147 151 152 154 155 170 171 172 315 316 51 52 53 111 112] -[1 2 3 4 5 80 81 82 83 84 307 308 309 310 311 320 321]

Table C4. Continued

5	150 x 325	14	7	[1 2 3 4 5 6 7 8 9 10 11 12 13 14] [1 2 3 4 31 32 33 34 35 101 102 103 104 105] [5 6 7 8 51 52 53 54 55 111 112 113 114 115] [31 33 104 53 54 113 151 152 153 154 155 200 201 202] [70 71 72 73 74 75 151 152 153 154 80 81 82 83] [170 171 172 173 174 111 112 113 114 80 81 82 1 2] [300 301 302 303 304 121 122 123 73 74 75 81 13 14] -[300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315] -[145 146 147 151 152 154 155 170 171 172 315 316 51 52 53 111 112] -[1 2 3 4 5 80 81 82 83 84 307 308 309 310 311 320 321]
6	200 x 500	10	8	[1 2 3 4 5 6 7 8 9 10] [51 52 53 54 55 56 57 58 59 60] [101 102 103 104 105 106 107 108 109 110] [1 2 3 56 57 58 107 108 109 131] [108 109 131 161 162 163 8 9 191 192] [191 192 251 252 253 254 255 52 53 54] [52 53 301 302 303 304 305 320 321 322] [303 304 305 8 9 10 400 401 402 403] -[1 2 3 4 56 57 58 59 301 302 303 304] -[451 452 453 454 200 201 202 105 106 107 108 109] -[471 472 473 190 191 192 193 4 5 6 7 8] -[50 51 52 53 54 450 451 452 471 472 473]
7	250 x 1000	17	10	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17] [1 2 3 4 31 32 33 34 35 51 52 53 54 55 56 57 58] [1 2 3 4 151 152 153 154 202 501 502 503 504 551 552 553 554] [7 8 9 10 11 101 102 103 151 152 153 154 155 201 202 203 204] [11 12 13 14 53 54 55 56 100 101 102 103 104 105 200 201 202] [11 12 13 14 55 151 152 153 154 301 302 303 304 701 702 703 704] [16 17 31 32 33 34 55 56 57 101 102 103 300 301 302 303 304] [16 17 51 52 53 54 351 352 353 354 355 356 400 401 402 403 404] [101 102 103 104 105 351 352 353 354 355 601 602 603 604 605 651 652] [501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517]

Table C4. Continued

7	250 x 1000	17	10	-[751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768] -[800 801 802 803 804 11 12 13 14 151 152 153 154 301 302 303 304 515 516] -[400 401 402 403 404 551 552 553 554 101 102 103 104 105 701 702 703 704] -[351 352 353 354 6 7 8 9 10 31 32 33 34 54 55 56 57 58] -[751 752 753 754 755 800 801 802 803 804 505 506 507 508 509 510 101 102 103 104]

REFERENCES

Ali, Y. and Narasimhan, S. (1993), "Sensor Network Design for Maximizing Reliability of Linear Process," *AIChE Journal*, Vol. 39, No. 5, pp. 820-828.

Ali, Y. and Narasimhan, S. (1995), "Redundant Sensor Network Design for Linear Process," *AIChE Journal*, Vol. 41, No. 10, pp. 2237-2249.

Apley, D. W. and Shi. J. (2001), "A Factor-Analysis Method for Diagnosing Variability in Multivariate Manufacturing Processes," *Technometrics*, Vol. 43, No. 1, pp. 84-95.

Bagajewicz, M. and Sanchez, M.C. (1999), "Design and Upgrade of Nonredundant and Redundant Linear Sensor Networks," AIChE Journal, Vol. 55, No. 9, pp. 1927-1938.

Candes, E. T. and Tao, T. (2005a), "Decoding by Linear Programming," *IEEE Transactions on Information Theory*, Vol. 54, No. 11, pp. 4203-4215.

Candes, E., Romberg, J. and Tao, T. (2005b), "Stable Signal Recovery from Incomplete and Inaccurate Measurements," *Communications on Pure Appl. Math,* Vol. 59, No. 8, pp. 1207-1223.

Chen, S. S., Donoho, D. L. and Saunders, M. A. (1998), "Atomic Decomposition by Basis Pursuit," *SIAM Journal on Scientific Computing*, Vol. 20, pp. 33-61.

Chen, Y. (2006), "Application of Matroid Theory for Diagnosability Study of Coordinate Sensing Systems in Discrete-Part Manufacturing Processes," *Technometrics*, Vol. 48, No. 3, pp. 386-397.

Cho, J.J., Chen, Y. and Ding, Y. (2007), "On the (Co)Girth of Connected Matroids," *Discrete Applied Mathematics*, Vol. 155, pp. 2456-2470.

Cho, J.J., Chen, Y. and Ding, Y. (2009a), "Calculating the Breakdown Point Condition of Sparse Linear Models," *Technometrics,* Vol. 51, pp. 34-46

Cho, J. J., Ding, Y., Chen, Y. and Tang, J. (2009b), "Robust Calibration for Localization in Clustered Wireless Sensor Networks," *IEEE Transactions on Automation Science and Engineering*, in-press.

Cho, J.J., Chen, Y. and Ding, Y. (2004), "Redundancy Analysis of Linear Sensor Systems and Its Applications," unsubmitted manuscript.

Conforti, M. and Rao, M.R. (1987), "Some New Matroids on Graphs: Cut Sets and the Max Cut Problem," *Mathematics of Operations Research*, Vol. 12, No. 2, pp. 193-204.

Ding, Y., Shi, J. and Ceglarek, D. (2002), "Diagnosability Analysis of Multi-Station Manufacturing Processes," *Journal of Dynamic Systems, Measurements and Control*, Vol. 124, No. 1, pp. 1-13.

Donoho, D. L. (2006a), "For Most Large Underdetermined Systems of Equations the Minimal ℓ^1-norm Near-solution Approximates the Sparsest Near-solution," *Communications on Pure Appl. Math,* Vol. 59, No. 7, pp. 907-934.

Donoho, D. L. (2006b), "For Most Large Underdetermined Systems of Linear Equations the Minimal ℓ^1-norm Solution is also the Sparsest Solution," *Communications on Pure Appl. Math,* Vol. 59, No. 6, pp. 797–829.

Donoho, D. L. (2006c), "Compressed Sensing," *IEEE Transactions on Information Theory*, Vol. 52, No. 4, pp. 1289-1306.

Donoho, D. L., Tasig, Y., Drori, I. and Starck, J. (2007), "Sparse Solution to Underdetermined Linear Equations by Stagewise Orthogonal Matching Pursuit," *IEEE Transactions on Information Theory, submitted*, Available at http://sparselab.stanford.edu/sparseLab_files/ local_files/StOMP.pdf.

Donoho, D.L. and Elad, M. (2006d), "On the Stability of the Basis Pursuit in the Presence of Noise," *Signal Processing*, Vol. 86, pp. 511-532.

Donoho, D.L. and Tsiag, Y. (2008), "Fast Solution of ℓ^1-norm Minimization Problems When the Solution May be Sparse," *IEEE Transactions on Information Theory*, Vol. 54, No. 11, pp. 4789-4812.

Dorr, R., Kratz, F., Ragot, J., Loisy, F. and Germain, J. (1997), "Detection, Isolation, and Identification of Sensor Faults in Nuclear Power Plants," *IEEE Transactions on Control Systems Technology*, Vol. 5, No. 1, pp. 42-60.

Elad, M. and Bruckstein, A.M. (2002), "A Generalized Uncertainty Principle and Sparse Representations in Pairs of Bases," *IEEE Transactions on Information Theory*, Vol. 49, pp. 2558 – 2567

Jin, J. and Shi, J. (1999), "State Space Modeling of Sheet Metal Assembly for Dimensional Control," *Journal of Manufacturing Science and Engineering*, Vol. 121, pp. 756-762.

Khachiyan, L., Boros, E., Elbassioni, K., Gurvich, V. and Makino, K. (2005), "On the Complexity of Some Enumeration Problems for Matroids," *SIAM J. Discrete Math*, Vol. 19, No. 4, pp. 966–984.

Khan, A., Ceglarek, D. and Ni, J. (1998), "Sensor Location Optimization for Fault Diagnosis in Multi-Fixture Assembly Systems," *Journal of Manufacturing Science and Engineering*, Vol. 120, pp. 781-792.

Lee, J. and Ryan, J. (1992), "Matroid Applications and Algorithms," *ORSA Journal of Computing*, Vol. 4, No. 1, pp. 70-98.

Luong, M., Maquin, D., Huynh, C.T. and Ragot, J. (1994), "Observability, Redundancy, reliability and Integrated Design of Measurement Systems," *IFAC Symp. On Intelligent Components and Instrument Control Applications,* Budapest, Hungary.

Madron, F. and Veverka, V. (1992), "Optimal Selection of Measuring Points in Complex Plants by Linear Models," *AIChE Journal*, Vol. 38, No. 2, pp. 227-236.

Mili, L., Phaniraj, V., and Rousseeuw, P.J. (1990), "High Breakdown Point Estimation in Electric Power Systems," *IEEE International Symposium on Circuits and Systems*, Vol. 3, pp. 1843-1846.

Nieto, J. A. and Marin, M. C. (2000), "Matroid theory and Chern–Simons," *Journal of Mathematical Physics*, Vol. 41, No. 12, pp.7997-8005.

Oxley, J., G., ed. 1992. *Matroid Theory*. Oxford University Press, NewYork

Pansoo, K. and Ding, Y. (2004), "Optimal Design of Fixture Layout in Multistation Assembly Process," *IEEE Transactions on Automation Science and Engineering*, Vol. 1, No. 2, pp. 133-145.

Sidiropoulos, N., D., Bro, R., and Giannakis, G., B. (2000), "Parallel Factor Analysis in Sensor Array Processing," *IEEE Transactions on Signal Processing*, Vol. 48, No. 8, pp. 2377-2388

Seymour, P., D. (1977), "The Matroids with the Max-Flow Min-Cut Property," *Journal of Combinatorial Theory*, Series B, Vol. 23, pp. 289-222.

Staroswiecki, M., Hoblos, G. and Aitouche, A. (2004), "Sensor Network Design for Fault Tolerant Estimation," *International Journal of Adaptive Control and Signal Processing*, Vol. 18, pp. 22-77.

Tropp, J., A. and Gilbert, A., C. (2007) "Signal Recovery from Random Measurements via Orthogonal Matching Pursuit". *IEEE Transactions on Information Theory*, Vol. 53, No. 12, pp. 4655-4666

Truemper, K. (1987), "Max-Flow Min-Cut Matroids: Polynomial Testing and Polynomial Algorithms for Maximum Flow and Shortest Routes," *Mathematics of Operations Research*, Vol. 12, No. 1, pp. 72-96.

Whitney, H. (1935), "On the Abstract Properties of Linear Dependence," *American Journal of Mathematics*, Vol. 57, No. 3, pp. 509-533.