Degree of Redundancy of Linear Systems using Implicit Set Covering

John D. Arellano, and Illya V. Hicks *

Abstract

In this paper, we present a set covering problem (SCP) formulation to compute the degree of redundancy of linear systems. Computing the degree of redundancy of a linear system allows for the evaluation of the quality of the sensor network. The formulation is equivalent to solving the linear matroid cogirth problem, finding the cardinality of the smallest cocircuit of a matroid. We also discuss existing methods developed to solve the matroid cogirth problem and the SCP. Computational results are provided to validate a branch-and-cut algorithm that addresses the SCP formulation.

Note to Practitioners—Automated systems such as sensor networks have begun to play a larger role in various aspects of our daily lives in recent years. With this, the ability to observe and measure the reliability of such networks has become increasingly important. There exist several methods to measure the reliability of the network. One way to measure the reliability of the network is to consider the degree of redundancy of the network. We present a simple, competitive method to compute the degree of redundancy of a sensor network based on a simple reformulation of the problem. However, as with other methods, there are some computational concerns that arise for large problem instances.

Keywords: matroids, linear matroid, set covering problem, sensor networks, degree of redundancy, cogirth, girth, NP-hard.

1 Introduction

Sensor networks play an important role in industry such as monitoring chemical plants [1]. Therefore, the ability to design a reliable sensor network is important. When designing a sensor network, the degree of redundancy of the network is a way to measure the reliability of the network. Measurements acquired from the network are *redundant* if they cannot only be obtained directly, but can also be obtained from another set of measurements. It is common practice to represent the relationship between sensor measurements y and the system states u via a linear model:

$$y = Hu + \varepsilon, \tag{1}$$

*The authors are with the Department of Computational and Applied Mathematics, Rice University, Houston, TX, 77005-1827 USA (e-mail: jda2@rice.edu; illya.hicks@rice.edu).

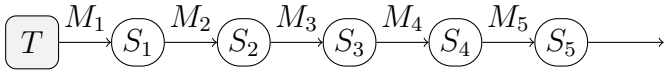

Figure 1: $S_1, ..., S_5$ are sensors in a sensor network. $M_1, ..., M_5$ represent streams of information in the network. This example is adapted from an example given by Bagajewicz and Sanchez [5].

where y is an $n \times 1$ vector, u is a $p \times 1$ vector, H is an $n \times p$ $(n > p)$ matrix with rank p, and ε is an $n \times 1$ vector representing noise accumulated by linearizing the system [2, 3]. The rank of the matrix refers to the dimension of the column space, the set of all possible linear combinations of its column vectors. In the remainder of this article including the computational results, ε is considered to be the zero vector (the system is noise free). The degree of redundancy (DoR) of the network refers to the number of sensors of the network that can fail while maintaining the ability to obtain all the measurements [4]. Therefore, a sensor network with a high DoR would be more reliable than a sensor network with a low DoR. By obtaining the DoR, it is possible to evaluate which sensor networks are more reliable than others.

Consider the example networks in Figures 1 and 2. In Figure 1, $S_1, ..., S_5$ represent sensors and $M_1, ..., M_5$ represent streams of information or measurements in the network measured by sensors to the right of them. Here, we assume that a measurement can be obtained by a sensor, or if a measurement to the right of it is known. In other words, if M_5 is known, then M_4 can be obtained. The same goes for M_1, M_2 and M_3. If all the measurements are known, then this network has a degree of redundancy of 4 since each measurement can be obtained from the measurement directly to the right of it. In Figure 2, we now have six sensors, and six streams of information. In this example, a measurement can only be obtained from other measurements if all the measurements directly to the right of it are known. For instance, both M_4 and M_5 are needed to obtain M_2 without using sensor S_2. If we assume that all the measurements are known except for M_4, then the network has a DoR of 2 since M_3 can be obtained from M_6, and M_1 can be obtained from M_2 and M_3. If all the measurements are known except for M_6, then we still have a DoR of 2 since M_2 can be obtained from M_4 and M_5 and M_1 is obtained as before. However, if M_4 and M_6 are not known, then the DoR is 1 since only M_1 can be obtained from other known measurements. Different sensor network configurations may provide different matrices in the linear system described above. That is, the matrices may or may not have structure, in particular a bordered block diagonal form (BBDF) which is discussed in section 2. However, the cases in which this structure does arise are still of great importance. In order to find the DoR of a network, we can describe a sensor failure using the matrix H.

Removing the kth row from H simulates a failure of the kth sensor in the network. Considering this, the degree of redundancy is defined as:

$$\eta = \min\{d - 1 : \exists\ H_{(-d)} \text{ s.t. } r(H_{(-d)}) < p\}, \tag{2}$$

where $H_{(-d)}$ represents a submatrix of H obtained by deleting d of the rows of H, $r(H_{(-d)})$ is the rank of the resulting matrix and p is the rank of H. It is assumed that H is a matrix with full column rank. The DoR of the network can be found by finding

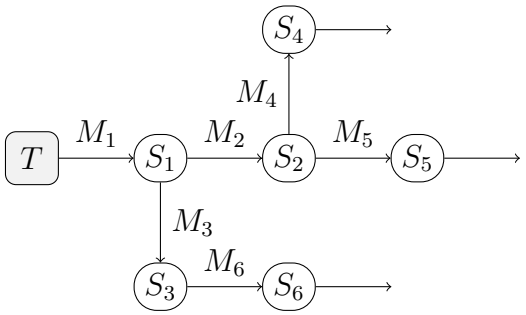

Figure 2: $S_1, ..., S_6$ are sensors in a sensor network. $M_1, ..., M_6$ represent streams of information in the network. This example is adapted from an example given by Bagajewicz and Sanchez [5].

the cogirth of the matroid obtained from the matrix H. The relationship between the DoR of a sensor network and the matroid cogirth problem will be discussed in the following subsections. In section 2, a brief overview of other existing methods that attempt to solve the cogirth problem accurately and efficiently will be given.

1.1 Matroids

The reader is referred to [7] for further discussion on terms defined in this section. A *matroid*, M, consists of an ordered pair $(S, \mathcal{I})$. S is a finite set, and $\mathcal{I}$ is a collection of subsets of S satisfying the following three properties:

(I1) $\emptyset \in \mathcal{I}$.

(I2) If $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$.

(I3) If $I_1 , I_2 \in \mathcal{I}$ and $|I_1| < |I_2|$, then $\exists\, e \in I_2 - I_1$ s.t. $I_1 \cup e \in \mathcal{I}$.

S and the elements of $\mathcal{I}$ are referred to as the *ground set* and *independent sets* of M respectively. All other subsets of S not in $\mathcal{I}$, $S - \mathcal{I}$, are *dependent sets*, and $|\cdot|$ is the cardinality of the corresponding set.

Consider the real-valued matrix given in Figure 3. In this example, the ground set, $\{1, 2, 3, 4, 5\}$, corresponds to the rows of Z, and the independent sets correspond to sets of linearly independent rows of the matrix.

$$Z = \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Figure 3: Example of a matroid $M = (S, \mathcal{I})$. The ground set S corresponds to row indices and $\mathcal{I}$ corresponds to sets of linearly independent rows.

The matroid obtained from Z is called a *linear matroid* and is denoted by $M[Z]$. Note that a matroid can also be obtained by letting the ground set correspond to

columns of a matrix and the independent sets correspond to sets of linearly independent columns. However, the matroid obtained by using the columns of the matrix is different than the matroid obtained by using the rows of the same matrix. For $M[Z]$, we have the following $S = \{1,2,3,4,5\}, \mathcal{I} = \{\emptyset, \{1\}, \{2\}, \{4\}, \{5\}, \{1,2\}, \{1,5\}, \{2,4\}, \{2,5\}, \{4,5\}\}$, and $S - \mathcal{I} = \{\{3\}, \{1,3\}, \{1,4\}, \{2,3\}, \{3,4\}, \{3,5\}\} \cup \{X \subseteq S : |X| \geq 3\}$

A *maximally independent set* of a matroid M is a set $J \in \mathcal{I}$ such that $J \cup x$ is dependent for all $x \in S - J$, and is referred to as a *basis* B of M. The collection of all bases of M is denoted as $\mathcal{B}(M)$. Consider rows 1 and 2 from Z. If you include any other row of Z with rows 1 and 2, then the set of rows becomes linearly dependent. Therefore, $\{1,2\}$ is a basis of $M[Z]$. The collection of bases of $M[Z]$ is $\{\{1,2\}, \{1,5\}, \{2,4\}, \{2,5\}, \{4,5\}\}$ and satisfy the following two properties. Observe that all the bases have the same cardinality.

(B1) $\mathcal{B}(M) \neq \emptyset$.

(B2) If B_1, $B_2 \in \mathcal{B}(M)$ and $x \in B_1 - B_2$, then $\exists\, y \in B_2 - B_1$ s.t. $(B_1 - \{x\}) \cup y \in \mathcal{B}(M)$.

A *minimally dependent set* of a matroid M is called a *circuit* C of M. A circuit C is a subset of S such that $C - x \in \mathcal{I}$ for all $x \in C$. The collection of all circuits of M is denoted as $\mathcal{C}(M)$. The collection of circuits of M satisfy the following three properties:

(C1) $\emptyset \notin \mathcal{C}(M)$.

(C2) If $C_1, C_2 \in \mathcal{C}(M)$, and $C_1 \subseteq C_2$, then $C_1 = C_2$.

(C3) If C_1, C_2 are distinct members of $\mathcal{C}(M)$ and $e \in C_1 \cap C_2$, then $\exists\, C_3 \in \mathcal{C}(M)$ s.t. $C_3 \subseteq (C_1 \cup C_2) - \{e\}$.

The cardinality of the smallest circuit is called the *girth* of the matroid. Referring back to the example, the collection of circuits of $M[Z]$ is $\{\{3\}, \{1,4\}, \{1,2,5\}, \{2,4,5\}\}$ and the girth is 1.

Given a matroid $M = (S, \mathcal{I})$, we consider another matroid whose ground set is also S. Given S, define the following collection of subsets of S, $\{S - B : B \in \mathcal{B}(M)\}$. This set, which will be denoted by $\mathcal{B}^*(M)$, is the set of bases of another matroid. This matroid denoted by M^* is called the *dual matroid* of M. Thus, $\mathcal{B}(M^*) = \mathcal{B}^*(M)$ and $(M^*)^* = M$. If $B \in \mathcal{B}^*(M)$, it is called a *cobasis* of M. Referring back to the example matrix Z, recall that $\mathcal{B}(M[Z]) = \{\{1,2\}, \{1,5\}, \{2,4\}, \{2,5\}, \{4,5\}\}$. Therefore, $\mathcal{B}^*(M[Z]) = \{\{3,4,5\}, \{2,3,4\}\{1,3,5\}, \{1,3,4\}, \{1,2,3\}\}$.

Similarly, $\mathcal{C}(M^*) = \mathcal{C}^*(M)$, and if $C \in \mathcal{C}^*(M)$, it is called a *cocircuit* of M. The cardinality of the smallest cocircuit is called the *cogirth* of the matroid. Referring back to the example, $\mathcal{C}^*(M[Z]) = \{\{1,2,4\}, \{1,2,5\}, \{2,4,5\}, \{1,4,5\}\}$ and the cogirth is 3. It should be noted that the bases and circuits of M^* are the cobases and cocircuits of M respectively and vice versa. The next few subsections describe how a solution for the set covering problem provides a solution for the degree of redundancy.

1.2 Finding the Degree of Redundancy

The degree of redundancy (DoR) of a sensor network provides a measure for the reliability of a given network. The ability to design reliable networks that are cost efficient is important. Therefore, it is important to find the DoR of a sensor network. The remainder of this subsection describes the relationship of the degree of redundancy (DoR) of a sensor network and the cogirth of a matroid.

Although the mathematical definition of the DoR of a sensor network provides a simple algorithm to find the DoR, the algorithm itself is by no means practical. Therefore, it is necessary to take a different approach towards the problem. Recall the linear system, $y = Hu + \varepsilon$, which is a standard description of the sensor network where H is an $n \times p$ matrix, $n > p$. Define $R = \{1, ..., n\}$ to be the set of row indices, SR to be the subset of rows of H removed, and d to be the cardinality of SR. The cardinality of the smallest subset, $\widehat{SR}$, of rows such that the rank of the resulting matrix is one less than the rank of H is what needs to be computed. The DoR is $\hat{d} - 1$, where $\hat{d} = |\widehat{SR}|$, the cardinality of $\widehat{SR}$. Note that one could consider H^T and remove sets of columns. Define $RowS = \{r_1, r_2, ...r_l\}$ where l is the number of distinct bases of the row space of H and r_j is a distinct basis for $j \in \{1, ..., l\}$. Therefore, $|\widehat{SR}| = min\{|J| : J \subseteq R, J \cap r_j \neq \emptyset \; \forall r_j \in RowS\}$. In other words, the rows of the resulting matrix $H_{(-d)}$ do not span the row space of H because the removed rows intersect every basis of the row space of H. Therefore, the rank of $H_{(-d)}$ must be less than the rank of H. Recall the example matrix from Figure 3. The row space of Z is $\mathbb{R}^2$. By removing rows $\{1, 2, 4\}$, which is a cocircuit of $M[Z]$, the resulting matrix does not span $\mathbb{R}^2$. Switching back to the matrix H, we can obtain a linear matroid $M[H]$ from H, where the distinct bases of $M[H]$ correspond to distinct bases of the row space of H. Based on the observation that $\{1, 2, 4\}$ is a cocircuit of $M[Z]$, we note that removing rows from H that correspond to a cocircuit will reduce the rank of the resulting matrix. The following subsection builds upon this observation, and describe why the cogirth of a linear matroid provides the DoR of the corresponding sensor network.

Recall that for a matroid $M = (S, \mathcal{I})$, the cobases are just the complement of the bases with respect to S. Since a basis, B, is maximally independent, $B \cup x$ contains a circuit for all $x \in S - B$. This might lead one to wonder if a circuit has a nonempty intersection with every basis in $\mathcal{B}(M)$. Consider the case of a matrix with a zero column, and obtain a matroid from this matrix. The zero column is a circuit of the resulting linear matroid, but does not intersect any basis. The following subsection discusses the how the linear matroid cogirth problem can be modeled using the set covering problem.

1.3 Set Covering Problem

It is not uncommon to construct linear and integer programs for combinatorial problems. This section shows how the cogirth problem can be described by an integer program. Let $N = \{1, ..., n\}$ and $K = \{1, ..., k\}$ for $n, k \in \mathbb{Z}^+$. Let $N_1, N_2, ..., N_k$ be a given collection of subsets of N. Each N_i is given a weight c_i. D which is a subset of K is called a *cover* of N if $\bigcup_{i \in D} N_i = N$. The weight of a cover D is $\sum_{i \in D} c_i$. The *set covering problem* (SCP) is min $\{c^T x \mid Ax \geq \mathbb{1}, \; x \text{ binary}\}$ where

A is an incidence matrix of $\{N_i \mid i \in K\}$. An incidence matrix is a matrix that represents a relationship between two types of objects. In this case, the incidence matrix indicates whether or not an element $j \in N$ is contained in a subset N_i. The entries of A, a_{ij}, are 1 if $j \in N_i$ and 0 otherwise for all $j \in N$. Further introduction of the SCP can be found in [8, 9].

In order to find the cogirth of $M[H]$, let $N = \{1, ..., n\}$, where n is the number of rows of H and let $K = \{1, ..., k\}$ where k is the number of distinct bases of $M[H]$. Let N_i correspond to a distinct basis of $M[H]$ with $c_i = 1$ for each $i \in K$. For the incidence matrix A, a_{ij} will be 1 if row j is a member of basis N_i and 0 otherwise. Therefore, the problem now becomes

$$\min \ \mathbb{1}^T x \text{ s.t. } Ax \geq \mathbb{1}, \ x \text{ binary} \tag{3}$$

The constraints demand that any feasible solution intersect every basis of $M[H]$, which is exactly what is required. In order to find the girth, let k be the number of distinct cobases of $M[H]$, and let N_i correspond to a distinct cobasis of $M[H]$. Therefore, a solution to the SCP provides a solution to the cogirth problem for linear matroids, which in turn provides the DoR of the corresponding sensor network. Further discussion of the SCP as well as methods devised to solve the SCP will be given in section 2.

2 Related Work

As an optimization problem, the set covering problem is NP-hard [6]. Therefore a polynomial time algorithm to find the cogirth of a general linear matroid does not likely exist. However, there have been several attempts to develop an algorithm or heuristic that solves the matroid cogirth problem accurately and/or efficiently. The first algorithm, which is fairly obvious and simple in nature, comes from the definition of the DoR. Recall that for a sensor network with model matrix H, the DoR is defined as the cardinality of the smallest set of rows whose removal from H reduces the rank of the resulting matrix $H_{(-d)}$ where d is the number of sets removed. Exhaustive rank testing is a brute force technique. As the definition of the DoR suggests, the idea is to iteratively remove sets of rows from H until the rank of $H_{(-d)}$ is one less than the rank of H. Although the algorithm is easy to understand, the number of possible combinations of sets of rows that are to be removed is n choose d where n is the number of rows of H. As d increases, so does the number of possible combinations. The singular value decomposition (SVD) is typically used to find the rank of matrices. However, since the computational time needed to compute the SVD of a matrix is dependent on the number of rows and columns of the matrix, the SVD can be computationally expensive on large submatrices. For practical system matrices, as d increases, the number of submatrices considered increases. In turn, the algorithm becomes computationally expensive as the number of submatrices increases. Therefore, it would not be advantageous nor practical to do this exhaustive rank testing as d and the size of the matrices increase.

Instead of performing an exhaustive rank test, there have been several algorithms proposed to address the cogirth problem for linear matroids. Boros et al. [10, 11]

present a circuit enumeration algorithm, using properties of circuits of matroids which can be found in [7]. Cho et al. [12] presents an algorithm that uses properties of disconnected matroids to further enhance an exhaustive rank testing procedure. It is shown that if a matrix, T, can be arranged in a Bordered Block Diagonal Form (BBDF), then the cogirth of the matroid derived from T can be found by focusing on smaller submatrices of the BBDF of T. This algorithm is dependent on the size of the border P, that is the number of rows in the border. Kianfar et al. [13]

$$\begin{pmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_n \\ P_1 & P_2 & \cdots & P_n \end{pmatrix}$$

Figure 4: Bordered block diagonal form [12].

also propose a 0-1 MIP formulation to solve the cogirth problem for general linear matroids to optimality, and make comparisons with the algorithm presented by Cho et al. [12]. Govindaraj [14] uses an approximation algorithm to address the cogirth problem. The approximation algorithm solves a sequence of problems taking the following form.

$$\min \ ||x||_1 \text{ s.t. } \Psi x = b \tag{4}$$

A problem is solved for each column of the matrix representation of the dual matroid, $M^*\left[H^T\right]$. In other words, each column is set as the right hand side b, and the remaining columns of the matrix is set as A. By doing so, an approximation for the smallest cocircuit can be found. A procedure to find the matrix representation of $M^*\left[H^T\right]$ can be found in [7]. It should be noted that (4) is a common problem in compressive sensing. Therefore, one could possibly consider other approximation algorithms in the compressive sensing literature.

As noted in 1.3, the cogirth problem for linear matroids can be solved using an SCP formulation. The SCP is a well-studied problem [6]. General discussions and methods proposed to address the SCP can be found in [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. Although these methods provide good alternatives, they cannot be used for this instance of the SCP because in order to use these methods, knowledge of all the constraints in (3) is required. This is equivalent to knowing all possible bases for the row space of H, which is generally not possible for large matrices. Although it is not possible to use these methods explicitly, it is possible to incorporate similar ideas and concepts in a branch-and-cut procedure that will be described in 3.

3 A Branch-and-Cut Procedure for the linear matroid cogirth problem

As discussed in 2, there are several exact methods and heuristics to solve (3) that cannot be used in the instance. For the SCP formulated from the cogirth problem,

every constraint corresponds to a basis of the row space of H, $Row(H)$, and each variable which has a cost of 1 corresponds to a row of H. Therefore, in order to have the entire system $Ax \geq \mathbb{1}$, all the bases of the $Row(H)$ must be known. It is important to find the exact degree of redundancy or at least provide a decent lower bound. However, since the entire set of bases is typically not known for a given matrix, the exact methods discussed above cannot be used. We need a procedure to add valid inequalities or cutting planes (as needed) for solving the SCP. Hence, this leads one to consider cutting plane methods. A discussion of cutting plane methods can be found in both [8] and [9]. In contrast, the general cutting plane methods discussed may not be viable options in this case, because the entire set of constraints (nor the separation algorithms to compute them) to change the SCP into an equivalent linear program is not known. Therefore, we consider a branch-and-cut procedure to address this instance of the set covering problem. Notice that a branch-and-cut strategy adds valid inequalities, finds lower bounds via a linear programming relaxation model, and provides feasible solutions as well.

Referring back to (3), since no bases of $Row(H)$ are initially known, there are not any constraints at the beginning of the formulation. Therefore, we must use a branch-and-cut algorithm to solve the SCP. A more general discussion of the branch-and-cut algorithm can be found in [9]. The implementation of the branch-and-cut algorithm is discussed below. The following subsections describe the intricacies of the algorithm.

Since no basis is initially known, the algorithm begins at the root node with no constraints. The constraint set, which will be referred to as Q, is empty, and the initial solution to the relaxation, the zero vector, is not a feasible solution to the SCP.

Branch-and-cut algorithm

Step 1: Find an initial feasible solution to the original problem using the greedy algorithm described in section 3.2.

Step 2: Given this feasible solution, $\bar{x}$, find an initial set of constraints to add to Q. That is, for each element $\bar{x}_i$ of $\bar{x}$, find a basis of $Row(H)$ using a greedy algorithm, and add it to Q. The weights of the greedy algorithm are described in section 3.1. Note, that because a greedy algorithm is used, a basis may appear more than once in this set of constraints. Redundant constraints are easily removed by Gurobi [27].

Step 3: Obtain the solution, x^*, to the linear relaxation of the integer program with constraint set Q.

Step 4: Using the greedy algorithm described in section 3.1 and x^* as weights, find a basis such that the inner product between the characteristic vector of the basis and x^* is less than 1, and add a corresponding constraint to Q. In practice, a tolerance of 0.1 is used to check a violation. The constraint set Q is a subset of the system $Ax \geq \mathbb{1}$. Because x^* is used as weights to find a new basis, it is guaranteed that a newly added constraint was not previously in Q. If it was, then x^* would have already satisfied the corresponding constraint. A solution that satisfies a constraint corresponding to a minimum weight basis will satisfy all other constraints. It is for this reason that the minimum weight basis is significant. It would not be advantageous to find every basis of $Row(H)$ since there may be a large number of them in general. With this in mind, it is often not the case that the

optimal solution to the SCP will be found at the root node since a relaxation of a subset of constraints for the SCP is being solved in the algorithm and this system of inequalities may not have the property of producing an integral solution with just a linear programming solver.
Step 5: Find a feasible solution as described in 3.2 to try to lower the upper bound and reduce the number of branch nodes considered. Steps 3-5 are repeated until no violated basis can be found.
Step 6: Begin branching, and repeat steps 3 and 4 in each branching subproblem. Stop when an optimal solution has been found or a specified time limit has been reached (36,000 sec for our runs).

Other cutting plane methods can also be incorporated to find other valid inequalities. The branching rule used in the branch-and-cut algorithm was to branch on most infeasible variable; i.e. the variable with a value closest to 0.5. We also considered branch-and-cut procedure that incorporates the bordered block diagonal form (BBDF) of a given matrix. Rearranging a matrix into BBDF is described by Cho in [26].

If a matrix can be arranged so that it is in BBDF, then we obtain the submatrices and run the proposed branch-and-cut procedure on each submatrix. As Cho [12] proved, given a matrix Z in BBDF, by finding the cogirth of a certain submatrix of Z, we can obtain the cogirth of Z. Therefore, we run the branch-and-cut procedure on the particular submatrices described in [12]. In order to obtain these submatrices, we follow the procedure described in [26]. We do not actually rearrange the matrix since there is no need. Instead, we keep track of each submatrix, and run the branch-and-cut procedure on each submatrix. The cogirth of the original matrix is the smallest cogirth of the submatrices considered. That is, we repeat steps 1-6 above. Computational results of the modified algorithm are presented in section 4. The rest of this section will concentrate on cutting plane generation subroutines that were incorporated in the algorithm, and a heuristic that was used to find feasible solutions to the SCP.

3.1 Generating Cutting Planes

As discussed above, constraints from $Ax \geq \mathbb{1}$ are added iteratively by finding bases of $Row(H)$. The algorithm attempts to add these constraints at the root node and at each branch node. In order to find bases, a greedy algorithm is used.
Greedy Algorithm for Constraints
Step 1: Given a solution, $x^* = (x_1, ...x_n)^T$, sort the elements.
Step 2: Using the sorted solution as weights, rows of H are added to a matrix, say H_s, as long as they increase the rank of H_s. The rows are added in order of minimum weight until the rank of H_s is equal to the rank of H.
Step 3: Given a basis, the corresponding constraint is added to Q if it is violated by the current solution x^*, the inner product between the x^* and the characteristic vector of the basis is less than 1. If no such basis is found, then x^* should not violate any basis.

A set of bases with a minimal intersection will provide a lower bound for the cogirth of the corresponding matroid. If there are a set of bases that are pairwise

disjoint, then at least one element from each basis must be chosen. Therefore, if an initial set of pairwise disjoint bases can be found, then a lower bound can be found. Along the same argument, if a set of bases where each pair of bases has the smallest intersection possible, then the set should also provide a good lower bound. An initial set of bases is found by first obtaining a feasible solution to the SCP. The method used to find a feasible solution is discussed in subsection 3.2. Once a feasible solution, $\bar{x}$, to the SCP is found, a greedy heuristic is used to find a set of bases. For each j such that $\bar{x}_j = 1$, a basis of $Row(H)$ containing row j is found. The weight of each row of H is initially set to zero. Every time a row is added to a basis, its weight is increased by one. Rows with minimum weight are added to a basis first. After this initial set of bases is obtained, bases are found based on the solution x^* to the updated linear relaxation. All the constraints obtained from bases are global cuts because they are all constraints in the system $Ax \geq \mathbb{1}$. The algorithm also attempts to add valid inequalities generated from these basis constraints.

Balas and Ng [24] consider the set covering polytope,

$$P_I(A) := conv\ \{x \in R^n \mid Ax \geq \mathbb{1}, x \text{ binary}\}. \tag{5}$$

They discuss the following class of inequalities. For a subset of rows S of the matrix A from $Ax \geq \mathbb{1}$, the inequality $\alpha^S x \geq 2$ associated with S is defined by

$$\alpha_j^S = \begin{cases} 0 & \text{if } a_{ij} = 0 \text{ for all } i \in S, \\ 2 & \text{if } a_{ij} = 1 \text{ for all } i \in S, \\ 1 & \text{otherwise.} \end{cases}$$

According to Balas and Ng [24], this class of inequalities, which will be referred to as CP and is valid for $P_I(A)$, can be generated using the following procedure D:

(i) add the inequalities $a^i x \geq 1$, $i \in S$

(ii) divide the resulting inequality by $|S| - \varepsilon, 0.5 < \varepsilon < 1$

(iii) round up all coefficients to the nearest integer

It would be advantageous to incorporate inequalities from this class. Fortunately, Balas and Ng [24] proved the following theorem. Let $\hat{x}$ be a fractional solution to the $Ax \geq \mathbb{1}, 0 \leq x \leq \mathbb{1}$, R be the row indices, N be the column indices, $I := \{j \in N | \hat{x}_j = 1\}$, and $R(I) := \{i \in R | a_{ij} = 0, \forall j \in I\}$.

Theorem 3.1 *Let $a^S x \geq 2$ be an inequality in the class CP that cuts off $\hat{x}$. Then $a_j^S = 0$ for all $j \in I$; i.e, $S \subseteq R(I)$*

The theorem states that the search for the set S of rows associated with the $\alpha^S x \geq 2$ that cuts of $\hat{x}$ can be restricted to $R(I)$. Procedure D is incorporated into the branch-and-cut algorithm at each branch node. After as many inequalities from $Ax \geq \mathbb{1}$ have been added to Q, the theorem proved by Balas and Ng [24] is used to try to find constraints using procedure D. Since only a subset of the original constraints is known, inequalities from the class CP may not always be found using this procedure. Further discussion of these cuts and their use in the branch-and-cut algorithm is discussed in section 5.

Beasley et al. [21] also discuss feasible solution exclusion constraints for the SCP. Suppose T_c is a set of column indices that correspond to the best feasible solution for

the SCP. It is assumed without loss of generality that $T_c - j$ is not a feasible solution for all $j \in T_c$. Then according to Balas et al. [21], the following two constraints can be added to the program.

$$\sum_{j \in T_c} x_j \leq |T_c| - 1 \tag{6}$$

$$\sum_{j \notin T_c} x_j \geq 1 \tag{7}$$

The idea is that if a better solution exists, it can be obtained by replacing at least one column in the current solution. These constraints are incorporated into the branch-and-cut algorithm. These constraints are only considered after branching has begun. After a feasible solution is found using the heuristic discussed below, a check is performed to see if it is better than the current best feasible solution. If so, then two feasible solution exclusion constraints are added. This procedure is done every time the best feasible solution is replaced in a branch node. In order to enhance the branch-and-cut algorithm, a method to find better feasible solutions to the SCP is also included. The next section discusses how feasible solutions are obtained in the algorithm.

3.2 Finding Feasible Solutions

Recall that a feasible solution to the SCP is actually a cocircuit of the matroid $M = (S, \mathcal{I})$ where the ground set S corresponds to the indices of the rows of H and $\mathcal{I}$ is the collection subsets corresponding to linearly independent sets of rows of H. Therefore, it would be beneficial to find feasible solutions as often as possible without enumerating all possible solutions. In order to find feasible solutions, a greedy algorithm is implemented. The algorithm is similar to that used to find bases. As before, the solution, x^*, to the linear program is used as weights. However, in this case, rows with max weight are removed from the matrix H one at a time until the rank of the resulting matrix is reduced by one. The rows removed represent a feasible solution to the SCP. The intuition behind removing rows in this manor is that rows with larger weights are more significant to the current relaxed subproblem and may be more likely to be contained in a cocircuit of $M\,[H]$. The only way to ensure that a feasible solution to the SCP found this way represents the smallest cocircuit is to enumerate all possible feasible solutions. However, it is not practical for this method to be used by itself. Instead, this method is used to find feasible solutions of smaller cardinality if possible. In doing so, better upper bounds for the branch-and-cut algorithm can be found and more branch nodes can be pruned earlier in the branch-and-cut algorithm. This method is applied after a new solution x^* to the relaxation is found.

3.3 0-1 MIP Heuristic

We also considered using the formulation presented by Kianfar et al. [13] as a heuristic. Instead of solving the 0-1 MIP as a MIP, for each $j \in \{1, ..., p\}$, set $z_j = 1$ and relax the binary constraints on q_i for all $i \in \{1, ..., n\}$; i.e., $0 \leq q_i \leq 1$ for all

$i \in \{1, ..., n\}$. Then, solve the resulting sequence of problems for each z_j. Let $\{z_{\sigma(1)}, ..., z_{\sigma(k)}\}$ be the k variables from $\{z_1, ..., z_p\}$, $k \leq p$ that provide the best k optimal values when each of the variables was set to 1. Note k is specified by the user. Since $\sum_{j=1}^{p} z_j = 1$ and z_j is binary for each $j \in \{1, ..., p\}$, only one variable is set to 1 in each instant. Given these k variables, k 0-1 MIP are solved with the original restriction that q_i is binary for all $i \in \{1, ..., n\}$ and the new restriction $z_{\sigma(j)} = 1$ for each $j \in \{1, ..., k\}$. In other words, for each $j \in \{1, ..., k\}$, solve the following problem.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} q_i \\ \text{s.t.} \quad & -q_i \leq \sum_{i=1}^{p} h_{ij} x_j \leq q_i, \quad i = 1, ..., n \\ & -1 + 2z_j \leq x_j \leq 1, \quad j = 1, ..., p \\ & \sum_{i=1}^{p} z_i = 1 \\ & z_{\sigma(j)} = 1 \end{aligned} \tag{8}$$

The results are presented in Table 2 in section 4.

4 Results

In this section, some computational results are run to validate the proposed branch-and-cut algorithm, and compare its performance to that of Cho et al. [12], and Kianfar et al. [13], two of the methods discussed in section 2. The branch-and-cut algorithm performed on the original matrix is called Algorithm 1, but performed poorly on the original matrix and was therefore left out of the results. The branch-and-cut algorithm that incorporates the BBDF is called Algorithm 2, and the 0-1 MIP approach of Kianfar et al. [13] is called Algorithm 3. The algorithms are compared using computational time.

The results are reported in Table 1. Test instance 1 is the same reported by Cho et al. [12]. Test instances 2-6 are those reported by Kianfar et al. [13]. Table 1 shows the size of the matrix ($n \times p$), the number of rows in the border,$|P|$, the DoR, η, and the computational times for the three algorithms for each test instance. For the test instances in which "$>$ 36000" is seen in the computational times, the algorithms were unable to solve the problem of interest to optimality. The algorithms were stopped manually. Algorithm 2 consists of running the branch-and-cut algorithm on submatrices of the original matrices. In order to find these submatrices, the BBDF of the matrix was computed. All three algorithms were tested on the same machine, a Dell Precision T3500 Tower Workstation with an $Intel^{®}$ $Xeon^{®}$ Dual Core W3505 2.53GHz processor. The algorithms were coded in C++, and Gurobi [27] was used to solve all the linear, and mixed integer program formulations created during the experiments.

Although not reported here, the best feasible solutions are found within seconds. For test instances 1, 2, and 5, the best feasible solution was an optimal solution. Although the best feasible solution found was not optimal for test instances 3, 4, and 6, the solutions found provided a bound for the optimal solution. Algorithm 2 performed better than Algorithm 3 in test instances 1, 2 and 4. Algorithm 3

performed better on the other three instances. In particular, Algorithm 3 appears to perform better when $\eta \leq 4$ and Algorithm 2 appears to perform better when $\eta > 4$. This may indicate that Algorithm 3 is better to use when η is small, and Algorithm 2 may be better to use when η is large and the given matrix has the suggested BBDF structure. Unfortunately, a range for η must be known beforehand in order to decide which algorithm to use. This is highly unlikely. In test instances 3-6, the algorithm took considerably longer to finish. This may be due to the number of blocks that have to be considered by the algorithm. In all, Algorithms 2 and 3 are competitive. The key components to the efficiency of Algorithm 2 appear to be the time it takes to find the border for the BBDF, the development of the constraint system, and the size and number of blocks that need to be solved.

Although the methods discussed in this article provide good upper bounds for the matroid cogirth problem, when considering a minimization problem, a lower bound on the optimal solution is more useful than an upper bound. However, a good lower bound is often harder to obtain. Similarly, when considering how many sensor failures are needed to lose the integrity of a sensor network, a lower bound is also more useful. With this in mind, we note a relationship between the spark of a matrix and the girth of a linear matroid. Donoho and Elad [28] define the spark of a matrix as the smallest number of linearly dependent columns of a matrix. Consider a matrix F and $M[F]$ with the indices of the columns as the ground set. The circuits of $M[F]$ are all the sets of minimally dependent columns. Therefore, the girth, cardinality of the smallest circuit, of $M[F]$ is equal to the spark of F. Also, the cogirth of $M[F]$ is equal to the spark of the matrix representation of $M^*[F]$, the dual matroid of $M[F]$. Donoho and Elad [28] provide a bound on the spark of the matrix. So, one could use this lower bound as a lower bound on the cogirth of a linear matroid. Since we are looking for the smallest cocircuit, finding the smallest set of pairwise disjoint bases can also be used as a lower bound. We note that the optimal value of the linear relaxation found during the branch-and-cut procedure also provides a lower bound.

The lower bound can be computed from the linear relaxation during the branch-and-cut procedure. Let Z of size $m \times n$, $m > n$, with rank r. The branch-and-cut procedure finds bases, feasible solutions and solves linear relaxations. To find a basis, we consider Z^T. Every column of Z^T might need to be considered, and the rank a submatrix of Z is computed to check if it provides a basis using the singular value decomposition (SVD). The complexity of the SVD is $O(mr^2 + r^3)$ [29] since the size of the submatrices considered is $O(r)$. Therefore, computing a basis is $O(n(mr^2 + r^3))$. A feasible solution is found in a similar manner, and thus can also be found in $O(n(mr^2 + r^3))$. The subroutines to find bases and feasible solutions are called in a while loop at each node of the branching procedure, but the number of iterations of the while loop can be limited. There are an exponential number of branching nodes, but in practice the branch-and-cut procedure does not consider all the nodes. A linear relaxation of the current SCP is also solved after a new basis is added to the model using Gurobi.

Table 2 has results for $k = 5$ and $k = 10$. In section 5, we present some concluding remarks and other observations.

Matrix H				Comp. Time	
Size ($n \times p$)	$\|P\|$	η	η^+	Alg. 2	Alg. 3
34×12	2	6	6	≈ 0	0.08
66×27	3	7	7	1	11.59
154×72	2	4	4	272	24.54
221×55	1	13	13	798	2090
318×144	2	4	4	918	54.42
1009×252	1	17	-	> 36000	> 36000

Table 1: Computational in seconds times for Algorithms 2 and 3. The algorithms reached optimality with the computed DoR,η^+ except for the instances when they did not finish.

Matrix H				Comp. Time	
Size ($n \times p$)	$\|P\|$	η	η^+	$k = 5$	$k = 10$
34×12	2	6	6	1	1
66×27	3	7	7	1	2
154×72	2	4	4	3	8
221×55	1	13	13	9	11
318×144	2	4	4	12	28
1009×252	1	17	17	298	572

Table 2: Computational times in seconds for Algorithm 3 as a Heuristic. η^+ is the optimal value computed by Algorithm 3.

5 Conclusion

In this paper, we proposed a branch-and-cut algorithm to solve an instance of the set covering problem. In turn, the solution to the set covering problem provides a solution to the linear matroid cogirth problem. Although the algorithm is designed to find a solution for the matroid cogirth problem, it can be adapted to find a solution for the linear matroid girth problem. One simply has to flip the values of the binary constraint matrix A for the set covering problem.

The results showed that Algorithm 1 did not perform very efficiently on most of the test instances. Some possible reasons are the following. First, as the size of the test matrices grew, the size of a basis for the matroid, and the number of bases grew as well. The number of significant bases required to solve the problems increased and so did the time needed to find them. Also, it was difficult to find cuts from the class of inequalities developed by Balas and Ng [24] discussed in Chapter 3.1 in both Algorithms 1 and 2. Therefore, it was harder to reduce the feasibility region using these cutting planes. However, the feasible solution exclusion constraints described by Beasley et al. [21] were useful in both Algorithms 1 and 2. Algorithm 2 performed better than Algorithm 1, but this was due to the number and size of matrices in the BBDF of the test matrices. A promising result is that both algorithms were able to find optimal solutions quickly even though they took longer to terminate. This was partially due to the number of branch nodes that needed to be pruned in both algorithms. Therefore, the algorithms could be manually terminated after

a given time, and used as heuristics. As We mentioned in section 4, the 0-1 MIP formulation presented by Kianfar et al. [13] can be used as a heuristic. In fact, the suggested heuristic appears to be accurate for most of the test instances and more efficient than just solving the 0-1 MIP formulation as it was originally presented.

The main goal of this research was to consider a well-known problem from a different perspective. The problem remains difficult to solve. For Algorithms 1 and 2, the ability to find cuts for the system of constraints faster is the biggest challenge. Even though both algorithms found optimal solutions within seconds, exploring as little branch nodes as possible in both algorithms is also important. Therefore, studying and finding good branch rules is also important. The ability of Algorithm 2 to find feasible solutions quickly points to the possibility of using them as heuristics. Using the 0-1 MIP formulation in a heuristic should also be considered. Solving the matroid cogirth problem for linear matroids using mixed integer programming techniques appears to be a promising avenue. In any case, ongoing efforts to solve this problem using any sort of MIP formulation should focus on finding strong valid inequalities for the problem.

Acknowledgment

The authors would like to thank the reviewers for their suggestions in the development of this paper.

References

[1] V. Václavek and M. Loucka, "Selection of measurements necessary to achieve multicomponent mass balances in chemical plant" *Chemical Engineering Science*, vol. 31, no. 12, pp. 1199-1205, 1976.

[2] K. Kianfar, A. Pourhabib, and Y. Ding. "An integer programming approach for analyzing the measurement redundancy in structured linear systems", *Automation Science and Engineering, IEEE Transactions on*, vol.8, no. 2, pp. 447-450, April 2011.

[3] W. J. Rugh. *Linear System Theory*. Prentice Hall Information and System Science Series. Prentice-Hall, Inc, New Jersey, second edition edition, 1996.

[4] M. Luong, D. Maquin, C. T. Huynh, and J. Ragot. "Observability, redundancy, reliability and integrated design of measurement systems", In *Proc of 2nd IFAC Symposium on Intelligent Components and Instruments for Control Applications, SICICA '94*, pp. 8-10, 1994.

[5] M. J. Bagajewicz and M. C. Sanchez. "Design and upgrade of nonredundant and redundant linear sensor networks", *AIChE journal*, vol. 45, no. 9, pp. 1927-1938, 1999.

[6] R. Karp. "Reducibility among combinatorial problems". In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pp. 85-103, Plenum Press, 1972.

[7] J. G. Oxley. *Matroid theory*. Oxford Science Publications. The Clarendon Press Oxford University Press, New York, 1992.

[8] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization*. Wiley-Interscience, New York, NY, USA, 1988.

[9] L. A. Wolsey. *Integer programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Inc., New York, 1998. A Wiley-Interscience Publication.

[10] E. Boros, K. M. Elbassioni, V. Gurvich, and L. Khachiyan. "Algorithms for enumerating circuits in matroids". In *Algorithms and Computation, 14th International Symposium, ISAAC 2003*, vol. 2906 of *Lecture Notes in Computer Science*, pp 485-494, Kyoto, Japan, December 2003.

[11] E. Boros, K. Elbassioni, V. Gurvich, L. Khachiyan, and K. Makino. "On the complexity of some enumeration problems for matroids." *SIAM J. Discrete Math.*, vol. 19, no., 4, pp.966-984 (electronic), 2005.

[12] J. J. Cho, Y. Chen, and Y. Ding. "On the (co)girth of a connected matroid." *Discrete Appl. Math.*, vol.155, no. 18, pp. 2456-2470, 2007.

[13] K. Kianfar, A. Pourhabib, and Y. Ding. "An integer programming approach for analyzing the measurement redundancy in structured linear systems". *Automation Science and Engineering, IEEE Transactions on*, vol. 8, no. 2, pp. 447-450, 2011.

[14] S. Govindaraj. "Calculation of sensor redundancy degree for linear sensor systems". Master's thesis, University of Iowa, 2010.

[15] V. Chvátal. "A Greedy Heuristic for the Set-Covering Problem". *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233-235, 1979.

[16] D. S. Johnson. "Approximation algorithms for combinatorial problems". *Journal of Computer and System Sciences*, vol. 9, no. 3, pp. 256-278, 1974.

[17] E. Marchiori and A. Steenbeek. "An iterated heuristic algorithm for the set covering problem". Kurt Mehlhorn, ed., *2nd Workshop on Algorithm Engineering WAE'98 - Proceedings*, pp. 155-166, 1998.

[18] N. Musliu. "Local search algorithm for unicost set covering problem". In Moonis Ali and Richard Dapoigny, editors, *Advances in Applied Artificial Intelligence*, vol. 4031 of *Lecture Notes in Computer Science*, pp. 302-311, 2006.

[19] J. E. Beasley and P. C. Chu. "A genetic algorithm for the set covering problem". *European Journal of Operational Research*, vol. 94 no. 2, pp. 392-404, 1996.

[20] J. E. Beasley. "An algorithm for set covering problem". *European Journal of Operational Research*, vol. 31, no. 1, pp. 85-93, 1987.

[21] J. E. Beasley and K. Jornsten. "Enhancing an algorithm for set covering problems". *European Journal of Operational Research*, vol. 58, no. 2, pp. 293-300, 1992.

[22] E. Balas. "Cutting planes from conditional bounds: A new approach to set covering". In R. W. Cottle, et al., editor, *Combinatorial Optimization*, vol. 12 of *Mathematical Programming Studies*, pp 19-36, 1980.

[23] E. Balas and A. Ho. "Set covering algorithms using cutting planes, heuristics, and subgradient optimization: A computational study". In R. W. Cottle, et al., editor, *Combinatorial Optimization*, vol. 12 of *Mathematical Programming Studies*, pp. 37-60, 1980.

[24] E. Balas and S. M. Ng. "On the set covering polytope: I. all the facets with coefficients in 0, 1, 2". *Mathematical Programming*, vol. 43, pp. 57-69, 1989.

[25] T. Grossman and A. Wool. "Computational experience with approximation algorithms for the set covering problem". *European Journal of Operational Research*, vol. 101, no.1, pp. 81-92, 1997.

[26] J. J. Cho. *On the robustness of clustered sensor networks*. PhD thesis, Texas A&M University,2007.

[27] Gurobi Optimization Inc. Gurobi Optimizer Reference Manual Version 4.0.0, 2010.

[28] D. L. Donoho and M. Elad. "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ-1 minimization". *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, no. 5, pp. 2197-2202, March 2003.

[29] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd ed.)* Johns Hopkins University Press, Baltimore, MD, USA, 1996