An Integer Programming Approach for Analyzing the Measurement Redundancy in Structured Linear Systems

Kiavash Kianfar, Arash Pourhabib, and Yu Ding, *Member, IEEE*

***Abstract*—A linear system whose model matrix is of size $n \times p$ is considered structured if some p row vectors in the model matrix are linearly dependent. Computing the degree of redundancy for structured linear systems is proven NP-hard. Previous computation strategy is divide-and-conquer, materialized in a bound-and-decompose algorithm, which, when the required conditions are satisfied, can compute the degree of redundancy on a set of much smaller submatrices instead of directly on the original model matrix. The limitation of this algorithm is that the current decomposition conditions are still restrictive and not always satisfied for many applications. We present a mixed integer programming (MIP) formulation of the redundancy degree problem and solve it using an existing MIP solver. Our numerical studies indicate that our approach outperforms the existing methods for many applications, especially when the decomposition conditions are not satisfied. The main contribution of the paper is that we tackle this challenging problem from a different angle and test a promising new approach. The resulting approach points to a path that can potentially solve the problem in its entirety.**

***Note to Practitioners*—People have long realized the importance of having sensor or measurement redundancy in a system as this redundancy safeguards the system against sensor failures or measurement anomalies, so much so that the degree of redundancy is a reflection of the system's reliability or fault-tolerance capability. Because of dependence relationship among the system's components or subsystems, computing the degree of redundancy is not a straightforward matter for practical systems which embed certain structure. There were just a few methods available for computing the degree of redundancy, and none of them can handle a wide variety of applications. Our paper presents an easy-to-use new method, which, although does not solve the computational issues entirely, does present a faster, competitive alternative for many applications wherein the existing methods were not able to calculate the degree of redundancy.**

***Index Terms*—Degree of redundancy, mixed integer programming, NP-hard, structured linear model.**

I. Introduction

THE research effort reported in this paper is concerned with an integer programming (IP) approach for evaluating the measurement redundancy level in linear systems. In the engineering literature, a linear model of the following format has been a popular choice for establishing connections between sensor measurements $\mathbf{y}$ and system states $\mathbf{x}$, through a system matrix $\mathbf{H}$

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{e} \tag{1}$$

where $\mathbf{y}$ and $\mathbf{e}$ are $n \times 1$ vectors, $\mathbf{x}$ is a $p \times 1$ vector, and $\mathbf{H}$ is an $n \times p$ matrix of rank p. The last term $\mathbf{e}$ is the residual term, including measurement noises as well as the higher order nonlinear effects neglected by the above model due to the action of linearization. This in fact is the observation equation used in a typical linear state-space model [1] and in the Kalman filter [2].

Manuscript received November 07, 2009; revised July 29, 2010; accepted October 08, 2010. Date of publication November 09, 2010; date of current version April 06, 2011. This paper was recommended for publication by Associate Editor M. Kamath and Editor Y. Narahari upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant CMMI-0727305.

The authors are with the Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX 77843-3131 USA (e-mail: kianfar@tamu.edu; arash.pourhabib@neo.tamu.edu; yuding@iemail.tamu.edu).

Digital Object Identifier 10.1109/TASE.2010.2089449

In (1), the number of measurements, n, should be greater than the number of states, p; otherwise one will run into an ill-posed system where there is no unique estimation of the states. Since there are more measurements than system states, loosely speaking, those beyond the smallest number of measurements necessary for uniquely estimating $\mathbf{x}$ are considered redundant. The degree of redundancy, d^*, was formally defined in the literature (see, for example, [3]) as follows:

$$d^* = \min\left\{d - 1 : \text{there exists } \mathbf{H}_{(-d)} \text{ s.t. } r\left(\mathbf{H}_{(-d)}\right) < p\right\} \tag{2}$$

where $\mathbf{H}_{(-d)}$ is a reduced matrix after deleting d rows from the original $\mathbf{H}$ matrix, and $r(\cdot)$ is the rank function. The interpretation of the above condition is simply that the redundancy degree of the linear system is characterized by how many sensor failures (or measurement outliers) the system can tolerate without sacrificing the identifiability of any state.

Researchers consider the measurement redundancy as an index of the system's reliability [3]–[5] and this explains the importance of quantitatively calculating it. When the system has no intrinsic structure, mathematically it means that any p row vectors of $\mathbf{H}$ are linearly independent and the degree of redundancy is $n - p$. In this case, $\mathbf{H}$ is called unstructured. However, when there exists some intrinsic structure in the system, some p row vectors of $\mathbf{H}$ are linearly independent and $\mathbf{H}$ is called structured. In this case, the degree of redundancy is smaller than $n - p$ but evaluating it quantitatively is no longer a straightforward task. Structured linear systems are pervasive in engineering applications. Some instances reported in the literature include a wireless sensor network [6], a distributed (wired) sensor system in multistation assembly [7], and sensor systems in electric power networks [8].

Due to the relevance of structured model matrices in engineering systems, it is important to devise efficient methods to quantitatively evaluate the degree of redundancy for such systems. There are not that many, but only a couple of methods available, which we will review in Section II. One difficulty in devising methods to address the above-posed technical question is that theoretically evaluating the degree of redundancy for a structured $\mathbf{H}$ is an NP-hard problem [9], which makes solving the problem very challenging, especially for large-scale systems.

The recently reported effort in [7] and [9] has made significant improvements toward devising efficient evaluation algorithms but the resulting methods therein work well under some restrictive conditions, and consequently, these methods are efficient for a subset of problems satisfying the required conditions but not so efficient otherwise. In this paper, we propose to look at the problem from a different angle. We formulate the redundancy evaluation problem as a (mixed) integer programming (IP) problem and solve the IP problem using a branch-and-cut method embedded in the commercial optimization solver CPLEX [10]. Our computational studies show merits of using the IP approach, and also point to the areas for potential improvement.

The remainder of this paper is organized as follows. Section II reviews the existing methods. Section III provides the details of our IP approach. Section IV includes the computational studies, as well as discussions on merits and limitations of the approaches (including the IP approach). Finally, we conclude our paper in Section V.

II. Related Work

By definition, calculating the degree of redundancy in a linear system reduces to the problem of finding the minimum number of row vectors in matrix $\mathbf{H}$ that if removed, rank of $\mathbf{H}$, i.e., $r(\mathbf{H})$, reduces. We introduce the following notations: for a $V \subseteq \{1, \ldots, n\}$, $\mathbf{H}_V$ denotes the matrix obtained by removing from $\mathbf{H}$ all rows $\mathbf{h}_i$, for $i \in \{1, \ldots, n\} \setminus V$. Also, define the operator I as $I(\mathbf{H}_V) = V$. The

problem of evaluating the degree of redundancy is to find the minimum integer value d^* such that there exists a $V \subseteq \{1,\ldots,n\}$ where $|V| = n - d^* - 1$ and $r(\mathbf{H}_V) < p$. The exhaustive rank testing procedure that finds d^* is to search over all row subsets of $\mathbf{H}$ as follows:

Algorithm 1. (Exhaustive Rank Testing [3])

Step 0—Set $d = 1$.
Step 1—Check all subsets $V \subseteq \{1,\ldots,n\}$ where $|V| = n - d$, to see if there exists any such that $r(\mathbf{H}_V) < p$. If yes, $d^* = d - 1$ and stop. Else set $d = d + 1$and repeat Step 1.

We note that the assumption that $r(\mathbf{H}) = p$ is what is usually the case in practice. The case of $r(\mathbf{H}) < p$ can obviously be handled by Algorithm 1 in a similar fashion.

If $\mathbf{H}$ possesses some special structure, the bound-and-decompose algorithm [7] has substantial computational benefits. By applying a transformation algorithm [7], [11], $\mathbf{H}$ becomes a bordered block diagonal form (BBDF) [7]

$$\mathbf{H} = \begin{bmatrix} \mathbf{B}_1 & & & \\ & \mathbf{B}_2 & & \\ & & \ddots & \\ & & & \mathbf{B}_r \\ \mathbf{S}_1 & \mathbf{S}_2 & \cdots & \mathbf{S}_r \end{bmatrix}$$

where the block $\mathbf{B}_t$ is an $n_t \times p_t$ matrix, and $\mathbf{S}_t$ is an $n_S \times p_t$ matrix for $t = 1,\ldots,r$. The $\mathbf{S}_t$ matrices form the border. We have $n_s + \sum_{t=1}^{r} n_t = n$ and $\sum_{t=1}^{r} p_t = p$. The unwritten elements are zero. Let U be a subset of $\{1,\ldots,r\}$, and define $\mathbf{H}[U]$ as the reduced BBDF obtained by removing from $\mathbf{H}$ the submatrices $\mathbf{B}_t$ and $\mathbf{S}_t$ for $t \in \{1,\ldots,r\} \setminus U$. Then, for any $k \in \{1,\ldots,r\}$, if $g^*(\mathbf{H}^T) \geq ((k+1)/k)n_S - 1$, we have $g^*(\mathbf{H}^T) = \min_U\{g^*(\mathbf{H}[U]^T) : U \subseteq \{1,\ldots,r\} \text{ and } |U| = k\}$, where $g^*(\mathbf{H})$ is the cogirth of the vector matroid defined over columns of the matrix $\mathbf{H}$ (for the concepts of matroid and cogirth, please refer to [9]).

Algorithm 2. (Bound-and-Decompose [7])

Step 0—Set $d = 1$.
Step 1—If $d < (r/(r-1))n_S - 1$, set $k^* = r$ go to Step 2; otherwise find k^* that gives minimum number of matrices to be rank-tested based on the decomposition property: $k^* = \arg\min_{n_S/(d-n_S+1)\leq k\leq r} [\sum_{U\subseteq\{1,\ldots,r\};|U|=k} \binom{n_S+\sum_{t\in U} n_t}{d}]$.
Step 2—Check all $\mathbf{H}[U]_V$, where $U \subseteq \{1,\ldots,r\}$, $|U| = k^*$, $V \subseteq I(\mathbf{H}[U])$ and $|V| = n - d$, to see if there is any such that $r(\mathbf{H}[U]_V) < p$: If yes, set $g^*(\mathbf{H}^T) = d$; stop. Else set $d = d + 1$ and go to Step 1. In the end, the redundancy degree $d^* = g^*(\mathbf{H}^T) - 1$.

III. 0-1 Mixed Integer Programming Approach

Efficiency of Algorithm 2 is related to the sparsity of the original system: if the system is highly sparse, meaning the subsystems are of very small sizes (i.e., small block sizes in BBDF) AND the interconnections are in small number (i.e., fewer border rows) as well, then the bound-and-decomposition can work effectively. Although sparse systems do exist, it is unfortunately not guaranteed that actual engineering systems always satisfy the required sparsity condition.

To address this issue, we propose a novel 0-1 Mixed Integer Programming (0-1 MIP) [12], [13] formulation for the redundancy degree problem and use it to find the redundancy degree of a structured linear system. Our 0-1 MIP formulation for the redundancy degree problem is based on the concept of null space (or kernel) of a matrix. The null space of $\mathbf{H}$, denoted by $null(H)$, is the set of vectors $\mathbf{x} \in \mathbb{R}^p$, where $\mathbf{Hx} = \mathbf{0}$. If $r(\mathbf{H}_V) = p$ then based on a fundamental property in linear algebra $null(\mathbf{H_V}) = \{\mathbf{0}\}$, and if $r(\mathbf{H}_V) < p$ then the null space will be of a higher dimension and contains nonzero vectors too. In fact, we have $r(\mathbf{H}_V) + \phi(\mathbf{H}_V) = p$, where $\phi(\mathbf{H}_V)$ denotes the nullity of $\mathbf{H}_V$, i.e., the dimension of its null space.

As such, the redundancy degree problem can be solved by finding the minimum number of vectors that if eliminated from $\mathbf{H}$, the remaining matrix, i.e., $\mathbf{H}_{V^*}$, has a nonzero null space (V^* is the index set of the remaining vectors). The redundancy degree would be one less than this minimum. This means that if $\mathbf{x} \in \mathbb{R}^p$ is the nonzero vector in $null(\mathbf{H}_{V^*})$, the number of $\mathbf{h}_i$'s, $i = 1,\ldots,n$ for which $\mathbf{h}_i\mathbf{x} \neq 0$ is minimized. Our 0-1 MIP formulation looks for such a vector $\mathbf{x}$. We assume that all the row vectors of $\mathbf{H}$ are scaled such that $\|\mathbf{h}_i\|_1 = \sum_{j=1}^{p} |h_{ij}| = 1$. In other words, if $\|\mathbf{h}_i\|_1 \neq 1$ then $\mathbf{h}_i \leftarrow \mathbf{h}_i/\|\mathbf{h}_i\|_1$. Then, our 0-1 MIP formulation is as follows:

$$\min \quad \sum_{i=1}^{n} q_i \tag{3}$$

s.t.

$$-q_i \leq \sum_{j=1}^{p} h_{ij}x_j \leq q_i \quad i=1,\ldots,n \tag{4}$$

$$-1+2z_j \leq x_j \leq 1 \quad j=1,\ldots,p \tag{5}$$

$$\sum_{j=1}^{p} z_j = 1 \tag{6}$$

$$x_j \in \mathbb{R},\ q_i,\ z_j \in \{0,1\} \quad i=1,\ldots,n;\ j=1,\ldots,p. \tag{7}$$

Based on constraints (4), if $\mathbf{h}_i\mathbf{x} \neq 0$ for any i, then the 0-1 variable q_i will get a value of 1 and if $\mathbf{h}_i\mathbf{x} = 0$ it will get a value of zero because the objective is to minimize the summation of all q_i's. Therefore, objective (3) is minimizing the total number of vectors out of all $\mathbf{h}_i$, $i = 1,\ldots,n$ for which $\mathbf{h}_i\mathbf{x} \neq 0$. For this reason, the optimal objective value will be $d^* + 1$. We will have $V^* = \{i : q_i = 0\}$. However, we need to make sure that the vector $\mathbf{x}$ is nonzero because with only constraint (4), $\mathbf{x} = \mathbf{0}$ will produce a trivial solution in which all q_i's are zeros while minimizing objective (3).

Constraints (5) and (6) are to force the requirement that $\mathbf{x}$ is nonzero. Based on the combination of these two constraints at least one element of $\boldsymbol{x}$, say x_{j^*}, will be equal to 1 where we will have $z_{j^*} = 1$. The other elements will be between -1 and 1. Therefore, $\mathbf{x}$ will be a nonzero vector, where $\|\mathbf{x}\|_\infty = \max_j |x_j| = 1$, which covers all possibilities in $\mathbb{R}^p$ up to a scaling factor.

In practical applications, it is usually true that $r(\mathbf{H}) = p$ and, therefore, the formulation (3)–(7) can be used to solve the problem. However, we would like to note that with a slight modification this formulation can also be used for the case where $r(\mathbf{H}) < p$. If $r(\mathbf{H}) < p$, then the nonzero vector $\mathbf{x}$ would already exist without eliminating any rows of $\mathbf{H}$. Then, the formulation (3)–(7) cannot be used directly.

On the other hand, a basis for the null space of $\mathbf{H}$ can be easily found using the usual techniques in linear algebra such as row reduction [14]. This basis will consist of $p - r(\mathbf{H})$ linearly independent vectors, say $\mathbf{h}'_1,\ldots,\mathbf{h}'_{p-r(\mathbf{H})}$. Now, we can solve the redundancy degree problem by adding the following set of constraints to formulation (3)–(7):

$$\sum_{j=1}^{p} a'_{ij}x_j = 0 \qquad i = 1,\ldots,p - r(\mathbf{H}). \tag{8}$$

TABLE I
PROPERTIES OF TEST INSTANCES

Model Matrix **H**					
No.	Size ($n \times p$)	n_s	r	$bound^{\#}$	d^*
1	26 × 12	2	4	1.67	4
2	66 × 27	3	8	2.42	7
3	154 × 72	2	2	3	4
4	221 × 55	2	11	1.2	13
5	318 × 144	8	4	9.67	4
6	1009 × 252	1	41	0.025	15
7	500 × 383	9	4	11	≤ 5

Notes: $^{\#}$: this is the bound $\boldsymbol{(r/(r-1))n_s - 1}$ used in Algorithm 2.

TABLE II
COMPUTATION TIMES OF ALGORITHMS 1, 2, AND 3 FOR TEST INSTANCES

Computation Time			
No.	Algorithm 1	Algorithm 2	Algorithm 3
1	8 sec.	0.1 sec.	0.1 sec.
2	> 120 hours	6.1 min	15 sec.
3	> 120 hours	120 min	66 sec.
4	> 120 hours	16.5 min	76 min
5	> 120 hours	> 120 hours	140 sec
6	> 120 hours	38.2 min	> 10 hours $^{\#\#}$
7	> 120 hours	> 120 hours	>10 hours $^{\#\#}$

Notes: $^{\#\#}$: out of memory.

These constraints force the vector $\mathbf{x}$ to be in the row space of $\mathbf{H}$, and so the optimal objective value to formulation (3)–(8) will give the minimum number of $\mathbf{h}_i$'s that must be removed in order to have a nonzero vector $\mathbf{x}$ orthogonal to all the remaining $\mathbf{h}_i$'s in the row space of $\mathbf{H}$.

We note that our 0-1 MIP formulation is completely general and can be applied to find the degree of redundancy of any linear system. This 0-1MIP formulation can be solved using existing MIP solution procedures [15]. Readers who may not be familiar with the MIP solution technique can consult references [12] and [13].

Several computer software packages are available for solving general 0-1 MIP problems using the techniques explained above. CPLEX is one of the best-known packages for this purpose [10], and thus is indeed what we used for solving our MIP formulation for several test instances.

IV. COMPUTATIONAL RESULTS AND DISCUSSION

In this section, we run some numerical experiments to test our proposed method, and compare its performance with the two existing methods, reviewed in Section II. Our 0-1 MIP method is called Algorithm 3. The criterion for comparison is of course the computation time.

The test results are summarized in Tables I and II. We take most of the testing instances from the literature. In fact, the first six instances are from three different papers: instances 1 and 2 are associated with multistation assembly applications, taken from [9], instances 3 and 5 are associated with the wireless sensor network applications, reported in [6], and instances 4 and 6 are reported in [7]. Instance 7 is a hypothetical instance created by us in order to illustrate some points we would like to make.

For each instance, Table I shows the size (n and p), number of border rows (n_S), number of blocks (r), the bound used in Step 1 of Algorithm 2 for decomposition, and the degree of redundancy (d^*). Table II shows the computation time using each algorithm. When ">120 hours" is listed in the table, it means that the routine is manually terminated after the reported amount of time. All the methods are tested on the same computer. Algorithms 1 and 2 were coded in C, and for Algorithm 3, we coded the formulation (3)–(7) in AMPL modeling language [16] and used CPLEX 10.0 solver to solve the generated formulation for each instance.

It is not surprising that Algorithm 1 is only appropriate to handle the small problems (instance 1). For smaller size instances 1, 2, and 3, the MIP performs significantly better than Algorithm 2 even for the cases where it is possible for Algorithm 2 to do a significant level of decomposition. In such cases, the 0-1 MIP solver is simply faster than the exhaustive searches that Algorithm 2 has to do on the submatrices.

For larger instances, we observe that only when Algorithm 2 can do significant decomposition, that is when the bound is much smaller than the degree of redundancy (d^*), it does better than 0-1 MIP (instances 4 and 6). However, when that is not the case (instance 5), MIP does significantly better. In instance 5, Algorithm 2 does no decomposition at any iteration because the bound condition is not satisfied.

Instance 5 demonstrates the superiority of 0-1 MIP over Algorithm 2. This case, which is from wireless sensor network, was originally reported in [6] and Algorithm 2 fails to find the exact degree of redundancy since the decomposition conditions are not satisfied. Structures like instance 5 are prevalent in the engineering applications and the 0-1 MIP approach not only can be used to find the exact solution but also can be incorporated into the lower bound-finding procedure in [6] to substantially speed up the lower bound-finding computation. However, note that if the problem is too large and significant decomposition in Algorithm 2 is not possible, all algorithms have difficulty in solving it. This is demonstrated by our hypothetical instance 7. As we see none of the algorithms is able to solve this instance within the given time limit.

It is important to note that even when 0-1 MIP is not able to solve the problem to optimality, it provides upper and lower bounds on the degree of redundancy, which could be useful in practice. Every feasible 0-1 MIP solution that is found in branch-and-cut gives an upper bound and the smallest LP relaxation optimal value over all active nodes gives a lower bound [12], [13]. For example, in instance 7 after about 4 hours of computation, our 0-1 MIP approach finds an upper bound of 5 and a lower bound of 1 for d^*. Unfortunately, it is not able to close this gap (find the optimal value) even in 10 hours and ultimately it runs out of memory because of the large size of the resulting branch-and-bound tree.

V. CONCLUDING REMARKS

We proposed a new method to calculate the degree of redundancy in linear systems using a 0-1 mixed integer programming formulation and its solution technique. This method significantly outperforms the existing algorithms in many cases, especially when the decomposition conditions are not satisfied. Combining the comparison results and observations made above, we would like to further articulate the following points.

(a) The main contribution of our research is to provide a new and useful MIP formulation for the redundancy degree problem, which can readily be solved by a commercial software. Furthermore, even if it is not able to solve a problem completely it provides upper and lower bounds on the degree of redundancy which may be useful in practice. Note that the proposed 0-1 MIP does not always outperform the previous bound-and-decompose method, especially for very large problems because it does not exploit explicitly and fully the structure embedded in the model matrix.

(b) The success of Algorithm 3 observed so far also indicates that a MIP formulation provides a promising solution approach. We believe that the true solution for this computational problem lies in efforts that ingeniously exploit the structure of a large-scale

system, beyond what has been done in the bound-and-decomposition algorithm. This does not appear to be an easy research problem, and is indeed our ongoing effort.

REFERENCES

[1] W. J. Rugh, *Linear System Theory*. Upper Saddle River, NJ: Prentice-Hall, 1996.

[2] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice*. Upper Saddle River, NJ: Prentice-Hall, 2001.

[3] M. Staroswiecki, G. Hoblos, and A. Aitouche, "Sensor network design for fault tolerant estimation," *Int. J. Adaptive Control and Signal Processing*, vol. 18, pp. 55–72, 2004.

[4] Y. Ali and S. Narasimhan, "Redundant sensor network design for linear processes," *AIChE Journal*, vol. 41, pp. 2237–2249, 1995.

[5] J. Levine and R. Marino, "On fault-tolerant observers," *IEEE Trans. Autom. Control*, vol. 35, no. 5, pp. 623–627, May 1990.

[6] J. J. Cho, Y. Ding, Y. Chen, and J. Tang, "Robust calibration for localization in clustered wireless sensor networks," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 81–95, Jan. 2010.

[7] J. J. Cho, Y. Chen, and Y. Ding, "Calculating the breakdown point condition of sparse linear models," *Technometrics*, vol. 51, pp. 34–46, 2009.

[8] L. Mili, M. G. Cheniae, and P. J. Rousseeuw, "Robust state estimation of electric power systems," *IEEE Trans. Circuits and Syst.*, vol. 41, pp. 349–358, 1994.

[9] J. J. Cho, Y. Chen, and Y. Ding, "On the (co)girth of connected matroids," *Discrete Appl. Math.*, vol. 155, pp. 2456–2470, 2007.

[10] IBM ILOG CPLEX Documentation. [Online]. Available: http://www-947.ibm.com/support/entry/portal/Documentation/Software/WebSphere/IBM_ILOG_CPLEX

[11] C. Aykanat, A. Pinar, and U. V. Catalyurek, "Permuting sparse rectangular matrices into block-diagonal form," *SIAM J. Scientific Comput.*, vol. 25, pp. 1860–1879, 2004.

[12] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York: Wiley, 1988.

[13] L. A. Wolsey, *Integer Programming*. New York: Wiley, 1998.

[14] K. M. Hoffman and R. Kunze, *Linear Algebra*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[15] E. L. Johnson, G. L. Nemhauser, and M. W. P. Savelsbergh, "Progress in linear programming-based algorithms for integer programming: An exposition," *INFORMS J. Computing*, vol. 12, pp. 2–23, 2000.

[16] R. Fourer, D. M. Gay, and B. W. Kernighan, AMPL: A Modeling Language for Mathematical Programming. Duxbury, CA, 2003.