The Sparse Null Space
Basis Problem

Thomas F. Coleman*
Alex Pothen**

TR 84-598
July 1984

*Department of Computer Science
Cornell University
Ithaca, New York 14853

**Center for Applied Mathematics
Cornell University
Ithaca, New York 14853

---

*This research was supported in part by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under contract DE-AC02-83-ER10369, and a Cornell fellowship**. This work was presented at the SIAM National Meeting at Denver, Colorado in June 1983, and at the SIAM Conference on numerical optimization at Boulder, Colorado in June 1984.

THE SPARSE NULL SPACE BASIS PROBLEM

Thomas F. Coleman
Alex Pothen

Computer Science Department
and
Centre for Applied Mathematics
Cornell University

July 1984

Abstract

The sparse null space basis problem is the following: A $t \times n$ matrix A $(t < n)$ is given. Find a matrix N, with the fewest nonzeros in it, whose columns span the null space of A. This problem arises in the design of practical algorithms for large-scale numerical optimization problems.

Surprisingly, this problem can be formulated as a combinatorial optimization problem under a non-degeneracy assumption on A. The theory of matchings in bipartite graphs—marriage theorems—can then be used to obtain the nonzero positions in N. Numerically stable matrix factorizations are used in the next phase to compute N.

We use conformal decompositions to characterize the columns of a sparsest null basis. Matroid theory is used to prove that a greedy algorithm constructs a sparsest null basis. We prove that finding a sparsest null basis is NP-hard by showing that associated matroidal and graph-theoretic problems are NP-complete. We propose two approximation algorithms to construct sparse null bases. Both of them make use of the Dulmage-Mendelsohn decomposition of rectangular matrices. One algorithm is a sparsity exploiting variant of the variable-reduction technique. The second is a locally greedy algorithm that constructs a null basis with an upper triangular submatrix. These results are extended to computing sparse orthogonal null bases. We discuss how this 'two-phase' approach can construct sparser null bases than a purely numerical approach; it is also potentially faster than the latter. Finally, we classify all known methods for constructing null bases, and show some unexpected equivalences between some of them.

1. Introduction and Overview

The development of practical algorithms for the Linear Equality Problem (LEP) is at the heart of numerical optimization. (LEP) can be expressed as

$$\begin{array}{ll} \underset{x \in \Re^n}{\text{minimize}} & f(x) \\ \text{subject to} & Ax = b. \end{array}$$

Here $f(x)$ is a nonlinear 'objective' function, $f : \Re^n \to \Re$, and we assume that f is twice-continuously differentiable. The matrix A has t rows and n columns, and $t < n$. Since each row corresponds to an equation, and each column to a variable, there are fewer constraining equations than there are variables. We assume also that A has full row rank, i.e., $\text{rank}(A) = t$.

One strategy for solving (LEP), *the null space method*, involves two phases: In phase 1, a 'feasible' vector y is determined that satisfies $Ay = 0$. In phase 2, y is corrected by a vector z in the null space of A that decreases the value of f; so, $Az = 0$, and $f(y+z) < f(y)$. We set $y := y + z$, and repeat phase 2 until f is small enough in value, or no further reduction in its value can be made.

A little cleverness in choosing the correction z will help the algorithm to converge speedily, i.e., at a quadratic rate, to a stationary point of f. We model f about the point y by a quadratic function, and choose $y + z$ to be the minimizer of this model function. This results in the system of equations

$$N^T H(y) N p = -N^T g(y), \tag{1}$$

which is solved for the vector p, and then the correction z is computed from the equation

$$z = Np.$$

Here N is a basis for the $(n-t)$-dimensional null space of A (a *null basis*), $g(y) \in \Re^n$ is the gradient of f at y, and $H(y) \in \Re^{n \times n}$ is the Hessian matrix of f at y. The system of equations (1) may be solved by computing a factorization of the projected Hessian $N^T H N$ when $n - t$ is small. For problems where $n - t$ is large, an iterative technique such as the conjugate gradient method may be used. Gill, Murray, and Wright (1981) contains a more detailed discussion of the (LEP).

Efficient algorithms to solve (LEP) are needed for two reasons: First, (LEP)s arise from mathematical models of several optimization problems that arise in practice. Second, (LEP)s arise as subproblems of more general optimization problems. Nonlinearly constrained optimization problems are often solved by linearizing the constraints, and solving a succession of resulting (LEP)s. Thus, the generalized gradient method, the augmented Lagrangian method, and the projected Lagrangian method to solve these problems are based on efficient algorithms to solve (LEP)s. (LEP)s also arise as subproblems when active set strategies are used to solve optimization problems with inequality constraints.

Our concern will be with the large-scale (LEP). In large-scale problems, the constraint matrix A has a large number of rows and columns. Fortunately, however, most of the matrix elements of A are zeros and do not need to be stored. This redeeming feature results from each equation being involved with only a few variables, and each variable occurring only in a small number of equations. The nonzero elements, a small fraction of the total number of matrix elements, can be stored without exceeding storage capacities of the computer. Such matrices, whose zero-nonzero structure can be used to advantage, are called sparse matrices. Coleman (1984) discusses the various issues that arise in large sparse numerical optimization.

Sparsity in A is good, but is not enough. The null space algorithm needs a representation of a null basis N of A. Such a basis, being a set of $n - t$ vectors that span the null space of A, is not unique. Hence, unless proper care is taken in the construction of N, even though A is sparse, N may be dense and beyond the storage capabilities of the computer.

With the above discussion to motivate us, we study the sparse Null Space Basis Problem:

	A $t \times n$ matrix A of rank $r \leq t$ is given.
(NSP)	Find a matrix N with the fewest nonzeros,
	whose columns span the null space of A.

Hereafter, we will abbreviate this to the Null Space Problem. Such an $n \times (n - r)$ matrix N is a basis for the null space of A. We have called N a null basis. It is also a *sparsest null basis*, since it contains the fewest nonzeros.

Surprisingly, (NSP) can be formulated as a combinatorial optimization problem under a non-degeneracy assumption on A. Consequently, concepts from both discrete and continuous mathematics are involved in our study of (NSP).

We describe our distinctive combinatorial approach to (NSP) in the second and third sections of this paper. In Section 2, we show how a linearly dependent set of columns can be recognized from a knowledge of which of the matrix elements of A are nonzeros. This dependence which we can identify is independent of the numeric values of the nonzeros of A, so no numeric computations are involved at this stage.

In Section 3, we show how to construct such a 'structurally' dependent set of columns using the theory of matchings in bipartite graphs. Under a non-degeneracy assumption on A, this dependent set is minimal—that is, all proper subsets of this set of columns are linearly independent. The algorithm we have designed finds a minimal dependent set from a maximum matching of A. The requisite matching theory is introduced and discussed in this section.

Since a linear combination of dependent columns of A yields the zero vector, the coefficients of the combination form a null vector of A. From the dependent set we construct, we can identify the nonzero components of the corresponding null vector of A. The numeric values of the nonzeros of the null vector can be computed in a second stage from a numeric factorization of A.

Solving (NSP) in two distinct stages is helpful in the preservation of sparsity in N, and in reducing the computational effort needed to construct a null basis. From the first stage, we can predict the storage needed for N, and even a sparse data structure for it since we know where its nonzeros occur. Hence the second numeric stage can use a static data structure, which is advantageous from an algorithmic perspective.

Sparsest null bases are characterized in Section 4. First, we show that any column of such a basis can arise only from a minimal dependent set of columns of A. Then we show that a greedy algorithm solves (NSP). This algorithm chooses, at a given step, a sparsest null vector linearly independent of the null vectors it has chosen in previous steps. This is quite surprising; a short-sighted local strategy seldom solves optimization problems. This result is true because underlying (NSP) is a combinatorial structure called a matroid. We use the theory of network flows and matroid theory to prove these results. Again, our discussion is self-contained, and we do not assume any previous knowledge of these subjects by the reader.

We characterize the zero-nonzero structure of sparsest null bases in Section 5. The complexity of (NSP) is investigated in the next section. We show that (NSP) is NP-hard, so it is unlikely that a polynomial time algorithm can be designed to solve it. We prove that finding sparsest fundamental null bases is also NP-hard by using the fact that an associated spanning tree problem on undirected graphs is NP-complete.

A decomposition of matrices introduced by Dulmage and Mendelsohn is the theme of Section 7. We show how a rectangular matrix can be decomposed into a block lower triangular structure by means of a maximum matching. Surprisingly, even though this decomposition is induced by a matching, it is independent of the matching. That is, two different maximum matchings lead to the same matrix decomposition, so this is a canonical decomposition. The relevance of the Dulmage-Mendelsohn decomposition to (NSP) is also discussed in this section.

For general matrices, we propose an approximation algorithm for (NSP) in Section 8. We show that it can be viewed as a sparsity exploiting variant of Wolfe's variable-reduction technique. Computational issues associated with this algorithm are discussed in Section 9. We also show that the Dulmage-Mendelsohn decomposition can be used to reduce the computational effort needed to find null bases; the complexity of the combinatorial algorithm is also reduced by the decomposition. A variant algorithm that constructs a null basis with an upper triangular submatrix is described in Section 10. In the next section we show how to find null bases when the non-degeneracy assumption we made, the weak Haar property, does not hold for a matrix. Null bases whose columns are orthogonal can be found by modifying the algorithms we have constructed. This is discussed in Section 12. We classify all known methods for the construction of null bases, show some surprising equivalences between some of these methods, and summarize our work in Section 13.

Finally, in the appendix we discuss a paradox that arises from this paper and the work of Hoffman and McCormick (1982) on the sparse range space problem.

2. A Structural Approach to (NSP)

Let A be as in Section 1. The following observation is central to our approach to the solution of (NSP).

Suppose we can find a subset C of the columns of A that have nonzeros *only* in a subset R of the rows of A. If C has more columns than R has rows, i.e., $|C| > |R|$, then the columns in C are linearly dependent. This follows from linear algebra, since

$$\operatorname{rank}(C) \leq \min \ \{|R|, \ |C|\} = |R|.$$

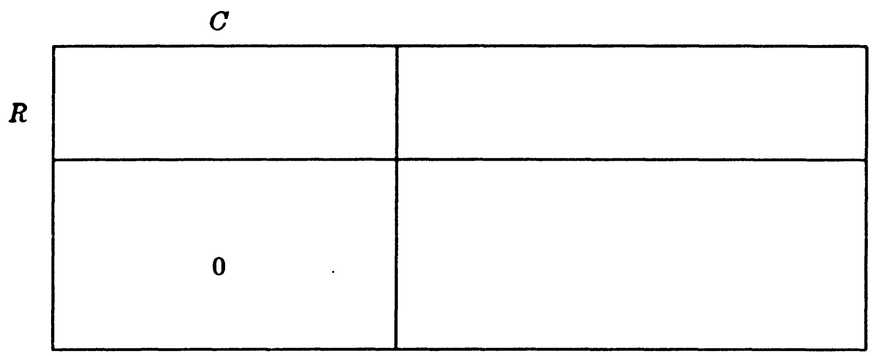

Figure 2.1.

The dependent set C.

We call such a set of columns C a *dependent set*. Figure 2.1 shows a dependent set. We can form a linear combination of the columns in C to get the zero vector. Equivalently, the coefficients of the linear combination yield a vector in the null space of A.

Suppose C has the following structure:

$$C = \begin{pmatrix} \times & 0 & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Here '$\times$'s denote strictly nonzero elements of the matrix, and '0's denote zero elements of the matrix.

No matter what the numerical values of the nonzero matrix elements are, these four columns are linearly dependent. This follows from the rank of this matrix being at most two, the number of rows in which C has nonzeros. Hence this is a structural property, and we say that the columns are ***structurally** dependent*. So we can find a vector x such that

$$C\,x = 0.$$

If the columns of A are partitioned as

$$A = (\,C \quad D\,),$$

then

$$A \begin{pmatrix} x \\ 0 \end{pmatrix} = 0,$$

and we have a vector in the null space of A.

We can be a bit more choosy. Since we want sparse null bases, we do not want C to be any bigger than necessary. Since the maximum rank of C is $|R|$, in general we need $|C| = |R| + 1$. Such a submatrix C is necessarily structurally dependent. We call a set of columns a *circuit*, if it is a minimal dependent set—that is, if C is dependent, but all proper subsets of C are independent. In the previous example, the first three columns form a circuit. Hence, we could construct a null vector of A from those three columns.

Let B denote the submatrix defined by the first three columns and the first two rows of C. The null vector associated with the circuit in our example can be computed from an LQ decomposition of B,

$$B\,Q = (\,L \quad 0\,).$$

Here Q is a 3×3 orthogonal matrix, and L is a 2×2 lower triangular matrix. Let z denote the last column of Q. Since

$$B\,z = 0,$$

the vector z is a null vector of B. As before, from z we obtain a null vector of A by appending $n - 3$ zeros to it.

We will be schizophrenic in calling the null vector associated with a minimal dependent set of columns also a circuit. Note that the null vector is numerically determined to within a scalar constant by the set of columns and the numeric values of the matrix. However, we make a distinction between a set of columns and the submatrix of A defined by the set of columns. The latter will not be called a circuit. It should be clear from the context in which the word circuit is used whether a set of columns or a null vector is meant.

What makes our structural approach attractive is the existence of a 'good' (polynomial time) algorithm to find circuits, if we are willing to make a non-degeneracy assumption about A. 'Marriage theorems', the theory of matchings in bipartite graphs, will help us to find circuits faster than numerical techniques. We avoid the 'fill' in A caused by a numerical technique, since we are dealing with only the structure of A. We can predict the number of nonzeros in the null basis N by first constructing circuits. Further, we can compute the null vectors in the basis N using a static data structure for N since we know where nonzeros occur in N. This is a great advantage since dynamic storage allocation is quite expensive in terms of time.

The non-degeneracy assumption we make won't hurt us since in the computational phase we can detect the situations when it fails. This failure arises from 'lucky' numerical dependence in the columns of A, and it will help us to get even sparser bases. This assumption is introduced and discussed in the next section.

3. A Bipartite Graph Model

Our solution to (NSP) is through the structural approach outlined in Section 2, so we now introduce a model of the problem that reflects this approach. This model will help us design a polynomial time algorithm to construct circuits—minimal dependent sets of columns. In keeping with our philosophy, we ignore the numeric values of the nonzero matrix elements of A, and distinguish only two types of matrix elements—those that are strictly nonzero, and those that are strictly zero. A model from graph theory captures this zero-nonzero structure (hereafter we will call this *structure*) of the matrix A.

The bipartite graph $G(A)$ of the matrix A has a row vertex corresponding to each row of A, and a column vertex corresponding to each column of A. An edge joins a row vertex to a column vertex if and only if the corresponding matrix element is nonzero. An example is shown in Figure 3.1.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} 1 & 3 & 0 & 5 & 0 & 0 \\ 2 & 0 & 4 & 6 & 0 & 8 \\ 0 & 0 & 0 & 0 & 7 & 9 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array}\begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{pmatrix} \times & \times & 0 & \times & 0 & 0 \\ \times & 0 & \times & \times & 0 & \times \\ 0 & 0 & 0 & 0 & \times & \times \end{pmatrix} \end{array}$$

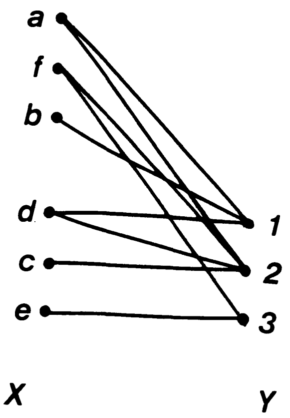

Figure 3.1.
The matrix A. the structure of A. and the bipartite graph $G(A)$.

The bipartite graph $G(A)$ is denoted by the triple $G(A) = (X, Y, E)$, where X is the set of columns, Y is the set of rows, and E is the set of edges. Clearly there are no edges between two vertices in X, or between two vertices in Y. The tuple (X, Y) is referred to as the *bipartition* of $G(A)$.

A *matching* in A is a set of nonzeros of A such that no two elements in the set are chosen from the same column or the same row. A matching of A corresponds in $G(A)$ to a set of edges (and hence a subset of E) no two of which are incident on a common vertex. A matching $\mathcal{M}$ of A (or $G(A)$) is shown in Figure 3.2. A vertex is $\mathcal{M}$*-saturated* if it is an endpoint of an edge in $\mathcal{M}$. A vertex that is not $\mathcal{M}$-saturated is $\mathcal{M}$*-unsaturated.*

$$\begin{array}{c c} & \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} \times & \times & 0 & \otimes & 0 & 0 \\ \times & 0 & \otimes & \times & 0 & \times \\ 0 & 0 & 0 & 0 & \otimes & \times \end{pmatrix} \end{array}$$

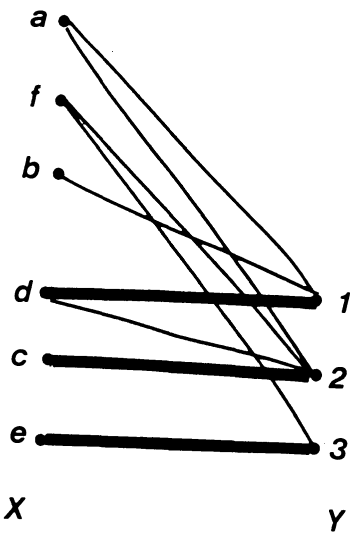

Figure 3.2.
A matching $\mathcal{M}$ in A.

The matched nonzero elements of A are circled, and the corresponding matched edges in $G(A)$ are drawn with thick lines. The matching $\mathcal{M}$ in the previous example has maximum cardinality, and hence is a *maximum matching* of A. The cardinality of a maximum matching of A is the *matching number* of A.

The theory of matchings in bipartite graphs has been called 'marriage theorems' due to the following colorful interpretation. Think of the vertices in Y as women, and the vertices in X as men, all eminently eligible to marry. If Ms. r and Mr. c are willing to marry each other, we indicate that fact by the edge (r, c) in the graph $G(A)$. If Mr. c is willing to marry Ms. r, but she is not willing, or vice versa, no edge results in the graph. Each person can express as many preferences as he/she likes, but for a marriage to be possible (for an edge to be present in the graph) such preferences have to be mutual.

We consider only heterosexual monogamous marriages. A matching corresponds to a pairing of men and women willing to marry each other. A maximum matching attempts to maximize initial nuptial happiness, by arranging as many marriages as possible. There exist several polynomial time algorithms to find maximum matchings in bipartite graphs. The fastest known has time complexity $O(|V|^{1/2}|E|)$, and is due to Hopcroft and Karp (1973). Good discussions of matching algorithms may be found in Papdimitriou and

Steiglitz (1982), and Lawler (1976). We make use of the maximum matching algorithm to construct circuits, under a non-degeneracy assumption on A.

We assume that in any maximum matching $\mathcal{M}$ of A, all the row vertices are $\mathcal{M}$-saturated. Such a matching is a *complete matching* (row-perfect matching) of A. When we consider the Dulmage-Mendelsohn decomposition in Section 7, we shall elaborate on this assumption.

A complete matching $\mathcal{M}$ of A partitions the columns of A into two sets: M, the set of $\mathcal{M}$-saturated columns, and U, the set of $\mathcal{M}$-unsaturated columns. For each column $u \in U$, we can construct a circuit of A, $n(u)$, containing u by an 'alternating path algorithm'.

A *path* in a graph is a sequence of distinct vertices $v_1, \ldots, v_k$, where (v_{i-1}, v_i) is an edge of the graph, for $1 < i \leq k$. An $\mathcal{M}$*-alternating path* is a path whose edges are alternately chosen from the set $\mathcal{M}$.

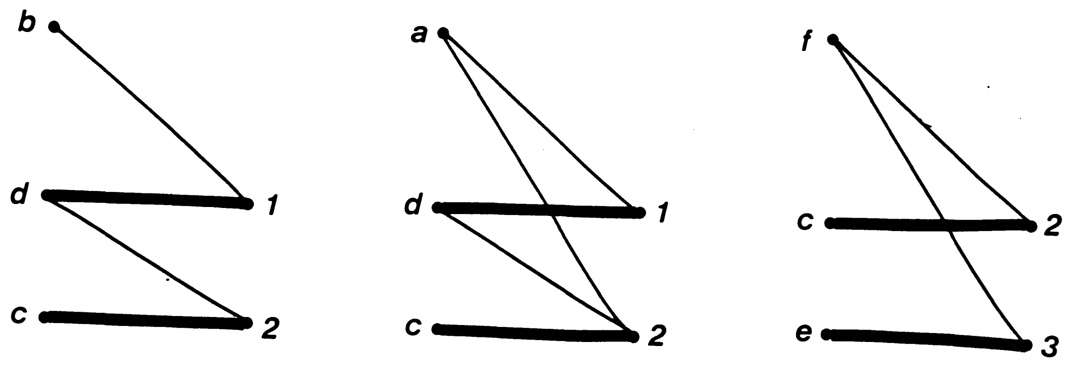

Figure 3.3.
$\mathcal{M}$-alternating paths in A.

Some of the alternating paths in a graph are shown in Figure 3.3. As before, the thick edges form the matching $\mathcal{M}$ in the graph. The sequence of edges $(b,1)$, $(1,d)$, $(d,2)$, $(2,c)$ is an $\mathcal{M}$-alternating path in A. The sequences $(a,1)$, $(1,d)$, $(d,2)$, $(2,c)$ and $(a,2)$, $(2,c)$ are alternating paths from the unmatched column a. From f there are two such paths, $(f,2)$, $(2,c)$ and $(f,3)$, $(3,e)$. For this matching, $M = \{c,d,e\}$, and $U = \{a,f,b\}$. We say that c and d are reachable from b by $\mathcal{M}$-alternating paths. We indicate this by $b \xrightarrow{M} c$ and $b \xrightarrow{M} d$. Often we will say simply that c is reachable from b. The rows 1 and 2 are also reachable from b.

For $u \in U$, the following algorithm constructs a circuit $n(u)$ containing u under a non-degeneracy assumption on A.

Algorithm 3.1.

1. Find a complete matching $\mathcal{M}$ of A; partition the columns of A as

$$A = (\, M \quad U\,).$$

2. For $u \in U$, construct the set $n(u)$ by following all $\mathcal{M}$-alternating paths from u. Thus

$$n(u) = u + \{\, v \in M : u \xrightarrow{M} v\,\}.$$

For each row vertex reachable from u by an $\mathcal{M}$-alternating path, the column m matched in $\mathcal{M}$ to r is reachable too. Hence $n(u)$ is a set of columns whose cardinality is one greater than the set of rows it is adjacent to. Under a non-degeneracy assumption on A, the set of columns $n(u)$ is indeed a circuit. From Figure 3.3, it is easy to see that

$$n(a) = \{\, a, d, c\,\}, \quad n(b) = \{\, b, d, c\,\}, \quad \text{and} \quad n(f) = \{\, f, c, e\,\}.$$

This algorithm that constructs a circuit is the major building block we use to construct sparse null bases.

Let r be the number of nonzeros in A. The algorithm of Hopcroft and Karp takes $O(\, r\,(t+n)^{1/2}\,)$ time to find a maximum matching. The set $n(u)$ can be constructed in $O(r)$ time by a depth first search. Hence the worst-case complexity of the Algorithm 3.1 is $O(r\,(t+n)^{1/2})$.

The time has come to speak of the non-degeneracy assumption we make on A. A matrix has the *weak Haar property* (wHP) if every set of columns C satisfies

$$\operatorname{rank}(C) = m(C),$$

where $m(C)$ is the matching number of C. This assumption prevents 'lucky' numerical dependences in the columns. It is always possible to assign numeric values to the nonzero elements of A such that it has wHP. For instance, if the nonzero elements are chosen to be algebraic indeterminates, then A has wHP. However, algebraic indeterminacy implies, and is therefore a stronger condition than, wHP.

The weak Haar property can be considered as a much weaker version of the *Haar property*, and hence its name. A $t \times n$ matrix with $t \le n$ has the Haar property if every subset of its columns C with $|C| \le t$ has numeric rank $|C|$. Clearly the Haar property is quite a strong condition on the columns of A. Cheney (1966) discusses the importance of the Haar property in approximation theory.

The weak Haar property is also weaker than the *matching property* (MP), an assumption under which Hoffman and McCormick (1982) solved the sparse range space problem. The matrix A has MP if for every subset of its rows R, and for every subset of its columns C, the submatrix A_{RC} has

$$\operatorname{rank}(A_{RC}) = m(A_{RC}),$$

where $m(A_{RC})$ is the matching number of the submatrix. Since we do not consider subsets of the rows of A, but deal only with subsets of columns, wHP is a weaker condition on A than MP. Clearly, MP implies the weak Haar property.

In what follows, we extensively use two properties of matrices which we introduce now. A $t \times n$ matrix A, with $t \leq n$, has the *Hall property* if every subset of k rows has nonzeros in at least k columns. This condition is necessary and sufficient for the matrix to have a complete (row-perfect) matching.

Theorem 3.1. *(Philip Hall Theorem) A has a complete matching if and only if it has the Hall property.*

Bondy and Murty (1976) present a proof. ∎

A stronger condition on A is the Strong Hall property. A $t \times n$ matrix A, with $t \leq n$, has the *Strong Hall property* if every subset of $0 < k < n$ rows has nonzeros in at least $k + 1$ columns. The terms Hall property and Strong Hall property are due to Coleman, Edenbrandt, and Gilbert (1983).

Theorem 3.2. *(Circuit Theorem) If A has wHP, then $n(u)$ is a circuit.*

Proof: Let C be the set of columns in the dependent set $n(u)$, and let C have nonzeros only in the row set R. For ease of notation, denote by B the submatrix A_{RC}.

We first show that B has the Strong Hall Property. Consider any subset S of k rows of B. S is matched in $\mathcal{M}$ to k columns, all of which are in C. If the unmatched column u has a nonzero in any of the rows in S, then S has nonzeros in at least $k + 1$ columns.

Suppose that u has no nonzero in S. Since rows in S are reachable from u by $\mathcal{M}$-alternating paths, there must exist a column, not matched to any row in S, with a nonzero in S. Again, S has nonzeros in at least $k + 1$ columns.

Let b be any column in B. Since B has the Strong Hall property, $B \setminus b$ has the Hall Property. By Theorem 3.1, the necessary and sufficient condition for a matrix to have a complete matching is that it should have the Hall property. Hence $B \setminus b$ has a complete matching of size $|R|$. Since A has wHP, the rank of $B \setminus b$ is also $|R|$, and so the columns in $B \setminus b$ are linearly independent. Since B is dependent, it follows that $n(u)$ is a circuit. ∎

Our assumption of wHP is not a restrictive one; suppose A does not have wHP, and in particular that it does not hold for $n(u)$. By construction, the matching number of $n(u)$ is $|n(u)| - 1$, and its rank is now less than this number. But $n(u)$ is a dependent set, and so it contains at least one subset that is minimal and dependent. Define the submatrix B as in the proof of Theorem 3.2. Since $n(u)$ does not have wHP, B is rank deficient. As in Section 2, the LQ decomposition of B can be used to construct a null vector, and hence a minimal dependent set of columns of B.

Theorem 3.3. *(Consolation Theorem) If $n(u)$ does not have wHP, it contains a circuit. Further, a numeric factorization of $n(u)$ can construct one such circuit.* ∎

Theorem 3.4. *Every circuit of a matrix A with wHP can be constructed by Algorithm 3.1 from some maximum matching $\mathcal{M}$ of A.*

Proof: Let C be the set of columns in a circuit, and let R and B be as in the proof of Theorem 3.2. Denote $|C|$ by c, and distinguish any one column of B as u. We claim $B \setminus u$ has the Hall property.

Suppose not. Then R has a subset of k rows adjacent to fewer than k columns, for some k. The submatrix B then looks like Figure 3.4.

(When we say B looks like a figure, we mean its columns and rows can be permuted to the structure shown in the figure.) The submatrix B then contains a dependent set of columns of size $c - k$, violating the minimality of C. So $B \setminus u$ has the Hall property.

Again by Theorem 3.1, $B \setminus u$ has a complete matching $\mathcal{M}_1$. Partition A as shown in Figure 3.5.

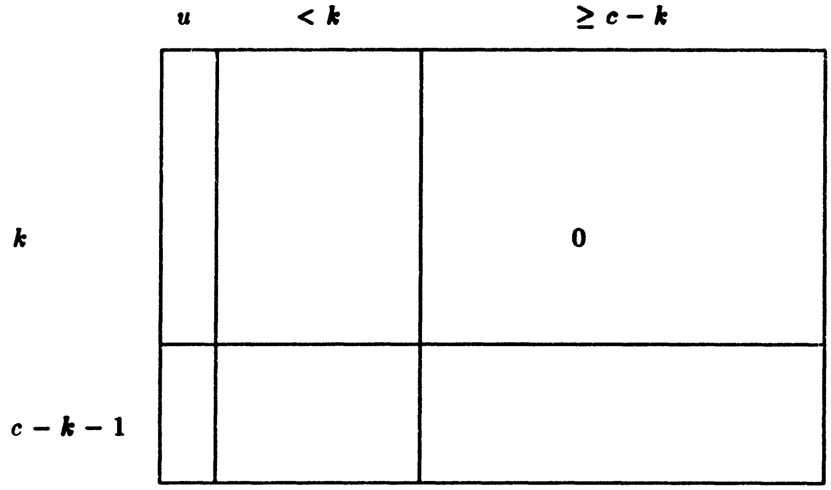

Figure 3.4.
The submatrix B.

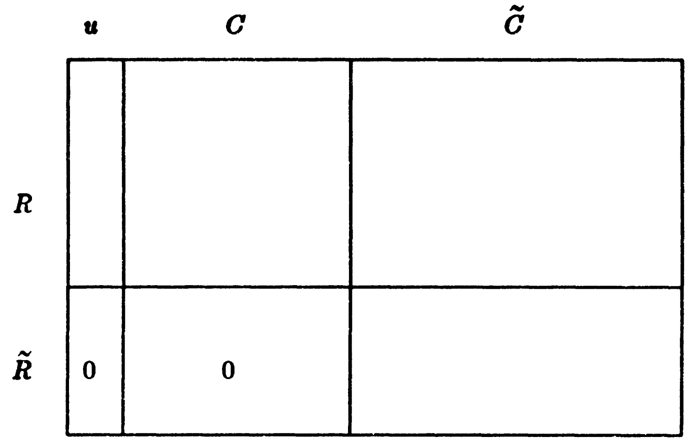

Figure 3.5.
A partition of A.

In any maximum matching of A, the row set $\tilde{R}$ can match only to the column set $\tilde{C}$. Let $\mathcal{M}_2$ be a maximum matching of the submatrix $A_{\tilde{R}\tilde{C}}$. The required maximum matching is $\mathcal{M}_1 \cup \mathcal{M}_2$. ∎

4. A Characterisation of Sparsest Null Bases

To solve (NSP), we clearly need to recognize when a null basis we have constructed is the sparsest possible basis. In this section, we characterize sparsest null bases by means of a 'greedy' algorithm, which chooses at each step a sparsest possible null vector to be in the basis.

Algorithm 4.1 (Greedy Algorithm). Given a $t \times n$ matrix A with $\text{rank}(A) = r \leq t$, find a null basis N.

> **for** $i = 1, \ldots, n-r$ **do**
> find a sparsest null vector n_i
> such that $\text{rank}((n_1, \ldots, n_i)) = i$.
> **od**
> $N = (n_1, \ldots, n_{n-r})$.

Theorem 4.1. *(Optimality Theorem) The matrix N is a sparsest null basis of A if and only if it can be constructed by the greedy algorithm.*

Algorithm 4.1 is greedy, since it augments the partial null basis at each step by a sparsest null vector linearly independent of those previously chosen. Theorem 4.1 is a surprising result; locally greedy strategies seldom lead to globally optimal solutions to optimization problems. Its proof uses network flow theory and matroid theory. No previous knowledge of these subjects is required by the reader, since our discussion is more or less self-contained.

Let the j-th component of a vector x be denoted by $(x)_j$. (This should not to be confused with the notation for a vector, say n_i.) We define the *support* of x, $S(x)$, to be

$$S(x) = \{ j : (x)_j \neq 0 \}.$$

Since circuits are minimal dependent sets of A, for a circuit c, there cannot exist a null vector x with $S(x) \subset S(c)$.

Lemma 4.2. *If c, d are circuits of A, and $S(c) = S(d)$, then c is a scalar multiple of d.*

Proof: Suppose the lemma is false. Then we can pick a scalar λ such that $(c)_j - \lambda(d)_j = 0$, for some $j \in S(c)$. But then $S(c - \lambda d) \subset S(c)$, and c is not minimal. ∎

Hence circuits of A are unique to within a multiplicative constant. We now introduce a linear algebraic concept from network flow theory, *conformal decomposition*, studied first by Camion (1968), Fulkerson (1968), and Rockefellar (1969). Lemmas 4.2 through 4.4 follow immediately from their work.

A vector x *conforms* to a vector y if

$$(x)_j \neq 0 \Rightarrow (y)_j \neq 0, \quad \text{and} \quad \text{sgn}\{(x)_j\} = \text{sgn}\{(y)_j\}.$$

Here sgn denotes the sign function. For example, let

$$\begin{aligned} \text{sgn}(x) &= (\; + \; 0 \; - \; 0 \; + \; 0 \;) \\ \text{sgn}(y) &= (\; + \; + \; - \; 0 \; + \; - \;), \end{aligned}$$

then x conforms to y, but y does not conform to x. Note that if x conforms to y, then $S(x) \subseteq S(y)$.

Lemma 4.3. *Given a null vector n, there exists a circuit c that conforms to it.*

Proof: Again, the proof is by contradiction. Choose a null vector x with the smallest $|S(x)|$ such that no circuit of A conforms to it. Let c be a circuit with $S(c) \subset S(x)$. Define the set

$$J = \{\, j : (c)_j \neq 0, \text{ and } (c)_j \text{ and } (x)_j \text{ disagree in sign} \,\}.$$

J is not the empty set, else c would conform to x. Let

$$a = \min_{j \in J} \; - \frac{(x)_j}{(c)_j}.$$

Consider the vector $z = x + ac$. By construction, z conforms to x, and $S(z) \subset S(x)$. By the selection of x there is a circuit d that conforms to z. But then d conforms to x. ∎

We can now apply Lemma 4.3 repeatedly to get

Lemma 4.4. *A null vector x can be expanded in a sum of distinct circuits*

$$x = c_1 + \cdots + c_p,$$

where each circuit c_i conforms to x. ∎

We call the above expansion the *conformal decomposition* of a null vector of A. It is not necessarily unique. A more general decomposition exists for a vector of any subspace of $\Re^n$, and is discussed by both Fulkerson and Rockefellar. We can now use Lemma 4.4 to prove that we need concern ourselves with only circuits to solve (NSP).

Theorem 4.5. *Each sparsest null vector n_i chosen by the greedy algorithm is a circuit.*

Proof: The proof is by induction on i. The result is clearly true for n_1. By the inductive hypothesis, assume that the theorem is true for all n_j, where $1 \leq j < i$.

Suppose that n_i is not a circuit. Conformally decompose n_i into a sum of circuits. At least one of the circuits in this sum, say c, must be linearly independent of $(n_1, \ldots, n_{i-1})$ since n_i is independent of them. Since n_i is not a circuit, $S(c) \subset S(n_i)$, and c is a sparser null vector than n_i which the algorithm could have chosen at this step. ∎

A similar argument can be used to prove

Theorem 4.6. *Each column of a sparsest null basis N is a circuit.* ∎

Theorem 4.6 states that the only dependent sets of interest in (NSP) are circuits. Since the greedy algorithm chooses only circuits by Theorem 4.5, the possibility now looms that the greedy algorithm could find a sparsest null basis. As Theorem 4.1 states, this suspicion is correct; a stronger result holds, namely, every sparsest null basis can be found by the greedy algorithm.

We now develop the matroid theory needed to prove the Optimality Theorem (Theorem 4.1). Let C be a finite set. Some of the subsets of C are defined to be *independent*; a subset of C that is not independent is *dependent*. Let

$$\mathcal{I} = \{ I \subseteq C : I \text{ is independent} \}.$$

We consider the situation when the independent sets satisfy the following two properties:

(M1) All subsets of an independent set are independent. (The empty set is independent by this property.)

(M2) Let I_p and I_{p+1} be independent sets with p and $p+1$ elements respectively. Then there is an element $c \in I_{p+1} \setminus I_p$ such that $I_p + c$ is independent.

Let the family of independent sets $\mathcal{I}$ satisfy (M1) and (M2). Then the structure $\mathcal{C} = (C, \mathcal{I})$ is defined to be a *matroid*.

The reader may find it convenient to think of C as the set of columns of a matrix. An independent subset of C has linearly independent columns. By linear algebra, one can establish that both (M1) and (M2) hold. Hence $\mathcal{C}$ is a matroid, and we call it the matroid generated by the columns of the matrix C.

A minimal dependent set of a matroid is called a *circuit*. We have used the word circuit to denote a minimal linearly dependent set of columns of a matrix. This usage is consistent with the definition of a circuit of a matroid. What we call a circuit of a matrix is indeed a circuit of the matroid generated by the columns of the matrix.

Hassler Whitney (1935), in founding matroid theory, observed that (M1) and (M2) were satisfied by matrices, and generalized these properties of linear independence of matrices to other combinatorial structures. Matroids that can be generated by the columns of matrices are called *matric matroids*. Columns of vertex-edge incidence matrices of graphs generate *graphic matroids*. There exist matroids that cannot be generated by any matrix, and so matroid theory is a proper generalization of linear algebra.

Consider now the following problem on a matroid $\mathcal{C}$. Each element $c \in C$ has a non-negative weight $w(c)$ given to it. The weight of an independent set is defined to be the sum of the weights of its elements. A maximal independent set is an independent set all supersets of which are dependent. We call such a set a *basis* of $\mathcal{C}$. Every basis of $\mathcal{C}$ has the same size, which is called the *rank* of $\mathcal{C}$. We are required to construct a basis of minimum weight.

Edmonds (1971) and Rado (1957) proved that the locally greedy strategy for solving the above problem constructs a basis of minimum weight. To state this result more precisely, consider the following algorithm:

Algorithm 4.2 (Matroid Greedy Algorithm). Given a matroid $\mathcal{C} = (C, \mathcal{H})$, and non-negative weights $w(c)$ for each element $c \in C$, find a basis N of minimum weight.

1. [initialize] Let N be the empty set.
2. [augment partial basis]
   Choose $c \in C \setminus N$ of minimum weight such that $N + c$ is independent.
   $N := N + c$; $C = C \setminus c$.
3. Repeat 2 until further augmentation is not possible.

Theorem 4.7. *A basis N has minimum weight if and only if it can be constructed by Algorithm 4.2.*

A proof may be found in Lawler (1976). In addition to the work of Edmonds and Rado, it draws upon a result of Gale (1968). ∎

We state one more result before proving the Optimality Theorem.

Theorem 4.8. *A matroid of rank r has at most $\binom{n}{r+1}$ circuits.*

Welsh (1976) contains a proof. ∎

Proof of Theorem 4.1:

By Theorems 4.5 and 4.6, we can restrict our attention to circuits of A. By Theorem 4.8, A has at most $\binom{n}{r+1}$ circuits. Let C be the *circuit matrix* whose columns are all the circuits of A. Thus

$$C = (c_1, \ldots, c_q).$$

Let $\mathcal{C}$ be the matroid generated by the columns of C. Recall that $S(c_j)$ is the support of the circuit c_j. To each circuit c_j, assign the positive integer weight $|S(c_j)|$.

For the circuit matroid $\mathcal{C}$, Algorithms 4.1 and 4.2 are equivalent under this choice of weights. We can now conclude from Theorem 4.7 that a null basis N is the sparsest possible if and only if it can be constructed by Algorithm 4.1. ∎

We remark that if we wanted to find a basis of $\mathcal{C}$ of maximum weight, the locally greedy strategy would work. Now, at each step, the greedy algorithm would choose an element of largest weight independent of elements previously chosen.

Unfortunately, the proof of Theorem 4.1 does not lead immediately to a polynomial time algorithm to solve (NSP). The difficulty is that the a matrix A of rank r might have $O(n^r)$ circuits.

It is difficult to determine the number of bases and circuits of a matroid. Let $b(\mathcal{C})$ denote the number of bases of a matroid $\mathcal{C}$. The following bounds are easy to show:

$$1 \leq b(\mathcal{C}) \leq \binom{n}{r}.$$

Theorem 4.8 gives an upper bound on the number of circuits of a matroid. Welsh (1976) feels that most matroids have more bases than circuits. He cites W. Quirk and P. D. Seymour, who show that graphic matroids have more bases than circuits. This result holds when the graphs underlying these matroids do not have loops. Knuth (1974) studies these numbers for matroids generated at random.

5. The Structure of Sparsest Null Bases

Null space algorithms used currently to solve the Linear Equality Problem make use of the variable-reduction technique proposed by Wolfe (1962) to construct null bases. We assume that the $t \times n$ matrix A has rank t. The matrix A is partitioned as

$$A = (M \quad U),$$

where M is a $t \times t$ nonsingular matrix. Then we construct the matrix

$$N = \begin{pmatrix} -M^{-1}U \\ I_{n-t} \end{pmatrix}.$$

Here I_{n-t} is the identity matrix of dimension $n - t$. Since $AN = 0$, the columns of N are null vectors of A. Further, each of the last $n - t$ rows of N has only one nonzero in it, and so linear combinations of the columns of N cannot produce the zero vector. Hence N is a null basis.

We call a basis with an embedded identity submatrix a *fundamental null basis*. However, sparsest null bases need not be fundamental. Choosing null bases constrained to have this structure may cost us dearly in terms of sparsity. We may be constrained to construct relatively dense fundamental bases where sparse non-fundamental null bases may exist.

But what structure should a sparsest null basis have? Theorems 5.2 and 5.3 attempt to answer that question. First, we prove Lemma 5.1 which is needed in the proof of Theorem 5.2. In what follows, by the value of a matrix we mean an assignment of nonzero numerical values to its nonzero elements.

Lemma 5.1. *Let V be a "tall and thin" matrix with n rows and m columns, where $n > m$. Distinguish some elements of V as nonzeros and the rest as zeros. If V has at least two nonzeros in each row, then there exists a vector x, and a value for V such that $Vx = 0$.*

Proof: Let row i have $|r_i| \geq 2$ nonzeros. We assign to any $|r_i| - 1$ nonzeros the value $+1$, and to the remaining element the value $1 - |r_i|$. Choose $x = (1 \ldots 1)^T$. Multiplication now shows that $Vx = 0$. ∎

Theorem 5.2. *Let the matrix V be as in Lemma 5.1. Again, distinguish some of its elements as zeros and the rest as nonzeros. V has rank m for all values if and only if it has the following structure:*

$$V = \begin{pmatrix} B \\ U_m \end{pmatrix},$$

where U_m is an $m \times m$ upper triangular matrix.

Proof: The if part is obvious. We prove the only if part. Suppose that V has rank m, but does not have the structure claimed. Permute the rows and columns of V so that it has the structure

$$V = \begin{pmatrix} B & C \\ 0 & R \end{pmatrix},$$

where R is upper triangular and maximal with respect to this property. Since R is maximal, B has at least two nonzeros in each of its rows. By Lemma 5.1, we can now find a vector x and numeric values for the nonzeros of B so that $Bx = 0$. Since

$$V \begin{pmatrix} x \\ 0 \end{pmatrix} = 0,$$

we have constructed a null vector, and V does not have rank m. This contradiction proves the theorem. ∎

We could conclude from this theorem that a sparsest null basis should have an embedded upper triangular matrix, *if we could assign any value to N*. However, we are not free to do so. We can assign any value to A; then, once the structure of N is chosen, the values of the columns of N are uniquely determined to within a multiplicative constant.

Theorem 5.3 characterizes the structure of a sparsest null basis. This result is a matroid generalization of a theorem on cycles in graphs proved by Stepanets (1964). Unfortunately, Stepanets's proof is in a Russian journal for which an English translation is unavailable, and we feel his work is not widely known. His theorem is cited in the survey paper by Turner and Kautz (1970). Our proof uses his ideas; we show they are valid in a wider context.

Let A be a $t \times n$ matrix of rank $r \leq t$. We shall denote the set of columns of the matrix A also by A. Let $n(a_j)$ denote a circuit of minimum cardinality containing the column a_j.

Theorem 5.3. *(Generalized Stepanets Theorem) Let the columns $a_1, \ldots, a_k$ be chosen such that*

$$a_1 \in A,$$
$$a_2 \in A \setminus n(a_1), \quad \ldots,$$
$$a_k \in A \setminus \bigcup_{j=1}^{k-1} n(a_j).$$

There exists a sparsest null basis N among whose columns are the circuits $n(a_1), \ldots, n(a_k)$.

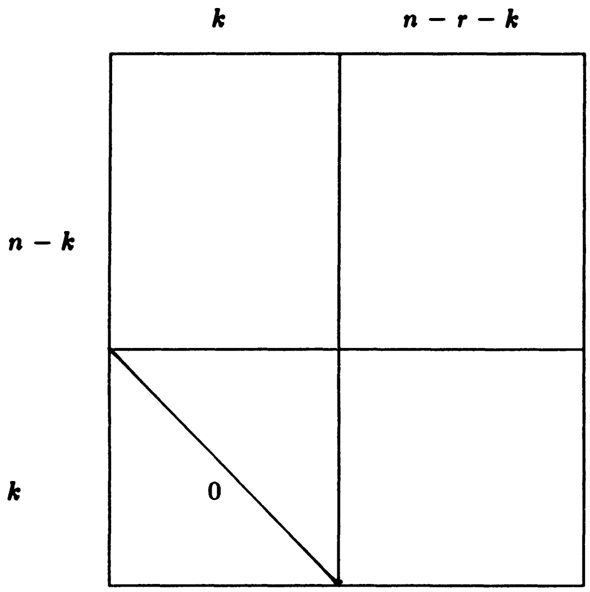

Figure 5.1.
The sparsest null basis N.

Proof: We prove the theorem by induction. Let $q = n - r$, and denote the set $(P \setminus p) \cup n$ by $P - p + n$.

Let $P = (p_1 \ldots p_q)$ be any sparsest null basis of A. Pick $n(a_1)$ to be a circuit of minimum cardinality containing the column a_1. Since P is a basis, we can expand $n(a_1)$ as

$$n(a_1) = c_1 p_1 + \cdots + c_m p_m.$$

We assume that all the coefficients c_ℓ in this expansion are nonzero. Of the circuits in this equation, there must exist at least one circuit, say p_h, which contains a_1. Consider the system

$$P_1 = P - p_h + n(a_1).$$

Clearly P_1 is a null basis. Further, since $n(a_1)$ has minimum cardinality over circuits containing a_1, P_1 is a sparsest null basis.

Recall that for $1 \le i \le k$, the i-th column a_i is chosen so that

$$a_i \in A \setminus \bigcup_{\ell=1}^{i-1} n(a_\ell).$$

For the inductive step, assume that P_{j-1} is a sparsest null basis of A, having among its columns $n(a_1), \ldots, n(a_{j-1})$, where each a_i is chosen as claimed. We choose $n(a_j)$ to be a circuit of minimum cardinality containing a_j. Expand $n(a_j)$ in the basis P_{j-1},

$$n(a_j) = c_1 p_1 + \cdots + c_m p_m,$$

where again each of the coefficients are nonzero. There is at least one circuit in this equation, say p_h, which contains a_j. The circuit p_h cannot be any one of $n(a_1), \ldots, n(a_{j-1})$ by the choice of a_j. Consider now the system

$$P_j = P_{j-1} - p_h + n(a_j).$$

As before, P_j is a sparsest null basis of A.

We take N to be P_k. This completes the proof. ∎

Since A has only n columns, there is some $k \le q$ for which choosing a column a_{k+1} is not possible, since the set differencing operation yields the empty set.

Corollary 5.4. *If $k = n - r$ in Theorem 5.3, then the system of circuits $n(a_1), \ldots, n(a_k)$ is a sparsest null basis N of A.* ∎

A similar theorem can be proven for a null basis of circuits which has a *maximum* number of nonzero elements. The proof proceeds as above by considering $n(a)$ to be a circuit of maximum cardinality containing the column a.

We remark that Theorem 5.3 does not help us to find the maximum value of k possible. If it is equal to $n - r$, then N has an upper triangular submatrix with $n - r$ columns.

6. The Complexity of (NSP) and its Variants

In Section 4, we showed that any sparsest null basis can be constructed by the greedy algorithm. So we consider the following strategy to solve (NSP): design an algorithm that finds a sparsest circuit linearly independent of the circuits chosen by the greedy algorithm in its previous steps. If we could design such an algorithm, then $n - r$ applications of it to the matrix A will solve (NSP).

Unfortunately, such a happy prospect is unlikely; we now discuss the reason why. The greedy algorithm chooses a circuit of minimum cardinality in its first step. We call such a circuit a *minimum circuit.* The cardinality of a minimum circuit is the ***girth*** of A. (Strictly, we should talk of a circuit and the girth of the matric matroid of A. However, for the sake of simplicity, we will speak of a circuit and the girth of A.)

Theorem 6.1 shows that the problem of finding a minimum circuit is a hard one; indeed, it belongs to a class of notoriously hard problems for which no good algorithms are known. This is in spite of intense effort to construct such algorithms. Unfortunately, no one has been able to prove the non-existence of a good algorithm for this class of problems, either. Theorem 6.1 states that the minimum circuit problem is NP-complete. Hence, it is as hard as any of the hard problems in the class NP. For the reader unfamiliar with this terrain, Garey and Johnson (1979) is an excellent introduction to the theory of NP-completeness.

Theorems 6.1 and 6.2 were proved also by Larry Stockmeyer; his proofs can be found in McCormick's thesis (1983). We obtained our results independently.

Theorem 6.1. *(Minimum Circuit Theorem) Given a positive integer k, it is NP-complete to find a circuit of A of cardinality k or less.*

Proof: It is easy to see that this problem is in NP. We will transform CLIQUE to the minimum circuit problem.

CLIQUE: Given a graph $G = (V, E)$ and a positive integer k, find a clique of size at least k.

Garey and Johnson (1979) prove that CLIQUE is NP-complete. Let $n = \binom{k}{2}$, and let $B(G)$ be the vertex-edge incidence matrix of G. Construct the matrix with the structure

$$A(G) = \begin{pmatrix} C_1 & C_2 \\ D & B(G) \end{pmatrix},$$

where C_1 and C_2 are fully dense matrices with $n - 1$ 'non-vertex' rows. D is a diagonal matrix with $|V|$ 'vertex' rows. We claim that $A(G)$ has girth $n + k$ if and only if G has a clique of size at least k.

For a positive integer ℓ, and any set K of ℓ vertex rows, there are at most ℓ columns of D, and $\binom{\ell}{2}$ columns of $B(G)$, adjacent to no vertex rows outside of K. Any set of columns of $A(G)$ is incident to all $n-1$ non-vertex rows. So any set of $\binom{\ell}{2}+\ell$ columns of $A(G)$ is adjacent to at least $n-1+\ell$ rows. Since the number of columns of a circuit is one greater than the number of rows it is incident on, the girth of $A(G)$ is at least $n+k$. It is now easy to see that if G has a clique of size k then the girth of $A(G)$ is $n+k$.

Conversely, if the girth is $n+k$, $A(G)$ has a set of $n+k$ columns adjacent only to $n+k-1$ rows. Since $n-1$ of the rows in this set are non-vertex, there are k vertex rows. D has at most k columns adjacent only to k vertex rows, hence $B(G)$ contributes n columns to the minimum circuit. So G has a k-clique. ∎

Theorem 6.1 leads to an easy proof that (NSP) is NP-hard. We do not claim that (NSP) is NP-complete since we do not know if (NSP) is in NP. Hence (NSP) is at least as hard, if not harder than, any of the hard problems in NP.

Theorem 6.2. *(Sparsest Null Basis Theorem) Given a positive integer k, it is NP-hard to find a null basis of A with k or fewer nonzeros.*

Proof: By Theorem 4.1, every sparsest null basis contains a minimum circuit. By Theorem 6.1, it is NP-complete to find a minimum circuit. ∎

If A is restricted to be the vertex-edge incidence matrix of a graph $G=(V,E)$, a minimum circuit can be found in $O(|V||E|)$ time by an algorithm of Itai and Rodeh (1977). In this situation, a minimum circuit corresponds to a cycle in the graph with the minimum number of edges. (This correspondence is discussed at greater length later in this section.) Recall that matroids generated by vertex-edge incidence matrices of graphs are called graphic matroids. If the matroid $\mathcal{C}=(C,\mathcal{I})$ is graphic, the set C is the set of edges of the graph G, and independent sets are subsets of edges that do not contain cycles.

Every matroid has a dual defined on the same ground set C. A basis of the dual matroid is the complement of a basis of the primal matroid. A matroid dual to a graphic matroid is cographic. Minimum circuits of cographic matroids can also be found in polynomial time.

A matrix A is totally unimodular if every subdeterminant of A is either $+1$, -1, or 0. The matroid generated by such an A is called a totally unimodular matroid. Seymour (1980) has shown that any totally unimodular matroid can be decomposed into a matroid sum of graphic matroids, cographic matroids, and copies of a special matroid on ten elements. From this result it follows that minimum circuits of totally unimodular matroids can be determined in polynomial time also.

We now proceed to show that constructing a null basis of a matrix A with the *maximum number of nonzeros* is also NP-hard when the columns in the basis are circuits of A. This problem is the Maximum Null Space Problem (MNSP). This result is obtained by first showing that an associated graph problem is NP-complete. We introduce the graph theoretical concepts necessary now. A discussion of these concepts may also be found in Christofides (1975).

Let $G = (V, E)$ be a connected graph, and let $M(G)$ be its vertex-edge incidence matrix. For ease of notation let $\nu \equiv |V|$, and $\epsilon \equiv |E|$. The matrix $M(G)$ has a row for each vertex $v \in V$, and a column for each edge $e \in E$. For an edge $e = (u, v)$, the column corresponding to e has the element 1 in the rows corresponding to vertices u and v, and zeros in all other rows.

A *cycle* in the graph G is a sequence of vertices $v_1, \ldots, v_{k-1}, v_k = v_1$, where the vertices $v_1, \ldots, v_{k-1}$ are distinct, and (v_{i-1}, v_i) is an edge in E for $i = 2, \ldots, k$. We denote a cycle and its edge incidence vector by Γ. A component γ_j of this vector is 1 if e_j is an edge of the cycle, and zero otherwise.

Since each vertex in a cycle is the endpoint of exactly two edges, we have

$$M(G)\,\Gamma^T = 0,$$

where the arithmetic is performed over the binary field. To see this, consider the matrix-vector product as a linear combination of the columns of $M(G)$. In this sum, only those edges in the cycle Γ are included. If v is an endpoint of the edge e_j, there is exactly one other edge in the cycle incident on v. So the linear combination yields a zero in the row corresponding to v. This is true for each vertex in the cycle, so Γ^T is a null vector of $M(G)$. Further, it is not possible to omit any edge of the cycle if Γ^T is to be a null vector of $M(G)$. So this vector is a circuit of the matrix.

Corresponding to a null vector of $M(G)$, there is a dependent set of columns of the matrix. Let v be the endpoint of one edge in this set of columns. There must be at least one other edge incident on vertex v, else this would not be a dependent set. Hence the degree of every such vertex is at least two. This set of edges then contains a cycle. Thus every circuit of $M(G)$ corresponds to a cycle in G.

The *cycle space* of a graph G is the null space of its vertex-edge incidence matrix $M(G)$. A *cycle basis* of a graph is a set of cycles that forms a basis for its cycle space. Let the Maximum Cycle Space Problem (MCSP) be that of determining a set of cycles which forms a basis of the cycle space of a graph, such that the total number of edges in the basis is maximum. A *maximum cycle* in a graph is a cycle with the maximum number of edges in it. A *maximum cycle basis* is a cycle basis with the maximum number of edges.

Theorem 6.3. *Given a positive integer k, it is NP-complete to find a cycle basis of a graph G with k or more edges.*

Proof: This problem is easily seen to be in NP. Consider a greedy algorithm that chooses a cycle with the largest number of edges linearly independent of cycles previously chosen. Let the edge incidence vector of each cycle of G be a column in a matrix, the cycle matrix of G. The set of columns of this matrix generates a matroid. Each cycle Γ is given a weight $K - |\Gamma|$, where K is a suitably large positive integer, and $|\Gamma|$ is the number of edges in the cycle. Algorithm 4.2 on this matroid under this weighting scheme is now equivalent to the greedy algorithm we are considering.

So a cycle basis is maximum if and only if it can be obtained by the greedy algorithm. Since the algorithm finds a maximum cycle in its first step, every maximum cycle basis contains a maximum cycle. However, it is NP-complete to find a cycle with k or more edges in a graph (Garey and Johnson (1979)). ∎

It is possible to use the above theorem to prove that (MNSP) is NP-hard. We proceed by defining a directed graph D from an undirected graph G, by arbitrarily directing the edges of the latter. The vertex-edge incidence matrix $M(D)$ of D has ν vertices and ϵ edges. Let u be the *tail*, and v the *head* of a directed edge $\{u, v\}$. The matrix $M(D)$ has in the column corresponding to this edge, the entry $+1$ in the row corresponding to v, -1 in the row corresponding to u, and zeros in all other rows.

A *cycle* in D is defined to be a cycle in G. Let $\Gamma(D)$ denote a cycle in D and also its edge incidence vector. We assign arbitrarily the clockwise orientation to all cycles in D. If e_j is an edge in $\Gamma(D)$, the component γ_j of its edge incidence vector is $+1$ if the directions of the edge and the cycle agree, and -1 if they disagree. If e_j is not an edge of the cycle $\Gamma(D)$, the component γ_j is zero.

We claim that

$$M(D)\,\Gamma(D)^T = 0,$$

where arithmetic operations are over the reals. Let v be any vertex in the cycle $\Gamma(D)$, and let edges e_1 and e_2 be incident on v. Consider the result of the linear combination of columns e_1 and e_2 in the row corresponding to v. If v is the head of both e_1 and e_2, since the edges disagree in orientation, the sum is zero. A similar result holds when v is the tail of both edges. If v is the head of one edge and the tail of the other, now the edges agree in orientation, and the sum is zero again. Since this is true of all vertices in the cycle, the claim is true.

As before, a null vector of $M(D)$ contains the edge set of a cycle in D. Hence circuits of $M(D)$ correspond to cycles of D.

Theorem 6.4. *Given a positive integer k, it is NP-hard to find null basis of A with k or more nonzeros, if each column in the basis is a circuit.*

Proof: Restrict A to vertex-edge incidence matrices of directed graphs. ∎

The Fundamental Null Space Problem

Since (NSP) is NP-hard, we cannot expect to construct sparsest null bases by a polynomial time algorithm. We now ask how hard it is to construct sparsest fundamental null bases. Recall that a fundamental null basis has the structure

$$N = \begin{pmatrix} B \\ I_{n-r} \end{pmatrix}.$$

We are lowering our sights in terms of sparsity, since sparsest bases need not be fundamental.

We formally state the Fundamental Null Space Problem:

(FNSP) Given a $t \times n$ matrix A of rank $r \leq t$, and a positive integer k, find a fundamental null basis N with k or fewer nonzeros.

Unfortunately, (FNSP) is also NP-hard. We prove this by showing that an associated graph problem is NP-complete. Hence we now introduce the additional graph theoretic concepts necessary to prove this result. More details may be found in Christofides (1975).

As before, let $G = (V, E)$ be a connected graph, and let $\nu \equiv |V|$, and $\epsilon \equiv |E|$. A *spanning tree* T of G is a connected subgraph with ν vertices and $\nu - 1$ edges. Hence $T = (V, E(T))$, where $E(T)$ denotes the edges of G in T. The corresponding *cotree* $\tilde{T}$ is the subgraph of G induced by the edges not in T. The vertex set of the cotree is the set of end points of the edges in $E(\tilde{T})$. $E(\tilde{T})$ will be the edge set of $\tilde{T}$. It is clear that $E = E(T) \cup E(\tilde{T})$. In obvious abuse of notation, we shall say an edge $e \in T$ when we mean $e \in E(T)$. Figure 6.1 shows a graph, its spanning tree, and the associated cotree.

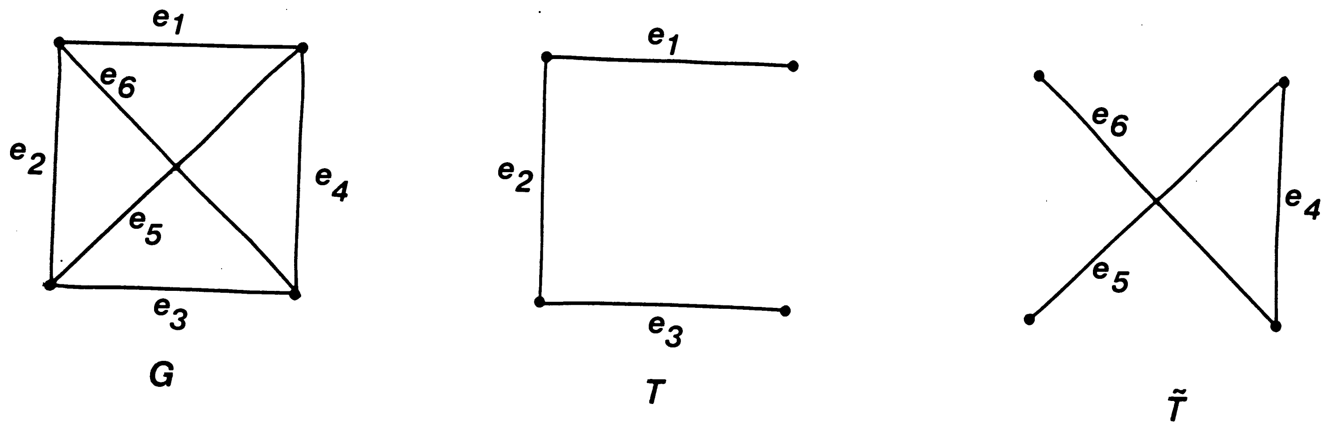

Figure 6.1.
A spanning tree and cotree of a graph.

Each edge $e \in \tilde{T}$ creates a unique cycle in the graph $T + e$, which we denote by $C(T, e)$. We call this a *fundamental cycle* created by the edge e with respect to T. Since T has $\nu - 1$ edges, $\tilde{T}$ has $\omega(G) = \epsilon - \nu + 1$ edges. Hence there are $\omega(G)$ fundamental cycles with respect to T. This number is the *cyclomatic number* of G. (This number is also referred to in the literature as the nullity or the first Betti number.)

The *fundamental cycle matrix* is a matrix $\Phi = [\phi_{ij}]$ with $\omega(G)$ rows and ϵ columns, where $\phi_{ij} = 1$ if e_j is an edge of the cycle Φ_i, and zero otherwise. If the edges of T are numbered from 1 to $\nu - 1$, and edges in $\tilde{T}$ from ν to ϵ, then Φ has the structure

$$\Phi = (\Phi_{11} \quad I),$$

where I is the identity matrix. For the graph G and spanning tree T in Figure 6.1,

$$\Phi = \begin{array}{c} \\ \Phi_1 \\ \Phi_2 \\ \Phi_3 \end{array} \begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

Let $F(T)$ be the total number of edges in the $\omega(G)$ fundamental cycles with respect to T. Hence

$$F(T) \equiv \sum_{e \in \tilde{T}} |C(T,e)|.$$

This is also the total number of nonzeros in the fundamental cycle matrix.

Let the Fundamental Cycle Space Problem be the following:

(FCSP) Given a graph H, and a positive integer B, find a spanning tree U for which $F(U) \leq B$.

Theorem 6.5. *(FCSP) is NP-complete.*

A proof of this theorem may be found in Deo, Prabhu, and Krishnamoorthy (1982). We proved this theorem independently, unaware of their work. However, our reduction is from the same problem used by these authors, the simple network design problem (Johnson, Lenstra, and Rinnooy Kan (1978)). Hence we omit the proof.

In the rest of this section, we reap the pay-offs from Theorem 6.5. First we show that (FNSP) is NP-hard.

Given a connected graph $G = (V, E)$, let D be the directed graph obtained by directing the edges of G arbitrarily. The vertex-edge incidence matrix $M(D)$ is defined as before. A *spanning tree* of D is defined to be any spanning tree of G. The *fundamental cycle matrix* of D, $\Phi(D)$ has $\omega(D) = \epsilon - \nu + 1$ rows and ϵ columns. We arbitrarily assign the clockwise orientation to all cycles in D. Let e_j be an edge of a cycle Φ_i. Then ϕ_{ij} is $+1$ if the direction of e_j agrees with the orientation of the cycle Φ_i, and -1 otherwise. If e_j is not an edge of the cycle Φ_i, then ϕ_{ij} is zero.

Each vertex in a cycle is the endpoint of exactly two edges of D. It is now easy to see that as before,

$$M(D)\, \Phi(D)^T = 0,$$

where the arithmetic is over the real field. So the matrix $\Phi(D)^T$ is in the null space of $M(D)$. Since it is fundamental, it is also a null basis of $M(D)$. Since a circuit of $M(D)$ is a cycle of D, a sparsest fundamental null basis for $M(D)$ is a sparsest fundamental cycle matrix for D.

Theorem 6.6. *(FNSP) is NP-hard.*

Proof: Restrict A to vertex-edge incidence matrices of directed graphs. ∎

$$
\begin{array}{c|c|c|}
 & n_1 & n_2 \\
\hline
t_1 & \text{body} & 0 \\
\hline
t_2 & & \text{head} \\
\hline
\end{array}
$$

Figure 7.1.

The Dulmage-Mendelsohn decomposition of a matrix with full rank.

There is a problem dual to (FNSP), the Fundamental Range Space Problem:

(FRSP) Given a $t \times n$ matrix A of rank r, and a positive integer k, find a fundamental basis $\hat{A}$ for its row space with k or fewer nonzeros.

It follows that $\hat{A}$ has the structure

$$\hat{A} = (\, I_r \quad B\,).$$

Theorem 6.7. *(FRSP) is NP-hard.*

Proof: Let

$$N = \begin{pmatrix} -B \\ I_{n-r} \end{pmatrix}$$

be a fundamental null basis of A. (We assume the columns of A are permuted to correspond to N.) Choose any r linearly independent rows of A, and call this submatrix $\tilde{A}$. Partition the columns of $\tilde{A}$ as

$$\tilde{A} = (\, \tilde{A}_1 \quad \tilde{A}_2\,)$$

to conform to the rows of N. Since $\tilde{A}$ and N are orthogonal,

$$B = \tilde{A}_1^{-1}\tilde{A}_2.$$

Let

$$\hat{A} = (\, I_r \quad B\,).$$

If $\hat{A}$ is a sparsest fundamental basis for the row space of A, then N is a sparsest fundamental null basis. Since (FNSP) is NP-hard, the theorem follows. ∎

7. The Dulmage-Mendelsohn Decomposition

Given a $t \times n$ matrix A of rank t, suppose we can partition it as shown in Figure 7.1.

Here $t_1 = n_1$, $t_2 < n_2$, $t = t_1 + t_2$, $n = n_1 + n_2$, and all elements of the upper right submatrix are zero. Then we need consider only the lower right submatrix—the head—of A to solve (NSP). This follows, since the dimension of the null space of A is $n - t$, which is equal to $n_2 - t_2$. The Dulmage-Mendelsohn decomposition gives a partition of A of this form.

If A does not have a complete matching (so A does not have rank t), the partition becomes more general, and has the structure shown in Figure 7.2. The matrix now has a tail in addition to the body and head. The head has the Strong Hall property, which will be useful in proving theorems about sparse null bases. So we develop this decomposition in this section. We state the theorems and indicate how they are applied in solving (NSP). Our proofs of these theorems use matching theory, and are new. However, they are not presented here; the proofs may be found in Pothen (1984).

Let $G = (C, R, E)$ be the bipartite graph of the matrix A. The vertex set C is the set of columns, and R is the set of rows of A. Let $\mathcal{M}$ be a maximum matching in G. In Figure 7.2, we use thick lines to represent the edges of G that are in $\mathcal{M}$. $\mathcal{M}$ induces a canonical decomposition of G which we shall call the Dulmage-Mendelsohn decomposition. The reader might find it helpful to recall the matching theory discussed in Section 2.3. By the notation $C \xrightarrow{M} r$, we mean that there is an $\mathcal{M}$-alternating path from c to r, for some $c \in C$.

We use $\mathcal{M}$ to define the following sets:

$$C_U = \{\, c \in C : c\ \mathcal{M}-\text{unsaturated}\,\} \qquad R_U = \{\, r \in R : r\ \mathcal{M}-\text{unsaturated}\,\}$$

$$R_M(C_U) = \{\, r \in R : C_U \xrightarrow{M} r\,\} \qquad C_M(R_U) = \{\, c \in C : R_U \xrightarrow{M} c\,\}$$

$$C_M(C_U) = \{\, c \in C : C_U \xrightarrow{M} c\,\} \qquad R_M(R_U) = \{\, r \in R : R_U \xrightarrow{M} r\,\}$$

$$C_{MM} = C \setminus (\, C_U \cup C_M(C_U) \cup C_M(R_U)\,)$$

$$R_{MM} = R \setminus (\, R_U \cup R_M(C_U) \cup R_M(R_U)\,).$$

In words, R_U and C_U are the sets of $\mathcal{M}$-unsaturated nodes in R and C respectively. $R_M(C_U)$ is the set of matched rows reachable by $\mathcal{M}$-alternating paths from C_U. $C_M(R_U)$ is the set of columns reachable by $\mathcal{M}$-alternating paths from R_U. $C_M(C_U)$ and $R_M(C_U)$ have similar interpretations. R_{MM} and C_{MM} are the remaining rows and columns, respectively.

For ease of notation, let $C_1 \equiv C_M(R_U)$, $C_2 \equiv C_{MM}$, $C_3 \equiv C_U \cup C_M(R_U)$. Similarly, let $R_1 \equiv R_U \cup R_M(R_U)$, $R_2 \equiv R_{MM}$, and $R_3 \equiv R_M(C_U)$.

Theorem 7.1. *(Coarse Decomposition Theorem) These sets define a structural decomposition of G shown in Figure 7.2. The sets C_i, R_i, for $i = 1, 2, 3$, are independent of the maximum matching $\mathcal{M}$.* ∎

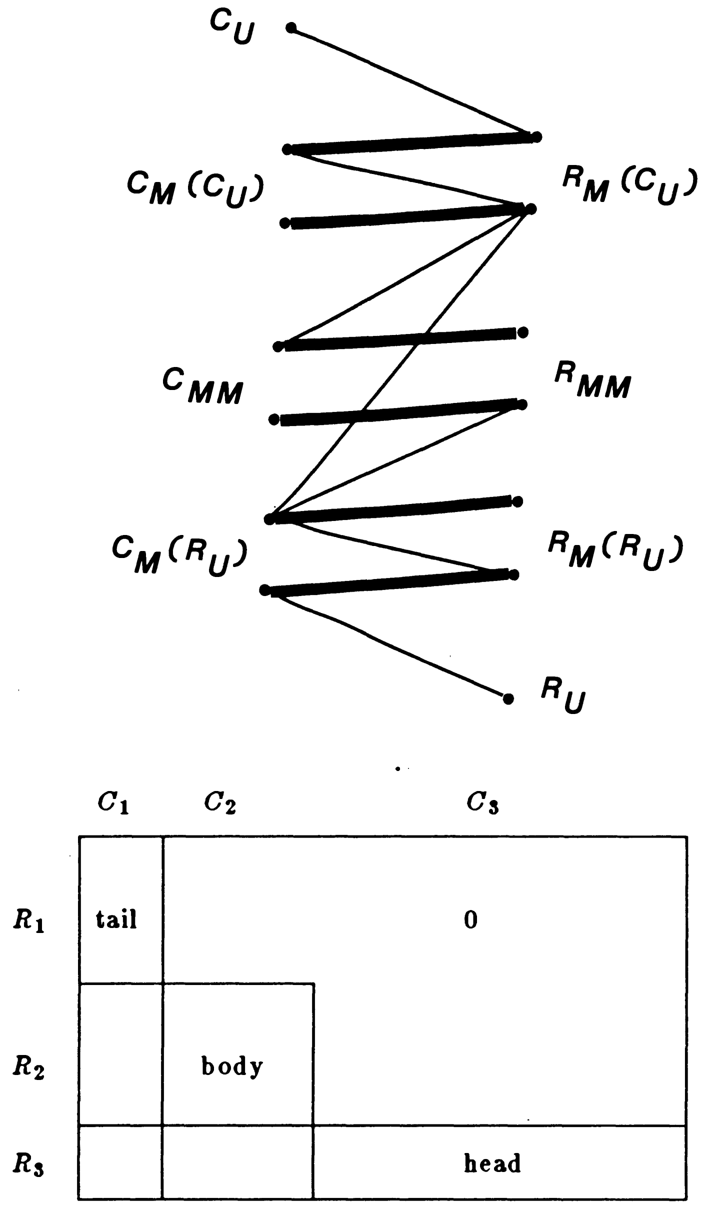

Figure 7.2.

The Dulmage-Mendelsohn decomposition.

The structural decomposition has the following features:

(i). $\text{adj}(C_U) \subseteq R_M(C_U)$.

(ii). $\text{adj}(R_U) \subseteq C_M(R_U)$.

(iii). $R_M(R_U)$ is not adjacent to $C_M(C_U)$.

(iv). $G(C_{MM}, R_{MM})$ is perfectly matched under $\mathcal{M}$.

(v). $C_M(C_U)$ is not adjacent to R_{MM}.

(vi). $R_M(R_U)$ is not adjacent to C_{MM}.

We call $A_{R_1C_1}$ the *tail*, $A_{R_2C_2}$ the *body*, and $A_{R_3C_3}$ the *head* of the matrix A. Recall that a $t \times n$ matrix has the Strong Hall property if every subset of $0 < k < n$ rows has nonzeros in at least $k + 1$ columns.

Theorem 7.2. *The head of A has the Strong Hall property. Also, the transpose of the tail has the Strong Hall property.* ∎

The head and the tail can be further decomposed into their connected components. The body has a block lower triangular decomposition, which has been extensively used to solve large systems of sparse linear equations.

Theorem 7.3. *(Fine Decomposition Theorem) A perfect matching induces the block lower triangular decomposition of the body shown in Figure 7.3. The diagonal blocks (the Dulmage-Mendelsohn blocks),* $\{A_{ii}\}$, *for* $i = 1, \ldots, k$, *are square and have the Strong Hall property. The column set* C_i *and the row set* R_i *associated with each block* A_{ii} *are independent of the matching.* ∎

We now study the implications of the coarse and fine decompositions for (NSP).

If the matrix A has a non-empty tail, it is easily seen from Figure 7.2 that A does not have full row rank. Hence if $\text{rank}(A) = t$, then A does not have a tail in its coarse decomposition. It might still have a body, and in general, has the structure shown in Figure 7.1. From the discussion immediately following that figure, only the head of A is relevant in solving (NSP).

Consider the situation when $\text{rank}(A) < t$, but A has the weak Haar property. Let the tail, head, and body be $t_1 \times n_1$, $t_2 \times n_2$, and $t_3 \times n_3$ submatrices, respectively. Here $t_1 > n_1$, $t_2 = n_2$, $t_3 < n_3$, $t = t_1 + t_2 + t_3$, and $n = n_1 + n_2 + n_3$.

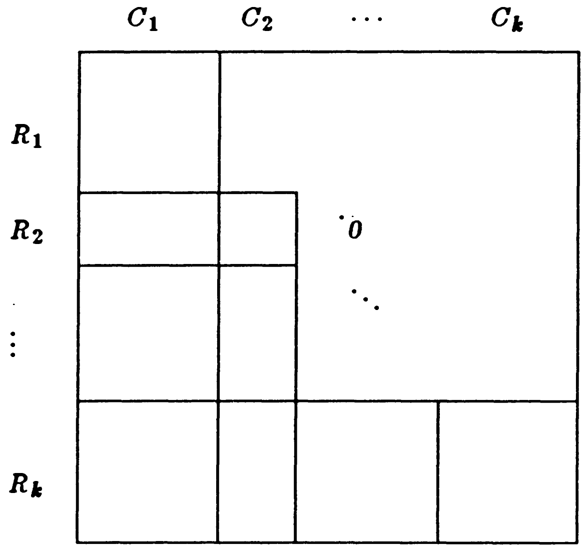

Figure 7.3.
The block lower triangular decomposition.

From Figure 7.2, the matching number of A is $m(A) = n_1 + n_2 + t_3$. Since A has the weak Haar property, $\text{rank}(A) = m(A)$. Hence the dimension of the null space of A is

$$n - m(A) = n_1 + n_2 + n_3 - (n_1 + n_2 + t_3) = n_3 - t_3.$$

Again, we need consider only the head of A to solve (NSP).

In Section 8, we consider an algorithm to construct null bases; there we show that the head does not need to be explicitly identified or isolated to make use of the above simplification. The algorithm will correctly pick out the head to construct the null basis. Thus, we save the work needed to isolate the head.

When A fails to have full row rank, and does not have the weak Haar property, the tail, body, and the head need to be identified, and the ranks of these submatrices determined.

We have already used the Strong Hall property of the head in proving Theorem 3.2. For an unmatched column u, Algorithm 3.1 constructs a dependent set $n(u)$ which has the Strong Hall property. When $n(u)$ has the weak Haar property, we proved that it is a circuit of A. In Section 10, we use the Strong Hall property of the head to design a 'myopic algorithm' to construct null bases.

The fine decomposition of the body described in Theorem 7.3 is valid for any square matrix which has a perfect matching. An important use of this decomposition is in finding LQ (or LU) factorizations of sparse matrices. It suffices to factor each of the square diagonal blocks instead of the entire matrix. As an example, consider a matrix A partitioned as

$$A = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix},$$

where A_{ii} are square for $i = 1, 2$. Also, let the factorizations

$$A_{11} Q_1 = L_{11},$$

and

$$A_{22} Q_2 = L_{22}$$

be available. Then since

$$\begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ \tilde{A}_{21} & L_{22} \end{pmatrix},$$

we have an LQ factorization of A. This results in enormous computational savings.

In Section 9, we incorporate the fine decomposition into an algorithm that constructs null bases. Instead of decomposing the body, however, we decompose a square submatrix of the head. This submatrix is constructed to have a perfect matching, and so Theorem 7.3 holds. This idea leads to savings in the work needed to compute null bases. Further, surprisingly, the complexity of the combinatorial algorithm that constructs the structure of the null basis is also reduced.

The coarse and fine decompositions were discovered by the Canadian graph theorists Dulmage, Mendelsohn, and Johnson. They proved Theorems 7.1, 7.2, and 7.3 in the series of papers, Dulmage and Mendelsohn (1958), (1959), (1963), and Johnson, Dulmage, and Mendelsohn (1962), and summarized their work in Dulmage and Mendelsohn (1967).

The fine decomposition theorem has been used to reduce the computational effort needed to solve sparse linear systems of equations. Howell (1976) contains a proof of Theorem 7.3. George and Gustavson (1980) prove Theorem 7.3 using linear algebraic techniques.

Algorithms that finely decompose a matrix by finding a maximum matching have been implemented by Gustavson (1976), and Duff and Reid (1978). A discussion of the factors involved in successful implementation of these algorithms, and an analysis of their performance may be found in Duff (1977).

The coarse decomposition is not as widely known. It is valuable in solving sparse linear least squares problems, as pointed out by Coleman, Edenbrandt, and Gilbert (1983).

The reader is warned that the terminology used here differs from those of Dulmage, Mendelsohn, and Johnson, and from those of other workers cited.

8. A Fundamental Null Basis

We now show how Algorithm 3.1, which constructs circuits, can be used repeatedly to find a null basis of A. The basis we construct is a fundamental null basis, one with an identity submatrix. Hence this is a sparse adaptation of the variable-reduction technique.

We assume that $\text{rank}(A) = t$. Any matrix with full row rank has a complete matching from our discussion in Section 7. We also assume that A has the weak Haar property. The following algorithm constructs a null basis of A.

Algorithm 8.1.

1. [initialize] Let N be the empty set.
2. [match]
   Find a complete matching $\mathcal{M}$ of A;
   partition $A = (\, M \;\; U \,)$.
3. [construct basis]
   for $u_i \in U$ **do**
      construct $n(u_i)$ by step 2 of Algorithm 3.1;
      $N := (\, N,\; n(u_i)\,)$;
   od

Theorem 8.1. *Algorithm 8.1 constructs a fundamental null basis of A.*

Proof: By Theorem 3.2, each dependent set $n(u_i)$ is a circuit. This circuit contains only one unmatched column, namely u_i; the others are matched columns that are reachable from u_i. Hence for $1 \leq i \leq n - t$, the column u_i is contained only in $n(u_i)$. Hence N has an $(n - t)$-dimensional identity submatrix. Further, N is a null basis since linear combinations of the circuits $n(u_i)$ cannot produce zeros in any of the row positions of N corresponding to the columns u_i. ∎

This algorithm constructs a null basis of the form

$$N = \begin{pmatrix} -M^{-1}U \\ I_{n-t} \end{pmatrix}.$$

Step 3 of the algorithm predicts the structure of the basis N. This information can be used to allocate storage for the nonzeros in N. Thus a static data structure for N can be used when the null vectors are being computed.

Recall that τ is the number of nonzeros in A. As discussed in Section 3, step 2 of the algorithm requires $O(\tau(n+t)^{1/2})$ time. Each circuit can be constructed in $O(\tau)$ time, so step 3 requires $O(\tau(n-t))$ time in the worst case. However, this algorithm can be modified to reduce its complexity. This modification is described in the next section.

Since A is sparse, the expected behavior of this algorithm will not be as bad as the above bounds might imply. Duff (1977) reports his numerical experience with computing the LU decompositions of several square sparse matrices of dimensions 50 and 100. In his experiments, finding a maximum matching takes about $O(\tau) + O(n)$ time. He finds that the combinatorial phase, which includes finding a maximum matching and the Dulmage-Mendelsohn blocks, typically takes about the time needed to solve a linear system of equations from the triangular factors of the matrix. Further, the time needed for either of these steps is insignificant compared to the time needed to compute the LU factors of the Dulmage-Mendelsohn blocks.

Since we consider only rows and columns that can be reached by $\mathcal{M}$-alternating paths from U, only the columns of A in the head of A are used in the construction of N. If A has a non-empty body, Algorithm 8.1 automatically excludes it since columns in the body cannot be reached from U by $\mathcal{M}$-alternating paths. Thus the head of A does not need to be identified or isolated in this situation.

To compute N, it might appear that we have to solve $n-t$ systems of equations of the form

$$C_i\, x_i = 0, \qquad \text{for} \quad i = 1, \ldots, n-t,$$

each involving a dependent set of columns of A associated with a null vector in N. Then we would have to compute LU or QR factors for each such submatrix C_i to compute the null basis. However, this turns out to be unnecessary; we need only LU or QR factors of each of the Dulmage-Mendelsohn blocks of the square submatrix M. We discuss this feature in the next section.

9. Computational Issues

In this section we discuss the issues involved in computing the fundamental null basis N constructed by Algorithm 8.1. In so doing, we will find we can modify the algorithm itself to reduce its complexity.

Let the complete matching $\mathcal{M}$ partition A into the set of matched columns M and the set of unmatched columns U. Since M is a square matrix with a perfect matching, by Theorem 7.3 we can find its Dulmage-Mendelsohn blocks $M_{11}, \ldots, M_{kk}$. Let each M_{ii} have the column set C_i and the row set R_i.

Theorem 9.1. *Let u be an unmatched column of A. If c and d are columns in C_i, and if $u \xrightarrow{M} c$, then $u \xrightarrow{M} d$.*

Proof: Since columns c and d are in C_i, by Theorem 7.3, there is an $\mathcal{M}$-alternating tour from c to d. The $\mathcal{M}$-alternating path from u to c concatenated with the tour contains an $\mathcal{M}$-alternating path from u to d. ∎

Theorem 9.2. *Let u be an unmatched column of A. Let $u \xrightarrow{M} c$ for $c \in C_i$, and for some $j > i$, let $d \in C_j$. If M_{ji} contains at least one nonzero element, then $u \xrightarrow{M} d$.*

Proof: Since M_{ji} is nonzero, there is an edge between a column $e \in C_i$ and a row $r \in R_j$. Let r be matched in $\mathcal{M}$ to column f of C_j. From Theorem 9.1, $u \xrightarrow{M} e$. This path can be continued by the edges $(e, r), (r, f)$ to $f \in C_j$. Again by Theorem 9.1, since f and d are in C_j, $u \xrightarrow{M} d$. ∎

By Theorem 9.1, if the circuit $n(u)$ contains one column from C_i, then it contains all columns of C_i. By Theorem 9.2, if the circuit $n(u)$ contains the columns in C_i, and C_i has nonzeros in a row set R_j for $j > i$, then $n(u)$ contains all the columns in C_j.

From a bipartite graph G, we define a condensed bipartite graph G^* as follows: Each of the C_i becomes a vertex, and similarly each of the R_i becomes a vertex. The other vertices are $u \in U$. The k edges (R_i, C_i) form the complete matching of G^*. There is an edge (R_i, C_j) for $j \neq i$, if for some $c \in C_j$, and $r \in R_i$, the edge (r, c) was present in G. Finally, there are edges of the form (R_i, u) if G had an edge (r, u) for some $r \in R_i$. An example is shown in Figure 9.1.

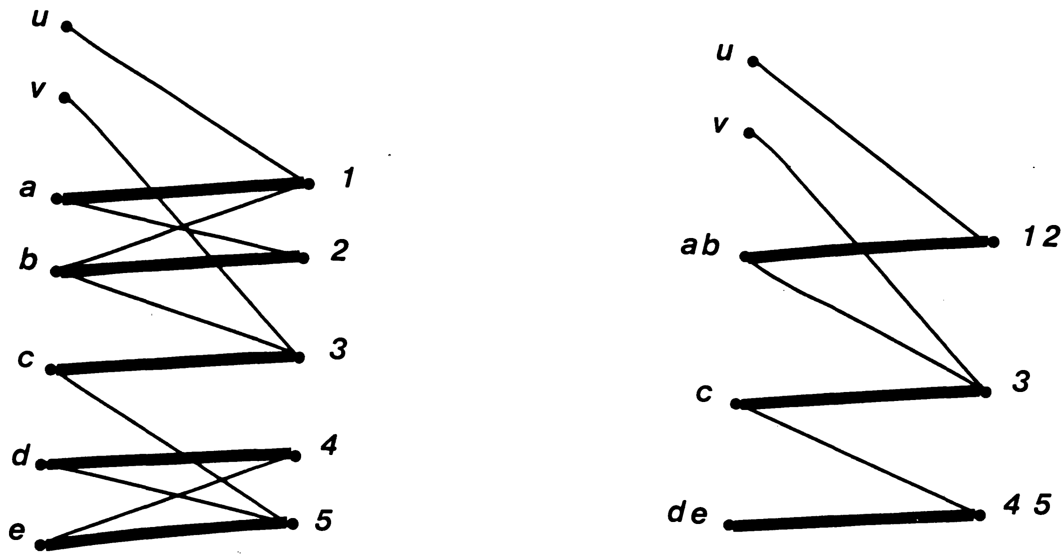

Figure 9.1.
The graph G and its condensed graph G^*.

The two theorems we have proved enable us to conclude that we can apply Algorithm 8.1 to the condensed graph G^* instead of the original graph G. This reduces the complexity of the algorithm.

We now discuss how graph condensation also helps in the optimal organization of the computation of the basis N. The QR (or LU) factors of the k Dulmage-Mendelsohn blocks of M are computed. As the orthogonal matrix Q of the block M_{ii} is computed by Givens rotations, we apply each rotation to the nonzero lower triangular blocks M_{ij}, for $j < i$, and to the columns in U; then the rotations are discarded. This saves us the storage of Q. Now the null vectors in the basis can be computed by solving triangular systems of equations.

As an illustration, consider the situation when $k = 2$, and $M_{21} \neq 0$. Partition the unmatched column u to conform with the row partition of M. Denote by x the null vector associated with the circuit $n(u)$. We need to solve the system

$$C x = 0,$$

where C denotes the columns of A in $n(u)$. Now

$$\begin{pmatrix} Q_{11} & 0 \\ 0 & Q_{22} \end{pmatrix} \begin{pmatrix} M_{11} & 0 & u_1 \\ M_{21} & M_{22} & u_2 \end{pmatrix} = \begin{pmatrix} R_{11} & 0 & \tilde{u}_1 \\ \widetilde{M}_{21} & R_{22} & \tilde{u}_2 \end{pmatrix}.$$

We can set the last component of x to 1, and partition the rest of x as x_1 and x_2 to conform to the row partition of M. The subvector x_1 can now be determined by solving a triangular system. Then x_2 can be determined by back substituting x_1 and solving a second triangular system.

Theorem 9.3. *Algorithm 8.1 constructs the same null basis N from any perfect matching $\mathcal{M}$ of M.*

Proof: From Theorems 9.1 and 9.2, the nonzeros in each column of N depends only on the Dulmage-Mendelsohn blocks of M. From Theorem 7.3, these blocks are unique, and hence independent of the perfect matching $\mathcal{M}$ of M. ∎

The sparsity of a fundamental null basis depends only on the partition of the columns of A into M and U. There are several strategies that might be used to find a partition that leads to a sparse N. One would be to minimize the total number of nonzeros in M. This can be done by assigning a weight to each column, equal to the number of nonzeros in the column. We now find a maximum cardinality matching of minimum weight.

A different strategy would be based on constructing M to have a specific structure. If M is block diagonal, then M^{-1} would be block diagonal also. Gilbert and Heath (1984) have experimented with this idea, finding such a submatrix M from a nested dissection of the matrix A^TA. We will report on the merits of these and other strategies after we have gained numerical experience with them.

10. A Myopic Algorithm

Algorithm 8.1, which may be used to approximately solve (NSP) for general matrices, constructs a fundamental null basis. However, sparsest null bases need not be fundamental. Restricting ourselves to such bases may cost us dearly in the sparsity of null bases. We seek to rectify that blemish of Algorithm 8.1 in this section.

We now describe an algorithm that constructs a null basis with an $(n-t)$-dimensional upper triangular submatrix. Choosing a null basis with this structure makes it easy to ensure that N has full column rank. We call this a *myopic algorithm* since it uses a short-sighted, locally greedy approach in constructing the basis. The following observation made in proving Theorem 3.2 is the basis for the algorithm.

Let $n(v)$ be a circuit containing an unmatched column v of A, with nonzeros in the column set C, and the row set R. Let A_{RC} be the submatrix of A defined by these sets.

From our discussion in Theorem 3.2, A_{RC} has the Strong Hall property. For any column $w \in C$, $A_{RC} - w$ has the Hall property, and hence a complete matching. If $w = v$, then the matching $\mathcal{M}$ remains unchanged. Else, we have the possibility of adding the column v to the set of matched columns M, in exchange of any column w which is in both the circuit $n(v)$ and M. We employ the locally greedy strategy of choosing w, the column deleted, to be the column of highest degree. (Other strategies are possible and will be discussed elsewhere.)

Algorithm 10.1.

1. [initialize] Let N be the empty set.
2. [match]
   Find a complete matching $\mathcal{M}$ of A;
   partition $A = (\, M \;\; U\,)$.
3. [augment null basis]
   for each $u \in U$ **do**
     construct $n(u)$ by step 2 of Algorithm 3.1;
   od
   Choose a smallest circuit $n(v)$ from those constructed;
   $N = (\, N \;\; n(v)\,)$.
4. [update matching]
   Choose a column w of maximum degree from the circuit $n(v)$;
   $U = U - v$; $M = M - w + v$;
   Go to 3.

For $i = 1, \ldots, n - t$, let w_i be the column deleted from A at the i-th step of the algorithm. It is now easily seen that N has the structure shown in Figure 10.1.

This algorithm requires more work than Algorithm 8.1. The worst case complexity of step 3 as described in the algorithm is $O(r\,(n - t)^2)$. There are several ways by which this can be reduced.

Let $d(u)$ denote the number of nonzeros in the unmatched column u. (This is also the number of edges incident on the column u in the bipartite graph of A.) Since each row adjacent to u is matched to a distinct column, the number of columns in the circuit $n(u)$ satisfies the equation $|n(u)| \geq d(u)$.

We order the unmatched columns such that $d(u)$ is non-decreasing. Consider the step when Algorithm 10.1 constructs all circuits $n(u)$ for $u \in U$ to decide which of them

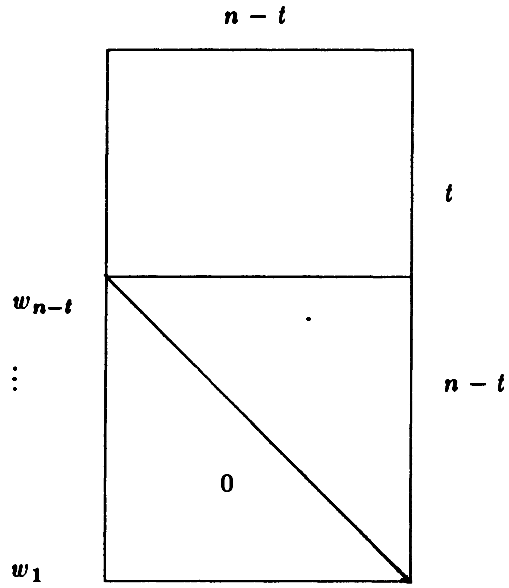

Figure 10.1.
A myopic null basis.

has smallest size. Let $n(v)$ be a smallest circuit obtained at some stage of this step. If $|n(v)| = k$, and $d(u) \geq k$ for all columns for which circuits have not been constructed yet, then $n(v)$ is the smallest circuit in step 3. In situations when the above strategy still constructs too many circuits, we can set an upper bound on the the number of circuits that may be constructed.

Unfortunately, it is now necessary to compute a numeric factorization of a submatrix of A for each null vector; graph condensation can not be used since the set of matched columns can change from step to step.

Our observation on the size of the circuit $n(u)$ may be used to derive a lower bound on the number of nonzeros in a null basis N. By summing the inequality on each unmatched column, we get

$$|N| \geq \sum_{u \in U} |n(u)|.$$

This is a novel technique for the construction of a sparse null basis which is prompted by our combinatorial approach to the (NSP). We classify all methods known to us for computing null bases in Section 13.

11. Coping with the Failure of wHP

In Sections 8 and 10, we presented algorithms to approximately solve (NSP) under the assumption of the weak Haar property. In this section we study how these algorithms should be modified when wHP fails. This is necessary since constraint matrices arising in optimization problems need not satisfy wHP. Theorems 3.2 and 3.3 imply that wHP is a 'benign' assumption, yet we need to exercise some caution in constructing null bases when it fails.

The following example makes our concern clear:

$$A = \begin{array}{c c} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}.$$

Assume that row 1 is matched to column a, and row 2 to column b. Hence $M = \{a, b\}$, and $U = \{c, d\}$. Since the size of the matching is 2, and the numeric rank of M is only 1, M does not have wHP. Algorithm 8.1 constructs the sets $n(c) = \{a, b, c\}$, and $n(d) = \{a, b, d\}$. These two sets contain the same circuit $\{a, b\}$ and no other, and the algorithm has failed to construct a structural null basis!

The remedy for this pathological situation is to make sure that a matching is chosen so that the set of matched columns M has numeric rank t. A close reading of Theorem 7.3 will convince the reader that it is enough to ensure that each Dulmage-Mendelsohn block of M has full numeric rank. Should a matching fail to do this, columns are exchanged between M and U to make this possible.

This rank deficiency can be detected at the stage when the LU factors of the Dulmage-Mendelsohn blocks of M are being computed. Linear dependence of columns in M results in a diagonal element of U becoming zero, or more likely in finite precision arithmetic, small. In the above example, column b is dependent on A, and the following situation results:

$$\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}.$$

We reject column b at this step, and replace it with column c.

Similar considerations arise when we consider numerical stability. If the second component of b was equal to $1 + \epsilon$ for some small ϵ, U will have a high condition number. Thus the null basis will be ill-conditioned. This situation is also avoided as above, since ill-conditioning can be detected by a diagonal element of U becoming small.

Each dependent set $n(u)$ contains exactly one unmatched column u. The other columns in $n(u)$ are from M. Since M has rank t, every subset of its columns has full rank. Hence, a dependent subset of $n(u)$ must contain u. Now, as in the case when wHP was true, the set of circuits obtained from the dependent sets $\{ n(u) \}$ forms a fundamental null basis.

In the previous example, we match row 2 to column c (row 1 is matched to column a as before), and now M has wHP. Algorithm 8.1 now yields the sets $n(b) = \{a, b, c\}$, and $n(d) = \{c, d\}$. The former does not have wHP, and it contains a circuit $\{a, b\}$; the latter is a circuit. These two circuits are now linearly independent, and so we do have a null basis N.

These observations lead us to

Theorem 11.1. *Let $\mathcal{M}$ be a complete matching of A, and let the set of matched columns M have numeric rank t. If $n(u)$ does not have wHP for some $u \in U$, at the numeric stage we obtain a sparser null basis than*

$$N = (n(u_1), \ldots , n(u_{n-t})).$$

∎

There might be (rather extreme) situations where it may not be possible to choose M to have rank t. Suppose $\text{rank}(M) = r < t$, and that this is the maximum possible.

Choose any r columns from M that yield a matching of cardinality r. We redefine M to be this set of r columns, and U to be the rest of the $n - r$ columns. For each $u \in U$, we use Algorithm 8.1 to construct the dependent set $n(u)$. Each $n(u)$ is either a circuit, or contains a circuit; the discussion preceding Theorem 11.1 now applies, and we get $n - r$ linearly independent circuits of A.

Again, an example might be helpful.

$$A = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}.$$

In the complete matching $\mathcal{M}$, let row 1 be matched to column a, and row 2 to column b; hence $M = \{a, b\}$, and $U = \{c, d\}$. However, $\text{rank}(M) = 1$, and this is the maximum possible. We redefine $M = \{a\}$, and $U = \{b, c, d\}$. The dependent set $n(u) = \{a, u\}$ for $u \in U$, and each one is a circuit linearly independent of the others.

A similar fix works for Algorithm 10.1. We make sure that the initial set of matched columns M has rank t. (If this is not possible, and M has maximum rank $r < t$, we choose only r columns to be in M, as before.) Exchange of columns in M is permitted at each step of the algorithm only if it does not reduce the rank. An analysis similar to the one for Algorithm 8.1 now shows that we do obtain a null basis of circuits at the numeric stage.

12. Orthogonal Null Bases

We now address the problem of constructing sparse null bases that have orthogonal columns. Such a basis N satisfies the equation

$$N^T N = I_{n-t},$$

where I_{n-t} is the $(n-t)$-dimensional identity matrix. This problem can be solved by adapting the algorithms we have developed.

Let A be a $t \times n$ matrix of rank t. Let n_1 be the first column of a null basis N which we construct by Algorithm 3.1. Modify A by appending a row:

$$\hat{A} = \begin{pmatrix} A \\ n_1^T \end{pmatrix}.$$

Apply Algorithm 3.1 to $\hat{A}$ to find a null vector n_2. By construction, n_2 is a null vector of A, and is orthogonal to n_1. Clearly this procedure can be continued until an orthogonal null basis is obtained.

We now indicate how Algorithm 8.1 should be modified to construct the orthogonal basis. Let $\mathcal{M}$ be a complete matching of A, and partition the columns of A into matched and unmatched columns as

$$A = (M \quad U).$$

For an unmatched column $u \in U$, construct the circuit $n(u)$. Add a row n_1^T to A which has nonzeros in those columns of A contained in $n(u)$. The updated matrix $\hat{A}$ is constructed as discussed before.

In the bipartite graph $G(A)$, we add a row vertex n_1 with edges to those columns of A contained in $n(u)$. The matching $\mathcal{M}$ is augmented by adding the edge (n_1, u) to it. We also update $\widehat{M} = M + u$, and $\hat{U} = U - u$. Algorithm 3.1 can now be applied to the graph $G(\hat{A})$.

The next theorem indicates how the Dulmage-Mendelsohn blocks (D-M blocks) of M change on augmentation to $\widehat{M}$.

Theorem 12.1. *Let the D-M blocks of M be $M_{11},\ldots,M_{kk}$. Let $K = \{1,\ldots,k\}$. Let C_i be the set of columns in M_{ii}. For a null vector $n(u)$, let*

$$I = \{\, i : C_i \subset n(u) \,\}.$$

$\widehat{M}$ has the D-M blocks $\{M_{jj}\}$ for each $j \in K \setminus I$, and $\cup_{i \in I} M_{ii}$.

Proof: Let the column sets $C_\ell \subset n(u)$ and $C_m \subset n(u)$. Then from Theorem 9.1, $u \xrightarrow{M} C_\ell$ and $u \xrightarrow{M} C_m$. By the rules for augmenting $G(A)$, n_1 becomes a row vertex, (n_1, u) becomes an edge of $\mathcal{M}$, and there are edges joining the row n_1 to each column in $C_\ell \cup C_m$. The sequence of edges

$$u \xrightarrow{M} C_\ell,\ C_\ell \xrightarrow{M} n_1,\ (n_1,u),\ u \xrightarrow{M} C_m,\ C_m \xrightarrow{M} n_1,\ (n_1,u)$$

is an $\mathcal{M}$-alternating tour between columns in C_ℓ and C_m. From the proof of Theorem 7.3, C_ℓ and C_m are in the same D-M block of $\widehat{M}$. ∎

In Section 4, we showed that a locally greedy algorithm that chooses a sparsest null vector at each step linearly independent of those chosen previously solves (NSP). We might suspect that a similar greedy strategy would construct a sparsest orthogonal null basis.

This suspicion is false. Consider the following counterexample:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & u & v & w & z \end{array} \\ \begin{pmatrix} \otimes & \times & 0 & 0 & \times & \times & \times & 0 \\ \times & \otimes & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \otimes & 0 & \times & \times & \times & \times \\ 0 & 0 & 0 & \otimes & 0 & 0 & 0 & \times \end{pmatrix} \end{array}.$$

The elements matched under $\mathcal{M}$ are circled. The set of matched columns $M = \{a,b,c,d\}$, and the set of unmatched columns $U = \{u,v,w,z\}$. A greedy strategy would choose at the first step the circuit $n(z) = \{c,d,z\}$. After updating the graph, the algorithm could choose, in order,

$$n(u) = \{a,b,c,d,z,u\},\quad n(v) = n(u) + v,\quad n(w) = n(v) + w.$$

Thus $|N| = 3 + 6 + 7 + 8 = 24$.

But it is possible to do better. If at the first step, we choose $n(u) = \{a, b, c, u\}$, then the algorithm could pick, in order,

$$n(v) = n(u) + v, \quad n(w) = n(v) + w, \quad n(z) = n(w) + d + z.$$

Now $|N| = 4 + 5 + 6 + 8 = 23$, which is a sparser null basis.

The greedy algorithm helped us to characterize sparsest null bases. This example shows that a greedy strategy may not find sparsest orthogonal null bases. Unfortunately, we do not know how to identify or construct the latter.

In view of Theorem 12.1, probably sparser orthogonal null bases may be obtained by the following strategy: We partition the columns of A into groups of $2t$ columns. For each partition, we construct an orthogonal basis by the algorithm described above. Additional columns of the null basis may be constructed by merging two partitions; at each such merger, the number of the columns in the partition of A doubles. So in about $\log_2 n$ steps, we obtain an orthogonal null basis of A.

13. Conclusions

Before summarizing our work, we classify the different methods we know about for constructing null bases. Heath, Plemmons, and Ward (1983) present a different classification scheme.

1. Variable-reduction.

Wolfe (1962) proposed partitioning the matrix A as

$$A = (A_1 \quad A_2), \tag{1}$$

and constructing a null basis of the form

$$N = \begin{pmatrix} -A_1^{-1}A_2 \\ I_{n-t} \end{pmatrix}, \tag{2}$$

where A_1 is chosen to be nonsingular, if necessary by permuting the columns of A.

There are three techniques that may be used to choose the submatrix A_1: Gauss-Jordan elimination, row elimination, and orthogonal row elimination. It might be surprising when we claim that these techniques are equivalent as far as sparsity of N is concerned. We show the equivalence by displaying the appropriate submatrices A_1 and A_2.

In the rest of this section, P will denote a permutation matrix, L a lower triangular matrix, U and R upper triangular matrices, Q an orthogonal matrix, and I an identity matrix. Each symbol is defined only locally for a particular method. For instance, L does not stand for the same lower triangular matrix throughout this section.

a. Gauss-Jordan Elimination.

$$P_1 A P_2 = B\ (I_t \quad C),$$
$$P_2 N = \begin{pmatrix} -C \\ I_{n-t} \end{pmatrix}.$$

Let the first t columns of the permuted matrix A be denoted by $\tilde{A}_1$ and the last $n-t$ columns by $\tilde{A}_2$. Then B is clearly $\tilde{A}_1$, and C is $\tilde{A}_1^{-1}\tilde{A}_2$.

b. Row Elimination.

$$L P_1 A P_2 = (U_1 \quad U_2),$$
$$P_2 N = \begin{pmatrix} -U_1{}^{-1} U_2 \\ I_{n-t} \end{pmatrix},$$

where U_2 is not upper triangular in general.

We show that the null basis in (b) can be obtained by choosing an appropriate partition of A as in equation (1).

Let

$$P_1 A P_2 = (\tilde{A}_1 \quad \tilde{A}_2).$$

Now

$$U_1{}^{-1} U_2 = U_1{}^{-1} L\, L^{-1} U_2 = (L^{-1} U_1)^{-1}\ (L^{-1} U_2) = \tilde{A}_1^{-1}\tilde{A}_2.$$

Conversely, let A be partitioned as in equation (1). Let

$$\tilde{L} A_1 = U_1, \quad \text{and} \quad \tilde{L} A_2 \equiv U_2.$$

Now

$$A_1^{-1} A_2 = A_1^{-1} \tilde{L}^{-1} \tilde{L} A_2 = \left(\tilde{L} A_1\right)^{-1} \left(\tilde{L} A_2\right) = U_1^{-1} U_2.$$

Hence, row elimination constructs a null basis obtainable by a partition of A as in equation (2).

c. Orthogonal Row Elimination.

$$Q\,P_1 A\,P_2 = (\,R_1 \quad R_2\,),$$
$$P_2 N = \begin{pmatrix} -R_1^{-1} R_2 \\ I_{n-t} \end{pmatrix},$$

where R_2 is not upper triangular in general. As before, we can show this method is equivalent to a partitioning of A.

The combinatorial method we propose in Section 8 finds the submatrix A_1 by finding a complete matching. It then partitions A_1 into its Dulmage-Mendelsohn blocks. The $L\,U$ or $Q\,R$ factorization of these blocks may be used to compute N. We have the freedom here to choose A_1 so that N is sparse. Also, the block decomposition reduces the work needed to compute the bases.

2. Column Elimination.

$$\begin{aligned} P_1 A\,P_2\,U &= (\,L \quad 0\,), \\ U &= (\,U_1 \quad U_2\,), \\ P_2 N &= U_2, \end{aligned}$$

where U_1 is an $n \times t$ matrix and U_2 is an $n \times (n-t)$ matrix.

3. Orthogonal Column Elimination.

$$\begin{aligned} P_1 A\,P_2\,Q &= (\,L \quad 0\,), \\ Q &= (\,Y \quad Z\,), \\ P_2 N &= Z, \end{aligned}$$

where Y is an $n \times t$ orthogonal matrix and Z is an $n \times (n-t)$ orthogonal matrix.

4. Singular Value Decomposition.

$$\begin{aligned} V\,P_1 A\,P_2\,W &= (\,\Sigma \quad 0\,), \\ W &= (\,W_1 \quad W_2\,), \\ P_2 N &= W_2, \end{aligned}$$

where W_1 is an $n \times t$ orthogonal matrix, and W_2 is an $n \times (n-t)$ orthogonal matrix. For most large sparse matrices, this method is prohibitively expensive.

5. Generalized Variable-reduction.

This is a new method motivated by the myopic algorithm presented in this paper.

$$\begin{aligned} P_1 A P_2 &= (A_1 \quad A_2), \\ P_2 N &= \begin{pmatrix} -A_1^{-1} A_2 U \\ U \end{pmatrix}, \end{aligned} \tag{3}$$

where U is an $(n-t)$-dimensional upper triangular matrix.

It is possible to modify this null basis to get a sparser basis. The null vectors in N are not necessarily circuits, unlike the fundamental null basis constructed by variable-reduction. The nonzeros in a null vector pick out a linear combination of a dependent set of columns of A. It is now possible to discard some columns of A from this set (thereby turning nonzeros in a null vector into zeros) to get a circuit. In some situations, it is possible to obtain sparser null bases than fundamental bases by this method.

The myopic algorithm constructs a null basis of this form. For a matrix A with the weak Haar property, it constructs null vectors that are circuits, so that the modification of the basis is done combinatorially. Also, the upper triangular matrix U is constructed as the algorithm proceeds, so that in general we do not know its structure.

We remark that instead of an upper triangular matrix U, we can choose any $(n-t) \times (n-t)$ nonsingular matrix B in the generalized variable-reduction method. But then modifying the null vectors for sparsity, and yet ensuring that we retain a null basis becomes tricky. When an upper triangular matrix is used, this is easy: We ensure that the modified N has the upper triangular submatrix of dimension $n-t$.

We now consider an example to show how this technique can be used to construct sparser bases than fundamental null bases.

Let

$$A = \begin{array}{c} \begin{array}{cccccccccc} a & b & c & d & e & f & g & h & i & j \end{array} \\ \begin{pmatrix} \times & \times & \times & \times & 0 & 0 & 0 & 0 & 0 & 0 \\ \times & 0 & 0 & 0 & \times & \times & \times & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & \times & 0 & 0 & \times & \times & 0 \\ 0 & 0 & \times & 0 & 0 & \times & 0 & \times & 0 & \times \\ 0 & 0 & 0 & \times & 0 & 0 & \times & 0 & \times & \times \end{pmatrix} \end{array},$$

and partition it as $A_1 = \{a, b, c, d, e\}$, and $A_2 = \{f, g, h, i, j\}$.

The fundamental null basis in equation (2) has the structure

$$N_1 = \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & 0 & \times & 0 & \times \\ 0 & \times & 0 & \times & \times \\ \times & \times & \times & \times & \times \\ \times & 0 & 0 & 0 & 0 \\ 0 & \times & 0 & 0 & 0 \\ 0 & 0 & \times & 0 & 0 \\ 0 & 0 & 0 & \times & 0 \\ 0 & 0 & 0 & 0 & \times \end{pmatrix}.$$

For the same partition of A_1 and A_2, and the choice of

$$U = \begin{pmatrix} \times & 0 & 0 & 0 & \times \\ 0 & \times & 0 & 0 & \times \\ 0 & 0 & \times & 0 & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{pmatrix},$$

a null basis N_2 given by the equation (3) has the same structure as N_1 except for the last column. This column has nonzeros in all its row positions. But note that the last null vector in N_2 is not a circuit. Since the set of columns $\{f, g, h, i, j\}$ is a circuit, replace the last column of N_2 with the vector

$$(0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \times \quad \times \quad \times \quad \times \quad \times)^T$$

to get a null basis N_3. This basis has the structure of a null basis constructed by the myopic algorithm; further, it is sparser than N_1.

We have shown that it is possible to find null vectors of a matrix from its structure. Bipartite matching theory can then be used to compute circuits. Only circuits can be columns of sparsest null bases, and a sparsest basis can be constructed a column at a time, in a greedy fashion. Nevertheless, finding a sparsest null basis is NP-hard, as is the problem of finding a sparsest fundamental null basis. We propose two classes of approximation algorithms to construct sparse null bases. Algorithm 8.1 constructs fundamental bases, while Algorithm 10.1 finds a basis with an upper triangular submatrix. When computing fundamental null bases, it is possible to use the Dulmage-Mendelsohn block structure to reduce the work required. No such simplification is possible when Algorithm 10.1 is used, though it can construct potentially sparser null bases.

We are currently implementing these algorithms to see how they perform in practice. We hope these ideas will lead to more efficient algorithms for large-scale numerical optimization problems.

This paper has dealt with the construction of sparse *explicit null bases*, when the nonzeros in the basis are stored. A different view point is to construct sparse *implicit null bases*. Here the basis is not computed directly, but is stored as a sequence of transformations needed to compute it. For instance, in the LQ decomposition of the matrix A,

$$A\,Q = (\,L \quad 0\,),$$

the sequence of Givens transformations used to compute Q is stored. A sparse basis in this situation would compute Q in a small number of rotations. Finding such bases remains an important open research problem. In these and other directions, we will continue our work on (NSP).

Appendix: (NSP) and the Resolution of a Paradox

Hoffman and McCormick (1982) have proposed a polynomial time algorithm to transform a matrix with the matching property (MP) to an equivalent, optimally sparse matrix. Let A be a $t \times n$ matrix, with $t < n$. They construct a nonsingular matrix $\tilde{T}$, such that $\tilde{A} = \tilde{T} A$ is a sparsest matrix over all possible nonsingular transformations T. Furthermore, they observed that if the columns of A can be permuted so that

$$A\,P = (\,\hat{A} \quad D\,)$$

for some diagonal matrix D, and permutation matrix P, then A is already optimally sparse. (In practice, MP may not be satisfied; the proposed algorithm can still be applied, but with no guarantee of optimality.)

This suggests the following strategy to solve (NSP): Determine a permutation matrix P such that

$$A\,P = (\,A_1 \quad A_2\,),$$

where A_1 is non-singular. A null basis of A is then given by $P\,N$, where

$$N = \begin{pmatrix} -A_1^{-1} A_2 \\ I \end{pmatrix}.$$

If N^T has MP, it is an optimally sparse null basis. This result is surprising, since the partition induced by P is arbitrary except for ensuring that A_1 is non-singular. Indeed, let $\tilde{P}$ be another permutation matrix such that

$$A\tilde{P} = (\tilde{A}_1 \quad \tilde{A}_2),$$

where $\tilde{A}_1$ is non-singular. Then another null basis of A is given by $\tilde{P}\tilde{N}$, where

$$\tilde{N} = \begin{pmatrix} -\tilde{A}_1^{-1}\tilde{A}_2 \\ I \end{pmatrix}.$$

Now, $\tilde{N}^T$ is also optimally sparse if it satisfies MP. This leads us to an apparent contradiction since $N^T P^T = T\tilde{N}^T\tilde{P}^T$ for some nonsingular transformation T. Yet, the sparsities of N and $\tilde{N}$ may differ dramatically. How then can both PN and $\tilde{P}\tilde{N}$ be optimally sparse null bases of A ?

The answer is that either N and $\tilde{N}$ have the same number of nonzeros, or *MP must fail to hold for all possible values of the nonzeros of A*. In this section, we study the latter possibility, which is illustrated by the following example. Recall that $m(A)$, the matching number of A, is the cardinality of a maximum matching of A.

Suppose the matrix A is partitioned as $A = (A_1 A_2)$, and that the structures of the submatrices are

$$A_1 = \begin{pmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ 0 & \times & \times \end{pmatrix}, \quad \text{and} \quad A_2 = \begin{pmatrix} \times & \times & \times \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{1}$$

For any assignment of values to the nonzeros of A, the matrix $A_1^{-1}A_2$ is dense, and $m\left(A_1^{-1}A_2\right) = 3$. The structures are

$$A_1^{-1} = \begin{pmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ \times & \times & \times \end{pmatrix}, \quad \text{and} \quad A_1^{-1}A_2 = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix}.$$

But each column of the latter matrix is a multiple of the first column of A_1^{-1}, so its rank is at most 1. Therefore, $A_1^{-1}A_2$ does not have MP for *any assignment of values to the nonzeros of A*.

We now generalize the above example. Consider the matrix product $AB = C$, where A has n columns and B has n rows (associate A_1^{-1} with A and A_2 with B). We exhibit a general class of structures for A and B such that C *does not have MP for any values of A and B.*

Let $G_{A \cdot B} = (VA, VR, VB)$ be a 'doubly bipartite' graph: VA is a set of vertices corresponding to columns of A^T, VB is a set of vertices corresponding to columns of B, and VR is the set of vertices corresponding to the n rows of B and the n rows of A^T. There is an edge between $a \in VA$ and $r \in VR$ if (r, a) is a nonzero of A^T. Similarly, there is an edge between $b \in VB$ and $r \in VR$ if (r, b) is a nonzero of B. For example, if

$$A = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \!\! \begin{array}{c} \begin{array}{ccc} r_1 & r_2 & r_3 \end{array} \\ \begin{pmatrix} \times & 0 & 0 \\ \times & \times & 0 \\ 0 & \times & \times \end{pmatrix} \end{array}, \quad \text{and} \quad B = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \end{array} \!\! \begin{array}{c} \begin{array}{ccc} b_1 & b_2 & b_3 \end{array} \\ \begin{pmatrix} \times & \times & \times \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array},$$

then $G_{A \cdot B}$ is as shown in Figure 1.

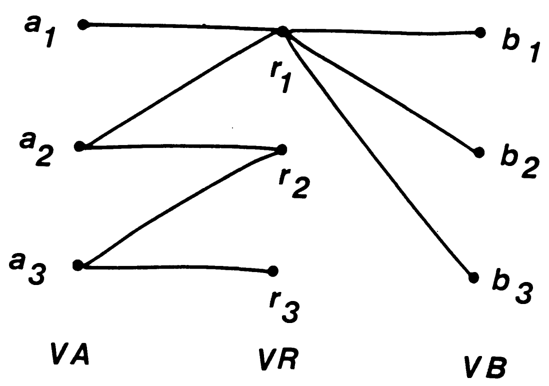

Figure 1.
The doubly bipartite graph $G_{A \cdot B}$

The graph $G_{A \cdot B}$ is called the concatenation graph by Doob (1984). Let the matrix C_{XY} denote the submatrix of C induced by the row set X and the column set Y. This matrix has a nonzero in position (i, j) when a column a_i^T of A^T, and column b_j of B both have nonzeros in the same row position, say r_k. Equivalently, the bipartite graph $G(C_{XY})$ has the edge (a_i, b_j) if and only if there is a path of length 2 from a_i to b_j in the doubly bipartite graph $G_{A \cdot B}$. Hence, the structure of C_{XY} is easily obtained from $G_{A \cdot B}$.

We denote by $N(S)$ the neighbor set of a vertex set S in $G_{A \cdot B}$. A set $X \subseteq VA$ has nonzeros only in the row set $N(X)$, and similarly, $Y \subseteq VB$ has nonzeros only in $N(Y)$. Let $Z \equiv N(X) \cap N(Y)$. Z corresponds to those row positions of VR where the sets X and Y have nonzero overlap. Nonzero elements in C_{XY} arise only from the rows in Z of the column sets X of A^T, and Y of B.

The following result leads to a structural condition on A and B under which the product matrix C has a numerically rank deficient submatrix, C_{XY}, for all values of A and B.

Lemma 1. *Let $X \subseteq VA$, $Y \subseteq VB$. Then*

$$\text{rank}\,(C_{XY}) \leq \min \;\{ |N(X) \cap N(Y)|, |X|, |Y| \}.$$

Proof: As before, let $Z \equiv N(X) \cap N(Y)$. If $|Z| < |X| \leq |Y|$, then each of the $|X|$ rows of C_{XY} is a linear combination of the same $|Z|$ rows of B. If $|Z| < |Y| \leq |X|$, then each of the $|Y|$ columns of C_{XY} is a linear combination of the same $|Z|$ columns of A. The result follows. ∎

From Lemma 1, we conclude that if X and Y have fewer common neighbors in VR than both $|X|$ and $|Y|$, then C_{XY} is numerically rank deficient. This deficiency is independent of the actual values of the nonzeros of A and B, since the condition forces either the rows or columns of C_{XY} to be *algebraically dependent*. In the previous example, let $X = VA$, $Y = VB$. Then $|X| = 3$, $|Y| = 3$, and $|Z| = 1$, and hence C has rank at most 1.

Of course, even if a submatrix of C is rank deficient for all values of A and B, the matrix C could have MP—the condition $m\,(C_{XY}) = \text{rank}\,(C_{XY})$ can still hold. Consider the same example with A_1 replaced by

$$\tilde{A}_1 = \begin{pmatrix} \times & 0 & 0 \\ 0 & \times & 0 \\ 0 & \times & \times \end{pmatrix}.$$

The matrix $\tilde{A}_1^{-1} A_2$ consists of zeros except for the first row, which is dense. Hence $m\,(\tilde{A}_1^{-1} A_2) = \text{rank}\,(\tilde{A}_1^{-1} A_2) = 1$.

We now state sufficient conditions for the matching number of a submatrix of C to be greater than its rank. We display a general class of structures (of A and B) for which the product matrices C do not possess MP, for any assignment of values to A and B. For $X \subseteq VA$, and $Y \subseteq VB$, let $N^2(X)$ denote $N(N(X)) \cap Y$ (i.e., those vertices in Y reachable from X via paths of length 2). Similarly we define the set $N^2(Y)$.

In what follows, we make use of the following result: If $|X| \leq |Y|$, then the matching number of C_{XY}, $m(C_{XY})$, satisfies

$$m(C_{XY}) = |X| - \max_{\tilde{X} \subseteq X} \{|\tilde{X}| - |N^2(\tilde{X})|\}. \tag{2}$$

This result is a consequence of

Theorem 2. *(König's Theorem) Let $G(X, Y, E)$ be a bipartite graph. Its matching number is*

$$m(G) = |X| - \max_{W \subseteq X} \{|W| - |N(W)|\},$$

where $N(W)$ is the neighbor set of W in G.

Berge (1973) contains a proof. ∎

From the relationship of $G(C_{XY})$ to $G_{A \cdot B}$ that we have observed previously, for a vertex set $W \subseteq X$, the set $N(W)$ in the former graph is $N^2(W)$ in the latter. Equation (2) now follows from Theorem 2.

Theorem 3. *If for some $X \subseteq VA$, $Y \subseteq VB$, either*

$$(i) \quad |X| \leq |Y|, \quad \text{and} \quad |X| - \max_{\tilde{X} \subseteq X} \{|\tilde{X}| - |N^2(\tilde{X})|\} > |N(X) \cap N(Y)|,$$

or

$$(ii) \quad |Y| \leq |X|, \quad \text{and} \quad |Y| - \max_{\tilde{Y} \subseteq Y} \{|\tilde{Y}| - |N^2(\tilde{Y})|\} > |N(X) \cap N(Y)|,$$

then C does not possess MP for any assignment of values to A and B.

Proof: Together with Lemma 1, and equation (2), conditions (i) or (ii) imply that

$$m(C_{XY}) > \operatorname{rank}(C_{XY})$$

for all values of A and B. ∎

The conditions mentioned in the above theorem are not pathological. Indeed, we expect them to be satisfied usually. Since these remarks apply directly to (NSP) with A replaced with A_1^{-1}, and B replaced with A_2, it follows that one cannot generally expect the matrix $A_1^{-1}A_2$ to have MP.

One final observation: If A_1^{-1} is dense, and A_2 has every subset of its columns incident to at least as many rows, then PN is an optimally sparse null basis for some values of A. Here

$$N = \begin{pmatrix} -A_1^{-1}A_2 \\ I_{n-t} \end{pmatrix}.$$

We can assign values to the nonzeros of A such that $A_1^{-1}A_2$ has MP.

Rank Deficiency and The Hall Property

The condition $|Z| \geq \min\{|X|, |Y|\}$ can be viewed as a generalized Hall Property. Let V be the set of rows, and W the columns of a matrix A. Also let $|V| \leq |W|$. A has the Hall Property if every subset $X \subseteq V$ of its rows has nonzeros in at least as many columns. Hence $|N(X)| \geq |X|$.

It is easy to see that if A does not have the Hall Property, then A is structurally rank deficient. Equivalently, a principal submatrix of AA^T is numerically rank deficient for all values of A (the rows (columns) of this principal submatrix are algebraically dependent). But this is just a special case of Lemma 1, with $B = A^T$, where only principal submatrices are considered. To see this, note that $B = A^T$ implies that $G(VB, VR)$ is a copy of $G(VA, VR)$; C_{XY} is a principal submatrix if and only if Y is the mirror image of X. Therefore, the condition $|Z| \geq \min \{|X|, |Y|\}$ is equivalent in this situation to $|N(X)| \geq |X|$.

Finally, we observe that if two compatible matrices A and B have the generalized Hall property,

$$|N(X) \cap N(Y)| \geq \min \{|X|, |Y|\}, \quad \text{for all} \quad X \subseteq VA, \quad Y \subseteq VB, \tag{3}$$

then the matrix product C is structurally dense. Else, suppose that for some i and j, $C_{ij} = 0$, for all values of A, B. Then $|N(a_i) \cap N(b_j)| = 0$, which violates (3), if we let $X = \{a_i\}$, and $Y = \{b_j\}$.

Acknowledgements

We wish to thank Professors John Gilbert and Bob Bland for several valuable discussions and helpful suggestions.

References

Berge, Claude (1973)
Graphs and Hypergraphs.
Translated from the French by Edward Minieka.
North-Holland Publishing Company, Amsterdam.

Bondy, J. A. and U. S. R. Murty (1976)
Graph Theory with Applications.
American Elsevier Publishing Company, New York.

Bose, R. C. and T. A. Dowling (1969), editors.
Combinatorial Mathematics and its Applications.
University of North Carolina Press, Chapel Hill, NC.

Bunch, James R. and Donald J. Rose (1976), editors.
Sparse Matrix Computations.
Academic Press.

Camion, P. (1968)
Modules Unimodulaires.
Journal of Combinatorial Theory **4**: 301-362.

Cheney, E. W. (1966)
Introduction to Approximation Theory.
McGraw Hill Book Company, New York.

Christofides, Nicos (1975)
Graph Theory: An Algorithmic Approach.
Academic Press.

Coleman, Thomas F., Anders Edenbrandt, and John R. Gilbert (1983)
Predicting Fill for Sparse Orthogonal Factorization.
Cornell University Department of Computer Science Technical Report TR 83-578.

Coleman, Thomas F. (1984)
Large Sparse Numerical Optimization.
Lecture Notes in Computer Science, Volume 165 (G. Goos and J. Hartmanis, editors).
Springer-Verlag.

Dantzig, G. B. and A. F. Veinott (1968), editors.
Mathematics of the Decision Sciences: Lectures in Applied Mathematics, Volume 2.
American Mathematical Society, Providence, RI.

Deo, Narsingh, G. M. Prabhu, and M. S. Krishnamoorthy (1982)
Algorithms for Generating Fundamental Cycles in a Graph.
ACM Transactions on Mathematical Software **8**: 26-42.

Doob, Michael (1984)
Applications of Graph Theory in Linear Algebra.
Mathematics Magazine **57**: 67-76.

Duff, Iain S. and J. K. Reid (1978)
An Implementation of Tarjan's Algorithm for the Block Lower Triangularization of a Matrix.
ACM Transactions on Mathematical Software **4**: 137-147.

Duff, Iain S. (1977)
A Survey of Sparse Matrix Research.
Proceedings of the IEEE **65**: 500-535.

Dulmage, A. L. and N. S. Mendelsohn (1958)
Coverings of Bipartite Graphs.
Canadian Journal of Mathematics **10**: 517-534.

Dulmage, A. L. and N. S. Mendelsohn (1959)
A Structure Theory of Bipartite Graphs of Finite Exterior Dimension.
Transactions of the Royal Society of Canada Section III **53**: 1-13.

Dulmage, A. L. and N. S. Mendelsohn (1963)
Two Algorithms for Bipartite Graphs.
Journal of the Society for Industrial and Applied Mathematics **11**: 183-194.

Dulmage, A. L. and N. S. Mendelsohn (1967)
Graphs and Matrices.
In Harary (1967), pages 161-227.

Edmonds, J. (1971)
Matroids and the Greedy Algorithm.
Mathematical Programming **1**: 127-136.

Fulkerson, D.R. (1968)
Networks, Frames, Blocking Systems.
In Dantzig and Veinott (1968), pages 303-334.

Gale, D. (1968)
Optimal Assignments in an Ordered Set: An Application of Matroid Theory.
Journal of Combinatorial Theory **4**: 176-180.

Garey, M. R. and D. S. Johnson (1979)
Computers and Intractability: A Guide to the Theory of NP-Completeness.
W. H. Freeman and Company, San Francisco, CA.

George, J. A. and F. G. Gustavson (1980)
A New Proof on Permuting to Block Triangular Form.
IBM Research Report RC 8238.
IBM Thomas J. Watson Research Center, Yorktown Heights, NY.

Gilbert, John R. and Michael T. Heath (1984)
Personal communication.

Gill, Philip E., Walter Murray, and Margaret H. Wright (1981)
Practical Optimization.
Academic Press.

Gustavson, Fred (1976)
Finding the Block Lower Triangular Form of a Sparse Matrix.
In Bunch and Rose (1976), pages 275-289.

Harary, Frank (1967), editor.
Graph Theory and Theoretical Physics.
Academic Press.

Heath, Michael T., Robert J. Plemmons, and Robert C. Ward (1983)
Sparse Orthogonal Schemes for Structural Optimization Using the Force Method.
Oak Ridge National Laboratory Technical Report ORNL/CSD-119, Oak Ridge, TN.

Hoffman, A. J. and S. T. McCormick (1982)
A Fast Algorithm that Makes Matrices Optimally Sparse.
Stanford University Systems Optimization Laboratory Report 82-13, Stanford, CA.

Hopcroft, J. E. and R. M. Karp (1973)
A $n^{2.5}$ Algorithm for Maximum Matching in Bipartite Graphs.
SIAM Journal on Computing **2**: 225-231.

Howell, Thomas D. (1976)
Partitioning Using PAQ.
In Bunch and Rose (1976), pages 23-37.

Itai, Alon and Michael Rodeh (1977)
Finding a Minimum Circuit in a Graph.
Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, pages 1-10.

Johnson, D. S., J. K. Lenstra, and A. H. G. Rinnooy Kan (1978)
The Complexity of the Network Design Problem.
Networks **8**: 279-285.

Johnson, Diane M., A. L. Dulmage, and N. S. Mendelsohn (1962)
Connectivity and Reducibility of Graphs.
Canadian Journal of Mathematics **14**: 529-539.

Knuth, Donald E. (1974)
Random Matroids.
Stanford University Department of Computer Science Technical Report STAN-CS-74-453, Stanford, CA.

Lawler, Eugene L. (1976)
Combinatorial Optimization: Networks and Matroids.
Holt, Rinehart, and Winston, New York.

McCormick, S. Thomas (1983)
A Combinatorial Approach to Some Sparse Matrix Problems.
Ph. D. Thesis, Stanford University.
Stanford University Systems Optimization Laboratory Technical Report SOL 83-5, Stanford, CA.

Papadimitriou, Christos H. and Kenneth Steiglitz (1982)
Combinatorial Optimization: Algorithms and Complexity.
Prentice-Hall, Englewood Cliffs, NJ.

Pothen, Alex (1984)
Sparse Null Bases and Marriage Theorems.
Ph.D. Thesis, Cornell University, Ithaca, NY.

Rado, R. (1957)
Note on Independence Functions.
Proceedings of the London Mathematical Society **37**: 351-353.

Rockefellar, R.T. (1969)
The Elementary Vectors of Subspaces of $\Re^n$.
In Bose and Dowling (1969), pages 104-127.

Seymour, P. D. (1980)
Decomposition of Regular Matroids.
Journal of Combinatorial Theory Series B **28**: 305-359.

Stepanets, G. F. (1964)
Basis Systems of Vector Cycles with Extremal Properties in Graphs.
Upsekhi. Mat. Nauk. **19**: 2, 171-175.

Turner, James and William H. Kautz (1970)
A Survey of Progress in Graph Theory in the Sóviet Union.
SIAM Review **12**: 1-68.

Welsh, D. J. A. (1976)
Matroid Theory.
Academic Press, London.

Whitney, Hassler (1935)
On the Abstract Properties of Linear Dependence.
American Journal of Mathematics **57**: 509-533.

Wolfe, Philip (1962)
The Reduced Gradient Method.
Unpublished manuscript, The RAND Corporation.