Obtaining triangular diagonal blocks in sparse matrices using cutsets

Tuğrul Dayar

Department of Computer Engineering, Bilkent University, TR-06800 Bilkent, Ankara, Turkey

tugrul@cs.bilkent.edu.tr

The paper proposes an algorithm which computes a (2×2) block partition of an irreducible, sparse matrix with a zero-free diagonal so that one of the diagonal blocks is triangular. The algorithm works by computing a cutset of the directed graph associated with the off-diagonal part of the matrix and is linear in the number of nonzeros therein. The proposed algorithm is then extended to reducible, sparse matrices with a zero-free diagonal. Experiments on benchmark unsymmetric matrices show that, in many cases the order of the triangular block can be increased by using another algorithm for computing cutsets from the literature, which is based on a greedy randomized adaptive search procedure; however, this is not efficient timewise unless the matrix is relatively small. A block iterative solver based on the partition returned by the proposed algorithm is compared with an industrial strength direct solver for time, space, and accuracy. Results indicate that there are cases in which it is advantageous to compute and use block partitions based on cutsets.

Keywords: sparse matrices; triangular blocks; cutsets.

1 Introduction

Sparse matrices are characterized by a large percentage of zero elements. Such matrices arise in many application areas and their processing is an important subject of study in numerical linear algebra. The books by Duff, Erisman & Reid (1986) and Saad (2003) discuss, respectively, direct and iterative methods for solving linear systems of equations that have sparse coefficient matrices. Since we consider sparse matrices throughout this work, we drop the word sparse before matrix and refer to a sparse matrix as a matrix. Now, let us recall that a matrix which can be permuted to block triangular form with two or more blocks along the diagonal is called *reducible*; if this is not possible,

then the matrix is said to be *irreducible* (Duff et al., 1986, p. 105). A matrix that is *structurally nonsingular* can be permuted to have a zero-free diagonal, meaning the matrix cannot be singular for all numerical values of its nonzero elements (Duff et al., 1986, p. 107). We remark that not all matrices are irreducible and not all matrices have (or can be permuted to have) a zero-free diagonal.

Now, let A be a given $(n_A \times n_A)$ irreducible matrix with nz_A nonzero elements and a zero-free diagonal. Without loss of generality, the problem is to compute a permutation matrix Q such that

$$QAQ^T = \begin{array}{c} \\ n_T \\ n_C \end{array} \begin{array}{c} \begin{array}{cc} n_T & n_C \end{array} \\ \begin{pmatrix} T & X \\ Y & C \end{pmatrix} \end{array}, \tag{1.1}$$

where $n_A = n_T + n_C$ and T is an $(n_T \times n_T)$ triangular submatrix. Hence, Q together with n_T defines a symmetric permutation on A yielding a (2×2) block partition in which the first diagonal block of order n_T is triangular and invertible. The solution of a linear system having the triangular block as coefficient matrix follows from back or forward *substitution* (Duff et al., 1986, p. 42) depending on the shape of triangularity.

In graph theoretic terms (Duff et al., 1986, p. 2), let $\mathcal{V} = \{1, 2, \ldots, n_A\}$ be the *node* (or *vertex*) *set* and $\mathcal{E} = \{(i,j) \mid a_{i,j} \neq 0, i \neq j, \text{ and } i,j \in \mathcal{V}\}$ be the *edge* (or *arc*) *set* of the *directed graph* (*digraph*) $G(\mathcal{V}, \mathcal{E})$ associated with the off-diagonal part of A. The irreducibility of A translates to the *strong connectedness* of $G(\mathcal{V}, \mathcal{E})$ (Duff et al., 1986, p. 114), that is, the reachability of each node by following a sequence of edges from every other node in $\mathcal{V}$. The problem then becomes one of computing a *cutset* (or *feedback vertex set*, Festa, Pardalos & Resende (1999, p. 209)) $\mathcal{C} \subset \mathcal{V}$ whose elements cut all the cycles in $G(\mathcal{V}, \mathcal{E})$. If $\mathcal{C}$ and the edges incident on $\mathcal{C}$ are removed from $G(\mathcal{V}, \mathcal{E})$ to give the node set $\mathcal{T} = \mathcal{V} - \mathcal{C}$, then the resulting subgraph should become *acyclic* (Duff et al., 1986, p. 115). In (1.1), the submatrix corresponding to this acyclic subgraph is denoted by the triangular matrix T. The smaller $n_C = |\mathcal{C}|$ is, the larger $n_T = |\mathcal{T}|$, and in general, the number of nonzeros, nz_T, in T become. The objective is then to compute as large and as fast a $\mathcal{T}$ as possible. Observe that if the diagonal elements of A are not omitted when forming $G(\mathcal{V}, \mathcal{E})$, we would obtain $\mathcal{C} = \mathcal{V}$ since each diagonal element in A represents a different cycle which needs to be cut.

When the coefficient matrix at hand is reducible, but can be permuted to block triangular form with irreducible diagonal blocks having zero-free diagonals, it is possible to compute (2×2) block partitions of the diagonal blocks (of order larger than 1) as in (1.1). Diagonal blocks of order 1 can be considered to be already triangular. Then the triangular diagonal blocks can be grouped into one larger triangular diagonal block and the remaining non-triangular diagonal blocks can be grouped

into a second larger diagonal block, implying once again a (2×2) block partition as in (1.1) with a zero-free diagonal in which one of the diagonal blocks is triangular. In this paper, we propose an algorithm which computes such a (2×2) block partition. We compare the order of the triangular block computed by the proposed algorithm with that computed by a greedy randomized adaptive search procedure on a number of benchmark matrices. Noticing that one use of such a (2×2) block partition is in *block iterative methods* (Saad, 2003, pp. 106–110), we compare a block iterative solver based on the partition returned by the proposed algorithm with an industrial strength direct solver for time, space, and accuracy. To the best of our knowledge, our work is the first which undertakes a study of cutsets on graphs associated with matrices.

In the next section, through a small example we provide background on the solution to the problem of computing cutsets for graphs that have appeared in other areas and highlight two algorithms that can be used to obtain triangular diagonal blocks in irreducible matrices with a zero-free diagonal. In section 3, we propose an algorithm for reducible matrices with a zero-free diagonal after extending one of these algorithms and discuss implementation issues. In section 4, we compare the two algorithms introduced in section 2 on benchmark matrices and provide the results of experiments with a direct solver versus a block iterative solver using the proposed algorithm. The paper ends with concluding remarks in the last section.

2 Background

We start by recalling the concept of a *cycle* in $G(\mathcal{V}, \mathcal{E})$ as a sequence of edges $(v_1, v_2), (v_2, v_3)$, $\ldots, (v_{k-1}, v_k)$ with $v_1 = v_k$, where $v_i \in \mathcal{V}$ for $i \in \{1, 2, \ldots, k\}$, v_i are distinct except v_1 and v_k, and $(v_j, v_{j+1}) \in \mathcal{E}$ for $j \in \{1, 2, \ldots, k-1\}$. In the following, we represent such a cycle, which is sometimes called a simple cycle, as $(v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \cdots \rightarrow v_{k-1} \rightarrow v_k)$ and remark that the length of the cycle is $(k-1)$ and needs to be less than or equal to n_A.

Example. Consider the matrix

$$
A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array}
\begin{array}{c}
\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\
\left(\begin{array}{cccccccc}
\mathrm{X} & & & & \mathrm{X} & & & \\
 & \mathrm{X} & & & & & & \mathrm{X} \\
\mathrm{X} & & \mathrm{X} & & & & & \\
 & & & \mathrm{X} & \mathrm{X} & \mathrm{X} & & \\
 & & & & \mathrm{X} & \mathrm{X} & \mathrm{X} & \\
 & \mathrm{X} & & \mathrm{X} & & \mathrm{X} & & \\
 & & & \mathrm{X} & & & \mathrm{X} & \\
 & & \mathrm{X} & & & & & \mathrm{X}
\end{array}\right)
\end{array}
$$

of order 8. Observe that A is irreducible, has a zero-free diagonal, and the graph associated with its off-diagonal part has the five cycles

$$(1 \to 5 \to 6 \to 2 \to 8 \to 3 \to 1), \quad (1 \to 5 \to 7 \to 4 \to 6 \to 2 \to 8 \to 3 \to 1),$$
$$(4 \to 5 \to 6 \to 4), \quad (4 \to 6 \to 4), \quad \text{and} \quad (5 \to 7 \to 4 \to 5)$$

of lengths six, eight, three, two, three, respectively. As can be observed, a cutset must include at least two nodes one of which is either 4 or 6 due to the need to cut the cycle of length two. If we choose to put 4 in the cutset, all cycles but the one of length six are cut and we can choose any of $\{1, 2, 3, 5, 6, 8\}$ to be the second node in the cutset. On the other hand, if we choose to put 6 in the cutset, all cycles but the last one of length three are cut, implying our second node of choice can be any of $\{4, 5, 7\}$. Hence, there are 8 different minimum cutsets of cardinality two in this problem.

Now, assume that the cutset is $\mathcal{C} = \{4, 6\}$ and let the permuted matrix A in (1.1) be

$$QAQ^T = \begin{array}{c} \\ 2 \\ 8 \\ 3 \\ 1 \\ 5 \\ 7 \\ 4 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccccc} 2 & 8 & 3 & 1 & 5 & 7 & 4 & 6 \end{array} \\ \left(\begin{array}{cccccc|cc} \mathrm{X} & \mathrm{X} & & & & & & \\ & \mathrm{X} & \mathrm{X} & & & & & \\ & & \mathrm{X} & \mathrm{X} & & & & \\ & & & \mathrm{X} & \mathrm{X} & & & \\ & & & & \mathrm{X} & \mathrm{X} & & \mathrm{X} \\ & & & & & \mathrm{X} & \mathrm{X} & \\ \hline & & & & \mathrm{X} & & \mathrm{X} & \mathrm{X} \\ \mathrm{X} & & & & & & \mathrm{X} & \mathrm{X} \end{array}\right) \end{array}$$

for which $\mathcal{T} = \{1, 2, 3, 5, 7, 8\}$, $n_T = 6$, and $n_C = 2$. Using column partitioning, the permutation matrix is written as $Q = (e_2\ e_8\ e_3\ e_1\ e_5\ e_7\ e_4\ e_6)^T$, where e_j denotes the jth principal axis vector (i.e., jth column of the identity matrix). In practice, permutation matrices are not stored explicitly, but represented using permutation vectors. Hence, Q may be equivalently represented as $q = (2\ 8\ 3\ 1\ 5\ 7\ 4\ 6)^T$. We remark that although the shape of triangularity defines a permutation on the nodes in $\mathcal{T}$, the nodes in $\mathcal{C}$ can be permuted arbitrarily.

Festa, Pardalos & Resende (1999) provide a comprehensive survey on feedback set problems. There are various versions of these problems that arise in areas such as combinatorial circuit design, constraint satisfaction, and operating systems. As shown by Karp (1972), the problem of computing the *minimum cutset* of a general graph is non-deterministic polynomial time complete (NP-complete). Nevertheless, there are certain classes of graphs for which the problem is solvable in polynomial time. One such class is *reducible (flow) graphs*, which is characterized by Hecht & Ullman (1974). These graphs arise, for instance, in the analysis of program flows with the purpose of code optimization

and detecting/breaking deadlocks. They are defined by the existence of a node, called the *root* (or *initial node*), from which every other node in $\mathcal{V}$ is reachable by following a sequence of edges, and the uniqueness of the *directed acyclic graph* (*dag*) generated by different depth first search (DFS) orders of $G(\mathcal{V}, \mathcal{E})$ starting from the root. DFS and related graph algorithms are discussed in detail by Tarjan (1972).

Through a sequence of refinements, Shamir (1979) gives an algorithm that computes a minimum cutset for a reducible graph in time and space linear in the sum of its number of nodes and number of edges. Recall that DFS partitions the edges of a graph into *dag edges* and *backward edges* (Shamir, 1979, p. 647). Assuming that destination and source nodes of a backward edge in a reducible graph are named respectively as head and tail (Shamir, 1979, p. 650), backward edges are defined to be those for which the head is already in the stack when the tail is pushed into the stack during DFS. Shamir's algorithm uses DFS starting from the root to number the nodes of a reducible graph in preorder (when they are pushed into the stack), to label them in postorder (when they are popped from the stack), to consider successive heads in postorder, and to add new nodes to the cutset (changing their labels to zero) when their postorder label and preorder number become the same (Shamir, 1979, p. 653). Although this algorithm has a linear time complexity, it may abort on a non-reducible graph. Nevertheless, when it does not abort on a non-reducible graph, the cutset produced by the algorithm is also minimum.

At this point we should stress that the reducibility of a graph should not be confused with the reducibility of a matrix. In order to clarify the difference between these two concepts, we refer to a reducible matrix as one whose corresponding graph is not strongly connected, and indicate that there are reducible graphs which are strongly connected (e.g., Figure 3 of (Shamir, 1979, p. 648)) and non-reducible graphs which are not strongly connected (e.g., Figure 2(a) of (Shamir, 1979, p. 648)). Since we consider strongly connected graphs, which may be non-reducible, the situation regarding the possibility of aborting in Shamir's algorithm needs to be rectified. It turns out that an algorithm due to Rosen (1982) takes care of this problem.

Non-reducible graphs for which Shamir's algorithm does not abort are named *quasi-reducible* in (Rosen, 1982, p. 206). Through a modification to Shamir's algorithm, Rosen gives an algorithm called Cutfind (Rosen, 1982, p. 209), which runs in linear time and space, also computing cutsets of graphs that are not (quasi-)reducible. However, the cutsets computed by the proposed algorithm may not be minimum for graphs that are not (quasi-)reducible. The modification introduced by Rosen is simple: the cutset is augmented with those nodes for which Shamir's algorithm would

abort (i.e., nodes having postorder labels larger than their preorder numbers). Furthermore, on an example (e.g., Figure 3.2 of (Rosen, 1982, p. 211)), which is not strongly connected, Rosen shows that the quasi-reducibility of a graph depends on the DFS order of visiting nodes starting from the root. Since Rosen's algorithm works on general graphs for which there is a root and each node in a strongly connected graph can be a root, it can be used to compute a cutset of the graph $G(\mathcal{V}, \mathcal{E})$ corresponding to the off-diagonal part of the irreducible matrix A with a zero-free diagonal.

As pointed out recently by Fages & Lal (2006, p. 2853), who use constraint programming to solve cutset problems in linear time, the greedy randomized adaptive search procedure (GRASP) due to Pardalos, Qian & Resende (1999) is currently considered to be the most effective algorithm for computing cutsets in graphs with a large number of nodes. The gfvs routine in GRASP provides an iterative algorithm, which returns the cutset with the smallest cardinality among all iterations as the solution. Each iteration consists of two phases. In the first phase, gfvs constructs a feasible solution, randomly selecting a node from a restricted candidate list in which nodes are ranked according to some adaptive greedy criterion. The available greedy criteria are based on *degrees* (i.e., numbers of outgoing or incoming edges) of nodes. In the second phase, the neighborhood of the solution constructed in the first phase is searched for a locally optimal solution. Various solution preserving graph reduction techniques discussed by Levy & Low (1988) are used to expedite the construction and local search phases. Similar to Rosen's algorithm, the gfvs routine of GRASP does not provide any guarantee on the quality of the computed cutset for general graphs. The implementation of GRASP for cutset problems is discussed by Festa, Pardalos & Resende (2001). For comparison purposes we also experiment with its gfvs routine, and remark that greedy algorithms for computing cutsets of graphs are also reported by Fourneau et al. (1994).

Continuing the example introduced in this section, the cutset returned by Rosen's algorithm is $\mathcal{C} = \{4, 5\}$, and $q = (7\ 6\ 2\ 8\ 3\ 1\ 4\ 5)^T$ is the corresponding permutation vector on which the permutation matrix Q is based. We remark that a minimum cutset is obtained although $G(\mathcal{V}, \mathcal{E})$ associated with the off-diagonal part of A is not (quasi)-reducible for the particular DFS order with root node 4. On the other hand, the cutset returned by the gfvs routine of GRASP in one iteration (which is also a minimum) and the corresponding permutation vector are as in the example.

In the next section, we introduce the (2×2) block partitioning algorithm which uses cutsets and discuss implementation issues.

3 Proposed algorithm

When A is reducible with a zero-free diagonal, it may be symmetrically permuted to block triangular form with irreducible diagonal blocks having zero-free diagonals. Without loss of generality and to simplify the discussion, let the given $(n_A \times n_A)$ reducible matrix with a zero-free diagonal be in the form

$$A = \begin{array}{c} \\ n_1 \\ n_2 \\ \vdots \\ n_K \end{array} \begin{array}{c} \begin{array}{cccc} n_1 & n_2 & \cdots & n_K \end{array} \\ \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1K} \\ & A_{22} & \cdots & A_{2K} \\ & & \ddots & \vdots \\ & & & A_{KK} \end{pmatrix} \end{array}, \tag{3.1}$$

where A_{kk} is an $(n_k \times n_k)$ irreducible submatrix with a zero-free diagonal for $k = 1, 2, \ldots, K$ and $n_A = \sum_{k=1}^{K} n_k$. An irreducible A is a special case with $K = 1$. Similar to (1.1), permutation matrices Q_{kk} can be computed such that

$$Q_{kk} A_{kk} Q_{kk}^T = \begin{array}{c} \\ n_{T_{kk}} \\ n_{C_{kk}} \end{array} \begin{array}{c} \begin{array}{cc} n_{T_{kk}} & n_{C_{kk}} \end{array} \\ \begin{pmatrix} T_{kk} & X_{kk} \\ Y_{kk} & C_{kk} \end{pmatrix} \end{array}, \tag{3.2}$$

where $n_k = n_{T_{kk}} + n_{C_{kk}}$ and T_{kk} is an $(n_{T_{kk}} \times n_{T_{kk}})$ triangular submatrix for $k = 1, 2, \ldots, K$. In order to be consistent with the block upper-triangular form in (3.1), let us assume that T_{kk} is upper-triangular. Observe that for the global index set $\{1, 2, \ldots, n_A\}$, Q_{kk} is a permutation matrix defined over the indices in $\{1 + \sum_{j=1}^{k-1} n_j, 2 + \sum_{j=1}^{k-1} n_j, \ldots, \sum_{j=1}^{k} n_j\}$. Furthermore, let the permutation vector q_k corresponding to the permutation matrix Q_{kk} be partitioned into two subvectors as $q_k^T = (q_{T_{kk}}^T \; q_{C_{kk}}^T)$. Then a permutation matrix Q can be obtained such that

$$QAQ^T = \begin{array}{c} \\ n_{T_{11}} \\ n_{T_{22}} \\ \vdots \\ n_{T_{KK}} \\ n_{C_{11}} \\ n_{C_{22}} \\ \vdots \\ n_{C_{KK}} \end{array} \begin{array}{c} \begin{array}{cccc|cccc} n_{T_{11}} & n_{T_{22}} & \cdots & n_{T_{KK}} & n_{C_{11}} & n_{C_{22}} & \cdots & n_{C_{KK}} \end{array} \\ \left(\begin{array}{cccc|cccc} T_{11} & T_{12} & \cdots & T_{1K} & X_{11} & X_{12} & \cdots & X_{1K} \\ & T_{22} & \cdots & T_{2K} & & X_{22} & \cdots & X_{2K} \\ & & \ddots & \vdots & & & \ddots & \vdots \\ & & & T_{KK} & & & & X_{KK} \\ \hline Y_{11} & Y_{12} & \cdots & Y_{1K} & C_{11} & C_{12} & \cdots & C_{1K} \\ & Y_{22} & \cdots & Y_{2K} & & C_{22} & \cdots & C_{2K} \\ & & \ddots & \vdots & & & \ddots & \vdots \\ & & & Y_{KK} & & & & C_{KK} \end{array}\right) \end{array}. \tag{3.3}$$

Letting $n_T = \sum_{k=1}^{K} n_{T_{kk}}$ and $n_C = \sum_{k=1}^{K} n_{C_{kk}}$ yields a (2×2) block partition as in (1.1). The first diagonal block in (3.3) is upper-triangular and the permutation vector on which Q is based is

$$q^T = (q_{T_{11}}^T \; q_{T_{22}}^T \; \cdots \; q_{T_{KK}}^T \; q_{C_{11}}^T \; q_{C_{22}}^T \; \cdots \; q_{C_{KK}}^T).$$

Algorithm 1 shows how one can compute a permutation for a (2×2) block partition of a matrix A with a zero-free diagonal and an order larger than one so that the first diagonal block is triangular as in (3.3). At the outset, A is assumed to be in block upper-triangular form with irreducible diagonal blocks having zero-free diagonals as indicated in the pre-condition (see Require). If it is not, but is structurally nonsingular, it can be row permuted to have a zero-free diagonal using the algorithm due to Duff (1981a,b). To that effect, the routine mc21a from the Harwell Subroutine Library (HSL, 2007) can be used. Once A has a zero-free diagonal, it can be symmetrically permuted to block upper-triangular form as indicated by Tarjan (1972) with the help of the routine mc13d from the HSL. We remark that mc13d as given by Duff & Reid (1978) is normally used to compute a permutation which block lower-triangularizes a matrix with a zero-free diagonal so that it has irreducible diagonal blocks. Hence, for a block upper-triangular matrix satisfying the same requirements, the returned permutation from mc13d should be reversed.

Algorithm 1 Computes a permutation for a (2×2) block partition of A as in (3.3).

Require: A is in form (3.1) with a zero-free diagonal and order $n_A > 1$.

Ensure: q is permutation vector corresponding to permutation matrix Q in (3.3), $n_T > 0$, $n_C > 0$, and $n_A = n_T + n_C$.

1: Compute cutset $\mathcal{C}_k$ of the graph $G(\mathcal{V}_k, \mathcal{E}_k)$ associated with the off-diagonal part of A_{kk} for $k = 1, 2, \ldots, K$.

2: Compute permutation vector $q_k^T = (q_{T_{kk}}^T \; q_{C_{kk}}^T)$ that triangularizes submatrix of A_{kk} associated with nodes in $\mathcal{T}_k = \mathcal{V}_k - \mathcal{C}_k$ for $k = 1, 2, \ldots, K$.

3: Form permutation vector $q^T = (q_{T_{11}}^T \; q_{T_{22}}^T \; \cdots \; q_{T_{KK}}^T \; q_{C_{11}}^T \; q_{C_{22}}^T \; \cdots \; q_{C_{KK}}^T)$, which has the $n_T = \sum_{k=1}^K n_{T_{kk}}$ nodes in $\mathcal{T} = \cup_{k=1}^K \mathcal{T}_k$ at its beginning and the $n_C = \sum_{k=1}^K n_{C_{kk}}$ nodes in $\mathcal{C} = \cup_{k=1}^K \mathcal{C}_k$ at its end.

For step 1, we consider an implementation of the Cutfind algorithm due to Rosen discussed in section 2, which takes the nonzero pattern of matrix A_{kk} as input using two integer arrays, one storing its column indices and is of length nz_k, where nz_k is the number of nonzeros in A_{kk}, while the other storing the beginning of row indices in the first array and is of length $(n_k + 1)$. In our implementation, the diagonal elements appear as the first elements in their corresponding rows, thereby enabling them to be skipped and the graph $G(\mathcal{V}_k, \mathcal{E}_k)$ associated with the off-diagonal part of A_{kk} to be considered. Other than these two arrays, the algorithm requires five integer work arrays of length n_k and a character work array of length n_k. Of the integer work arrays, two are used for implementing a stack of edges, one is used for labeling nodes, one is used for keeping track of the

next unprocessed edge to consider for each node, and one is used for recording preorders of nodes. The character work array is used for marking the nodes as they are processed.

The Cutfind algorithm requires the determination of a node called the root at the outset. Since $G(\mathcal{V}_k, \mathcal{E}_k)$ is strongly connected by assumption, any node in $\mathcal{V}_k$ qualifies as root. We choose the node with the largest *outdegree* (i.e., number of outgoing edges). When there are ties, the node with the smallest local index in $\mathcal{V}_k$ is chosen. We have also considered the maximum product of outdegrees and indegrees among nodes when choosing the root (as suggested by Pardalos, Qian & Resende (1999)). This has only made an insignificant difference, if at all, since it only affects the choice of the root and not the order of DFS. The operations carried out by Cutfind to obtain $\mathcal{T}_k$ are integer comparisons and amount to a time complexity of $O(nz_k)$.

Once $\mathcal{T}_k$ is determined, the triangularization in step 2 can be achieved again by using the algorithm due to Duff & Reid (1978). To that effect, again the routine mc13d from HSL can be employed. We remark that this routine, in this case returns the permutation for a lower-triangular matrix (in which there are $n_{T_{kk}}$ diagonal blocks and each block is of order one). Hence, if an upper-triangular matrix is desired, the returned permutation should be reversed. In order to call mc13d with the submatrix of A_{kk} associated with $\mathcal{T}_k$, the matrix A_{kk} is transformed to a block form in which the diagonal blocks of the partition are stored separately as rowwise sparse matrices and its off-diagonal part is also stored as a rowwise sparse matrix. A side benefit of this approach is that the nonzero patterns of the diagonal blocks become readily available and the second diagonal block C_{kk} in (3.2) can be analyzed for fill-in before its LU factorization (Duff et al., 1986, p. 46). The time complexity of step 2 is not larger than that of step 1 since mc13d runs in $O(n_{T_{kk}}) + O(nz_{T_{kk}} - n_{T_{kk}})$ (Duff & Reid, 1978, p. 138), where $nz_{T_{kk}}$ is the number of nonzeros in T_{kk}. The permutation defined by Q in (3.3) is returned after step 3 using an integer work array and an integer indicating the value of n_C (see the post-condition in Ensure).

In the next section, we provide the results of experiments on 19 test matrices arising in 9 groups from the literature.

4 Experimental results

We implemented the Cutfind algorithm due to Rosen, called rosen, in C as part of Algorithm 1 as discussed in section 3 and performed experiments on a 3.4 GHz Pentium IV processor and a 2 Gigabytes main memory under Cygwin using the O3 level optimization in compiling the code. The

code used in these experiments may be obtained from Dayar (2007). We considered unsymmetric matrices from the University of Florida Sparse Matrix Collection (2007). All matrices are preprocessed by removing elements corresponding to zero values, transposing the resulting matrices so that they are stored in rowwise sparse format, permuting to a zero-free diagonal using mc21a and then running mc13d to obtain a block triangular partition as in (3.1). We also performed experiments with the irreducible matrices using the gfvs routine of GRASP on the same platform. Noticing that the gfvs routine cannot be applied to larger matrices for computing cutsets efficiently, we refrained from experimenting with it further.

We also considered solving the linear system of equations $Ax = b$, where the chosen matrices are used as the coefficient matrix A. As the right-hand side vector, we took $b = Ae$, where e is the vector of ones, so that the solution vector is given by $x = e$ and the relative error in the computed solution can be determined explicitly. As solvers, we used the sequence of routines ma48a (for analyze), ma48b (for factorize), and ma48c (for solve) as discussed by Duff & Reid (1996) from the HSL as a direct solver with default settings of their parameters and a (forward) block Gauss-Seidel (BGS) solver we programmed on the partitioning in (3.3) for which the second diagonal block is analyzed and factorized at the outset using the routines ma48a and ma48b with default settings of their parameters, and the linear system of equations arising from it solved during the second block iteration using ma48cd again with default settings of its parameters. Clearly, the BGS solver is not expected to converge for arbitrary coefficient matrices, and therefore cannot be a replacement for the direct solver. Therefore, in presenting the results we selected a representative set of matrices for which both solvers can be run on the same platform and produce acceptable results.

4.1 Irreducible matrices

As benchmarks we considered 14 irreducible, unsymmetric test matrices from the UF Sparse Matrix Collection (2007). The orders of the matrices are between 1,000 and 85,000, and they arise in semiconductor simulation (ADD20, ADD32), 2D incompressible flow and 3D semiconductor device (SWANG1, WANG4), convective thermal flow (FLOWMETER5, CHIPCOOL0), finite element (POISSON3DA), DNA electrophoresis (CAGE8, CAGE9, CAGE10), and chemical processing (EPB0, EPB1, EPB2, EPB3). These 14 matrices come respectively from the 6 groups Hamm, Wang, Oberwolfach, FEMLAB, VanHeukelum, and Averous. The matrices from the vanHeukelum and Averous groups are chosen to demonstrate the scalability of Rosen's algorithm on different sized versions of a problem. All matrices except those in the Averous group have symmetric nonzero

Table 1: Characteristics of A for the irreducible test matrices and solution of $Ax = b$ with direct solver.

			outdegree			indegree					
Matrix	n_A	nz_A	min	max	avg	min	max	avg	Space_A	RelErr	Time_s
ADD20	2,395	13,151	1	83	4.5	1	83	4.5	34,122	1e−13	0.0
SWANG1	3,169	20,841	2	9	5.6	2	9	5.6	166,257	2e−15	0.1
ADD32	4,960	19,848	1	14	3.0	1	14	3.0	48,712	3e−15	0.0
FLOWMETER5	9,669	67,391	3	10	6.0	3	10	6.0	794,049	5e−13	0.9
POISSON3DA	13,514	352,762	5	109	25.1	5	109	25.1	12,215,412	3e−13	59.8
CHIPCOOL0	20,082	281,150	4	23	13.0	4	23	13.0	15,829,289	1e−12	97.6
WANG4	26,068	177,196	3	6	5.8	3	6	5.8	18,423,325	2e−9	108.7
CAGE8	1,015	11,003	4	17	9.8	4	17	9.8	207,923	5e−15	0.1
CAGE9	3,534	41,594	2	22	10.8	2	22	10.8	1,476,575	9e−15	2.5
CAGE10	11,397	150,645	4	24	12.2	4	24	12.2	21,304,670	2e−14	120.1
EPB0	1,794	7,764	1	10	3.3	1	5	3.3	37,128	3e−13	0.0
EPB1	14,734	95,053	2	6	5.5	2	8	5.5	2,200,972	2e−12	4.6
EPB2	25,228	175,027	2	86	5.9	2	86	5.9	5,567,867	6e−12	23.2
EPB3	84,617	463,625	2	5	4.5	2	6	4.5	15,236,063	1e−11	76.6

patterns.

The characteristics of A corresponding to the 14 test matrices appear in Table 1. Column Matrix has the name of the test matrix. Columns n_A and nz_A have the order and number of nonzeros in A. Columns four through six and columns seven through nine have respectively the minimum, maximum, average outdegrees and indegrees of the graph $G(\mathcal{V}, \mathcal{E})$ associated with the off-diagonal part of A. Columns Space_A, RelErr, and Time_s are related to the direct solver and provide the space required by the LU factorization of A, the relative error in the computed solution, and the time taken in seconds by the solver. Recall that A has a zero-free diagonal, and therefore, $nz_A = |\mathcal{E}| + n_A$. Average number of nonzeros per row/column of A range from 4.0 for ADD32 to 26.1 for POISSON3DA, although the maximum number of nonzeros per row/column can be as high as 110, which is also for POISSON3DA. It is interesting to note that the space required by the factorization is relatively large in the vanHeukelum matrices, yielding also a relatively large amount of time to obtain the solution. This implies that use of the direct solver is difficult to justify for these three matrices, even if the space required by the factorization is not considered.

Table 2 presents the results of using rosen (r) and one iteration of gfvs (g) in Algorithm 1 to obtain a (2×2) block partition on the 14 irreducible test matrices and of solving $Ax = b$ with the block iterative solver. Column M indicates the algorithm used for computing the cutset. Columns n_T and nz_T have the order and number of nonzeros in T. Column Time_p indicates the time spent in seconds to obtain a (2×2) block partition as in (1.1) using cutset computation. Columns n_C and nz_C have the order and number of nonzeros in C. Note that the former is the cardinality of the cutset. Columns Space_C, Iter, RelErr, and Time_s are related to the block iterative solver and provide the

Table 2: Characteristics of the blocks in the partition of A obtained using Algorithm 1 for irreducible matrices and solution of $Ax = b$ with block iterative solver.

Matrix	M	n_T	nz_T	Time$_p$	n_C	nz_C	Space$_C$	Iter	RelErr	Time$_s$
ADD20	r	1,182	1,182	0.0	1,213	6,319	17,835	26,665	1e−8	4.3
	g	1,190	1,190	1.7	1,205	6,161	17,415	26,666	1e−8	4.3
SWANG1	r	845	845	0.0	2,324	11,016	59,848	26	1e−14	0.0
	g	942	942	5.3	2,227	9,907	55,275	27	2e−14	0.0
ADD32	r	2,305	2,305	0.0	2,655	8,335	18,585	628	7e−11	0.2
	g	2,305	2,305	4.1	2,655	7,575	19,022	629	7e−11	0.3
FLOWMETER5	r	2,518	2,518	0.0	7,151	35,097	259,699	25,860	1e−8	58.4
	g	2,574	2,574	95.1	7,095	34,493	267,253	25,613	1e−8	58.9
POISSON3DA	r	1,659	1,659	0.0	11,855	289,303	9,654,207	1,201	3e−10	89.2
	g	1,982	1,982	3,086.0	11,532	281,954	8,590,916	1,296	3e−10	88.8
CHIPCOOL0	r	3,560	3,560	0.0	16,522	193,270	9,380,599	9,525	5e−9	501.2
	g	3,601	3,601	2,339.0	16,481	195,199	10,724,369	9,214	5e−9	578.1
WANG4	r	12,878	12,878	0.0	13,190	14,990	29,980	6,181	2e−9	10.7
	g	9,390	9,390	845.6	16,678	59,448	525,336	5,134	2e−9	23.7
CAGE8	r	235	235	0.0	780	6,344	92,491	26	6e−15	0.0
	g	236	236	0.8	779	6,439	101,912	31	1e−14	0.1
CAGE9	r	820	820	0.0	2,714	24,634	836,452	29	1e−14	1.1
	g	824	824	16.6	2,710	24,912	854,484	28	2e−14	1.2
CAGE10	r	2,382	2,382	0.0	9,015	93,035	12,915,048	22	1e−14	53.6
	g	2,404	2,404	475.1	8,993	93,449	12,177,427	21	2e−14	54.0
EPB0	r	1,495	2,985	0.0	299	299	598	22,639	1e−8	1.9
	g	1,495	2,985	0.8	299	299	598	22,639	1e−8	1.9
EPB1	r	5,723	9,705	0.0	9,011	38,150	263,140	12,997	6e−9	30.0
	g	7,314	14,558	114.5	7,420	36,748	344,605	13,941	7e−9	40.3
EPB2	r	10,460	26,880	0.0	14,768	64,648	634,386	1,436	4e−10	8.4
	g	12,098	23,466	364.9	13,130	63,458	673,439	1,186	3e−10	7.4
EPB3	r	33,368	57,395	0.1	51,249	171,549	357,636	46,955	2e−8	506.9
	g	35,190	63,666	5,065.0	49,427	169,049	540,423	32,227	2e−8	319.0

space required by the factorization of C using ma48ad and ma48bd, the number of iterations taken to convergence, the relative error in the computed solution, and the time taken in seconds by the solver. We remark that Time$_s$ includes time spent for factorizing C. For gfvs, we use the parameter settings `alpha` $= -1$, `look4` $= 0$, `maxitr` $= 1$, `prttyp` $= 1$, and `seed` $= 270001$ (Festa, Pardalos & Resende, 2001, p. 460), so that one iteration is performed. Clearly, the cutsets computed by gfvs are likely to improve with a larger number of iterations; but, this can happen only at the expense of larger processing time.

The cutsets computed by rosen indicate that the graphs $G(\mathcal{V}, \mathcal{E})$ corresponding to A for the 14 test matrices are not (quasi-)reducible. This implies that Shamir's algorithm could not be used in any of the test matrices for the choice of the root and the particular DFS order used. The ratio n_C/n_A ranges from 0.17 for EPB0 to 0.88 for POISSON3DA with rosen, and is smaller for gfvs except WANG4 for which rosen gives a smaller cutset, and ADD32 and EPB0 for which the cutsets computed by rosen and gfvs are of the same cardinality. The average value of the ratio n_C/n_A over the 14 matrices for rosen is 0.65. The time to obtain the cutsets with rosen does not exceed tenth

of a second for any of the matrices and is always smaller than that with gfvs even though only one iteration in gfvs is used. For matrices that are of order 9,000 or larger, gfvs takes in the order of minutes to obtain a cutset. The increase in time of gfvs can be clearly observed for the matrices in the vanHeukelum and Averous groups. In between rosen and gfvs, the block iterative solver with rosen is better than that with gfvs on all matrices when we include the time to obtain the (2×2) block partition based on cutsets.

It is interesting to note that cutsets obtained with rosen and gfvs yield diagonal T blocks in all the matrices except those in the Averous group and diagonal C blocks in EPB0. If we exclude the diagonal T matrices, the ratio nz_T/nz_A ranges from 0.10 for EPB1 to 0.38 for EPB0 with rosen and from 0.13 for EPB2 to 0.38 for EPB0 with gfvs. However, we must remark that the matrices in the Averous group are of different degrees of difficulty from the point of view of the block iterative solver as can be seen in the values of column Iter. The ratio of the space requirement for the first diagonal block, the two off-diagonal blocks, and the factorization of the second diagonal block in (1.1) with respect to the space requirement for the factorization of A is given by $(nz_A - nz_C + \text{Space}_C)/\text{Space}_A$. This ratio ranges from 0.01 for WANG4 to 0.80 for POISSON3DA with rosen and from 0.03 for WANG4 to 0.72 for ADD20 with gfvs.

The block iterative solver with rosen is able to compute the solution in at least 8 decimal digits of accuracy for all irreducible matrices and produces a solution that is about as accurate as that of the direct solver for WANG4 and the vanHeukelum matrices. It is the winner spacewise for all matrices and timewise for SWANG1, the vanHeukelum matrices, and EPB2. Interestingly, these five matrices are not the most favorable ones in terms of the ratios n_C/n_A, nz_T/nz_A, and $(nz_A - nz_C + \text{Space}_C)/\text{Space}_A$; however, the first four of these are matrices for which the block iterative solver takes no more than 31 iterations to converge

4.2 Reducible matrices

As benchmarks we considered 5 reducible, unsymmetric test matrices from the UF Sparse Matrix Collection (2007). The orders of the matrices are between 11,000 and 117,000, and they arise in active control of a supersonic engine inlet (INLET), unstructured 2D mesh (AIRFOIL_2D), circuit simulation (ASIC_100K), and circuit transient simulation and DC operating point (TRANS4, DC2). These 5 matrices come respectively from the 4 groups Oberwolfach, Engwirda, Sandia, and IBM_EDA. None of these matrices have a symmetric nonzero pattern.

The characteristics of A corresponding to the 5 test matrices appear in Table 3. The columns

Table 3: Characteristics of A for the reducible test matrices and solution of $Ax = b$ with direct solver.

				outdegree			indegree					
Matrix	n_A	nz_A	K	min	max	avg	min	max	avg	Space_A	RelErr	Time_s
INLET	11,730	328,323	331	0	40	27.0	8	38	27.0	5,323,627	2e−10	21.6
AIRFOIL_2D	14,214	259,688	638	0	22	17.3	3	22	17.3	2,975,975	1e−13	6.1
ASIC_100K	99,340	940,621	399	0	92,257	8.5	0	92,257	8.5	6,034,613	7e−11	12.6
TRANS4	116,385	749,800	19	0	114,189	5.4	0	114,173	5.4	1,964,601	4e−10	32.1
DC2	116,385	766,396	19	0	114,189	5.6	0	114,173	5.6	2,027,123	2e−8	34.8

Table 4: Characteristics of the blocks in the partition of A obtained using Algorithm 1 with rosen for reducible matrices and solution of $Ax = b$ with block iterative solver.

Matrix	n_T	nz_T	Time_p	n_C	nz_C	Space_C	Iter	RelErr	Time_s
INLET	1,894	4,159	0.0	9,836	249,119	2,444,964	623	1e−10	14.8
AIRFOIL_2D	2,308	3,069	0.0	11,906	194,932	2,127,527	5,471	3e−9	79.2
ASIC_100K	36,802	158,921	0.1	62,538	275,180	1,048,632	265	9e−11	4.8
TRANS4	86,194	111,135	0.1	30,641	203,815	501,602	1,043	3e−10	12.1
DC2	86,194	127,731	0.1	30,641	203,815	532,939	9,994	4e−9	109.5

have the same meaning as in Table 1. The extra column K provides the number of diagonal blocks in (3.1). Average number of nonzeros per row/column of A range from 6.4 for TRANS4 to 28.0 for INLET, although the maximum number of nonzeros per row and column can be as high as 114,190 and 114,174, respectively, which are for TRANS4 and DC2. The space required by factorization is relatively large for INLET, yielding also a relatively large amount of time to obtain the solution. Hence, the use of the direct solver is difficult to justify for this matrix, even if the space required by the factorization is not considered.

Table 4 presents the results of using Algorithm 1 with rosen (r) to obtain a (2×2) block partition on the 5 reducible test matrices and of solving $Ax = b$ with the block iterative solver. The columns have the same meaning as in Table 2. The cutsets computed indicate that the graphs $G(\mathcal{V}_k, \mathcal{E}_k)$ corresponding to A_{kk} in 3.1 for $k = 1, 2, \ldots, K$ in the 5 test matrices are not (quasi-)reducible. This implies that Shamir's algorithm could not be used in any of the test matrices for the choices of roots in each $G(\mathcal{V}_k, \mathcal{E}_k)$ and the particular DFS order used. The ratio n_C/n_A ranges from 0.26 for TRANS4 and DC2 to 0.84 for INLET and AIRFOIL_2D. The average value of the ratio n_C/n_A over the 5 matrices is 0.57. The time to obtain the cutsets does not exceed tenth of a second for any of the matrices.

For the 5 reducible test matrices, none of the cutsets yield diagonal T or C blocks. The ratio nz_T/nz_A ranges from 0.01 for INLET and AIRFOIL_2D to 0.17 for ASIC_100K and DC2. We must remark that the AIRFOIL_2D and DC2 matrices are particularly difficulty to solve by the block iterative solver as can be seen in the values of column Iter. The ratio $(nz_A - nz_C + \text{Space}_C)/\text{Space}_A$

ranges from 0.28 for ASIC_100K to 0.74 for AIRFOIL_2D.

The block iterative solver is able to compute the solution in at least 9 decimal digits of accuracy for all reducible matrices and produces a solution that is about as accurate as that of the direct solver for INLET, ASIC_100K, and TRANS4 matrices, and a solution that is more accurate for DC2. It is the winner spacewise for all matrices and timewise for INLET, ASIC_100K, and TRANS4.

5 Conclusion

In this paper, we have given an algorithm which computes a (2×2) block partition of an irreducible, sparse matrix with a zero-free diagonal so that one of the diagonal blocks is triangular. In doing this, we have computed a cutset of the graph associated with the off-diagonal part of the matrix using an algorithm due to Rosen. This algorithm does not seem to be as well known and used as Shamir's algorithm, which is for (quasi-)reducible graphs. The proposed algorithm runs in time linear in the number of nonzeros in the off-diagonal part of the matrix using information only about its nonzero pattern, and is shown to return a permutation for a block partition which can be expected to yield two diagonal blocks roughly of the same order. The orders and numbers of nonzeros of these blocks are observed to change depending on the nonzero pattern of the matrix. The triangular diagonal block in the (2×2) block partition obtained using a cutset is expected to become smaller as the matrix becomes denser since there will be normally a larger number of cycles to cut. The proposed algorithm is also extended to reducible, sparse matrices with a zero-free diagonal. Another algorithm, which is based on a greedy randomized adaptive search procedure, is also considered for computing cutsets. Although this alternative algorithm is iterative and has the possibility of reducing the cardinality of the cutset over a number of iterations (if not in one iteration), it is shown not to be efficient timewise, and therefore cannot be recommended for our purposes unless the matrix is relatively small. As we have shown, there are cases in which computing cutsets for sparse matrices and using a block iterative solver on the (2×2) block partitioning based on cutsets pays off in comparison to an industrial strength direct solver, especially in terms of the reduction in space. The results are representative of the relative cardinalities of cutsets that can be computed for sparse matrices, and therefore pave the way for further means of exploiting cutsets.

Acknowledgments. This work has been carried out through grant TÜBA-GEBİP from the Turkish Academy of Sciences. We thank Jean-Michel Fourneau for his comments on greedy heuristic

algorithms for computing cutsets.

References

Davis, T. (2007) University of Florida Sparse Matrix Collection. *NA Digest*, **92**, no. 42, October 16, 1994, *NA Digest*, **96**, no. 28, July 23, 1996, and *NA Digest*, *97*, no. 23, June 7, 1997. http://www.cise.ufl.edu/research/sparse/matrices/.

Dayar, T. (2007) *Software for obtaining triangular diagonal blocks in sparse matrices using cutsets.* http://www.cs.bilkent.edu.tr/˜tugrul/software.html.

Duff, I. S. (1981) On algorithms for obtaining a maximum transversal. *ACM Trans. Math. Software*, **7**, 315–330.

Duff, I. S. (1981) Algorithm 575: Permutations for a zero-free diagonal. *ACM Trans. Math. Software*, **7**, 387–390.

Duff, I. S., Erisman, A. M. & Reid, J. K. (1986) *Direct Methods for Sparse Linear Systems.* New York, NY: Oxford University Press.

Duff, I. S. & Reid, J. K. (1978) An implementation of Tarjan's algorithm for the block triangularization of a matrix. *ACM Trans. Math. Software*, **4**, 137–147.

Duff, I. S. & Reid, J. K. (1996) The design of MA48: A Code for the direct solution of sparse unsymmetric linear systems of equations. *ACM Trans. Math. Software*, **22**, 187–226.

Fages, F. & Lal, A. (2006) A constraint programming approach to cutset problems. *Comput. Oper. Res.*, **33**, 2852–2865.

Festa, P., Pardalos, P. M., & Resende, M. G. C. (1999) Feedback set problems. *Handbook of Combinatorial Optimization*, vol. A (D. Z. Du & P. M. Pardalos eds.). Boston, MA: Kluwer Academic Publishers, pp. 209–258.

Festa, P., Pardalos, P. M., & Resende, M. G. C. (2001) Algorithm 815: FORTRAN subroutines for computing approximate solutions of feedback set problems using GRASP. *ACM Trans. Math. Sotware*, **27**, 456–464.

Fourneau, J.-M., Kloul, L., Mokdad, L. & Quessette, F. (1994) A new tool to model parallel systems and protocols. *Proceedings of the European Simulation Symposium* (A. R. Kaylan,

A. Lehmann & T. I. Oren eds.). Istanbul, Turkey: Society for Computer Simulation International, pp. 220–224.

Harwell Subroutine Library. (2007) http://www.cse.clrc.ac.uk/nag/hsl/.

Hecht M. S. & Ullman, J. D. (1974) Characterizations of Reducible Flow Graphs. *J. ACM*, **21**, 367–375.

Karp, R. (1972) Reducibility among combinatorial problems. *Complexity of Computer Communications* (R. E. Miller & J. W. Thatcher eds.). New York, NY: Plenum Press, pp. 85–103.

Levy, H. & Low, D. W. (1988) A contraction algorithm for finding small cycle cutsets. *J. Algorithms*, **9**, 470–493.

Pardalos, P. M., Qian, T. B. & Resende, M. G. C. (1999) A greedy randomized adaptive search procedure for the feedback vertex set problem. *J. Comb. Optim.*, **2**, 399–412.

Rosen, B. K. (1982) Robust linear algorithms for cutsets. *J. Algorithms*, **3**, 205–217.

Saad, Y. (2003) *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA: SIAM Press.

Shamir, A. (1979) A linear time algorithm for finding minimum cutsets in reducible graphs. *SIAM J. Comput.*, **8**, 645–655.

Tarjan, R. E. (1972) Depth-first search and linear graph algorithms. *SIAM J. Comput.*, **1**, 146–160.