Discrete Mathematics 313 (2013) 1558–1563

Contents lists available at SciVerse ScienceDirect

Discrete Mathematics

journal homepage: www.elsevier.com/locate/disc

Note

A note on perfect partial elimination

Matthijs Bomhoff*, Walter Kern, Georg Still

University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, P.O. Box 217, 7500 AE Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 8 July 2011
Received in revised form 27 January 2013
Accepted 1 April 2013
Available online 18 April 2013

Keywords:
Gaussian elimination
Perfect elimination
Bipartite graph
Complexity

ABSTRACT

In Gaussian elimination it is often desirable to preserve existing zeros (sparsity). This is closely related to perfect elimination schemes on graphs. Such schemes can be found in polynomial time. Gaussian elimination uses a pivot for each column, so opportunities for preserving sparsity can be missed. In this paper we consider a more flexible process that selects a pivot for each nonzero to be eliminated and show that recognizing matrices that allow such perfect partial elimination schemes is NP-hard.

1. Introduction

Gaussian elimination is the classic algorithm for solving systems of linear equations. The core of the algorithm consists of using elementary row operations to reduce a matrix to triangular form. During each round of the elimination process, a nonzero element of the remaining matrix, the *pivot*, is picked. The row and column of the pivot element are called the *pivot row* and *pivot column* respectively. Using elementary row operations, all other nonzero elements of the pivot column are *cleared* by subtracting a multiple of the pivot row from their rows.

More formally: let M be a real-valued matrix and denote by $M_{i,j}$ the element of M in row i and column j. If $M_{k,l} \neq 0$ is used as a Gaussian pivot, the elements of the matrix M' we obtain after pivoting on (k, l) are given by (1):

$$M'_{i,j} = \begin{cases} M_{i,j} - \dfrac{M_{i,l}}{M_{k,l}} M_{k,j} & \text{if } i \neq k \text{ and } M_{i,l} \neq 0 \\ \dfrac{M_{i,j}}{M_{k,l}} & \text{if } i = k \\ M_{i,j} & \text{otherwise.} \end{cases} \tag{1}$$

All elements of M with $i \neq k$ and $j = l$ are turned into zeros in M'. Unfortunately, other elements of M that are zero may be turned into nonzeros in M' in columns other than the pivot column. This phenomenon is called *fill-in*.

When performing Gaussian elimination on sparse matrices it is often desirable to preserve sparsity. Avoiding fill-in completely during elimination, the so-called *perfect elimination*, has been treated extensively in the literature, both for the general case of square matrices [4,3,7] and for special cases such as symmetric matrices or pivots on the diagonal [6,5]. For each of these cases, determining whether perfect elimination is possible can be done in polynomial time.

It is characteristic for the Gaussian elimination algorithm that in each iteration a pivot element is picked and used to clear its entire column. If we want to avoid fill-in completely, this limits the set of matrices we can apply this method to. In order

* Corresponding author.
E-mail addresses: matthijs@bomhoff.nl, m.j.bomhoff@utwente.nl (M. Bomhoff), w.kern@utwente.nl (W. Kern), g.j.still@utwente.nl (G. Still).

0012-365X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.disc.2013.04.001

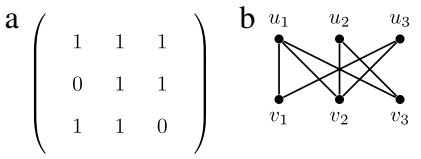

Fig. 1. A {0, 1}-matrix M and its bipartite graph $G[M]$ where (u_3, v_1) is bisimplicial.

to achieve perfect elimination for a broader set of matrices, we can use a more fine-grained elimination process in which we eliminate single nonzero values instead of entire columns at a time. Rose and Tarjan already mentioned such *partial elimination* as an alternative to ordinary Gaussian elimination for this reason [5]. In this paper we show that determining whether a matrix allows perfect partial elimination is NP-hard. The remainder of this paper is organized as follows: the next section introduces the {0, 1}-matrix and bipartite graph representations of the problem as well as the concept of perfect elimination. The third section formalizes the more fine-grained process we analyze. The section after that contains our main result on the NP-hardness of the related recognition problem. Finally, the fifth section contains some concluding remarks.

2. Perfect elimination

Our main interest in this paper is recognizing those matrices that admit perfect elimination. Under the assumption that subtracting a multiple of one row from another will not lead to 'accidental' cancellations besides zeroing the intended element, we can represent an instance of this problem by a {0, 1}-matrix M containing zeros in exactly the same places as the original matrix. In the {0, 1}-matrix representation, subtraction takes a different form: when subtracting two rows, only a single 1 is turned into a zero. This represents the assumption that no additional cancellations outside of the pivot column occur.

Formally, if we have a {0, 1}-matrix M representing the structure of a real-valued matrix and we pivot on the nonzero element (k, l), the elements of the matrix M' we obtain after pivoting are given by (2):

$$M'_{i,j} = \begin{cases} 0 & \text{if } j = l \text{ and } i \neq k \text{ and } M_{i,l} = 1 \\ \max\left(M_{i,j}, M_{k,j}\right) & \text{if } j \neq l \text{ and } i \neq k \text{ and } M_{i,l} = 1 \\ M_{i,j} & \text{otherwise.} \end{cases} \tag{2}$$

Note how in the {0, 1}-matrix representation the subtraction operation has been replaced by taking the maximum of two matrix elements. In particular the row operation turns a zero in a non-pivot row into a nonzero (causing fill-in) if the element of the pivot row in the corresponding column is a nonzero.

Interpreting the {0, 1}-matrix as a biadjacency matrix leads to a bipartite graph $G[M] = (U, V, E)$ with the rows and columns of M as its vertex classes U and V. An example matrix M and its associated bipartite graph $G[M]$ are shown in Fig. 1. In the remainder of this paper we borrow the following notation from Golumbic and Goss [4] for removing from a bipartite graph $G = (U, V, E)$ a set of vertices X as well as any edges incident to them: $G - X = \left(U', V', E'\right)$ with $U' = U \setminus X$, $V' = V \setminus X$ and $E' = \left\{uv \in E | u \in U', v \in V'\right\}$.

As shown by Golumbic and Goss [4], pivots suitable for perfect Gaussian elimination on M correspond to the so-called *bisimplicial* edges of $G[M]$.

Definition 2.1. An edge uv of a bipartite graph $G = (U, V, E)$ is called *bisimplicial* if the neighbors of its endpoints $\Gamma(u) \cup \Gamma(v)$ (where $\Gamma(u)$ denotes the neighbors of u) induce a complete bipartite subgraph of G.

Using this definition, the perfect elimination problem for bipartite graphs was first defined by Golumbic and Goss [4] as follows.

Definition 2.2. A bipartite graph $G = (U, V, E)$ is called *perfect elimination bipartite* if there exists a sequence of pairwise nonadjacent edges $[u_1v_1, \ldots, u_nv_n]$ such that u_iv_i is a bisimplicial edge of $G - \{u_1, v_1, \ldots, u_{i-1}, v_{i-1}\}$ for each i and $G - \{u_1, v_1, \ldots, u_n, v_n\}$ is empty. Such a sequence of edges is called a *(perfect elimination) scheme*.

The recognition of perfect elimination bipartite graphs corresponding to matrices that allow Gaussian elimination without fill-in is possible in polynomial time and several algorithms for this have been published [4,3,7].

3. Perfect partial elimination

In Gaussian elimination a single pivot element is used to clear its entire column. When trying to avoid fill-in, this process can be too restrictive and it may be beneficial to perform partial pivots, i.e., to select a new pivot element for every single nonzero element that we clear. A partial pivoting operation thus consists simply of subtracting a multiple of one row from another in order to clear a single nonzero element. For example, the matrix in Fig. 2(a) can be reduced to diagonal form by partial pivots without fill-in — although no perfect elimination scheme exists. To end up with the matrix in diagonal

a $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$ b $\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$

Fig. 2. Two matrices, both without a perfect elimination scheme, but (a) has a perfect partial elimination scheme whereas (b) does not.

$$\begin{pmatrix} \boxed{1} & \boxed{1} & 0 & 0 \\ \boxed{1} & \boxed{1} & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Fig. 3. The disposable elements (*indicated by squares*) of our example matrix.

$$\begin{pmatrix} 10 & 1 & 0 & 0 \\ 2 & 9 & 0 & 0 \\ 3 & 0 & 8 & 5 \\ 0 & 4 & 6 & 7 \end{pmatrix}$$

Fig. 4. An example perfect partial elimination scheme for the {0, 1}-matrix of Fig. 2(a).

form, we first clear the element (1, 2) using the second row. Then we clear the element (2, 1) using the first row which now only contains a single nonzero element. We then use the first and second rows to clear the elements (3, 1) and (4, 2). Subsequently, row four can now be used to clear element (3, 4) and after that row three can be used to clear element (4, 3).

Clearly, there are also matrices like the one in Fig. 2(b) for which even partial pivoting cannot avoid fill-in. It is thus natural to ask ourselves which matrices allow perfect elimination using such partial pivots.

As the partial elimination steps involve only row operations zeroing single nonzero elements in our matrix, we first look at elements that can be zeroed this way. Let us call nonzero elements of a matrix that can be cleared using a single partial pivot operation *disposable*. We first consider the elimination of edges in the corresponding bipartite graph. Recall that U denotes the vertex class corresponding to the rows of the matrix and V denotes the vertex class corresponding to the columns of the matrix. The edges corresponding to disposable elements of M can be easily characterized in the bipartite graph $G[M]$.

Definition 3.1. An edge uv of a bipartite graph $G = (U, V, E)$ is called *disposable* if there exists another edge $u'v \in E$ such that $\Gamma\left(u'\right) \subseteq \Gamma(u)$.

Before defining the analogous concept for a {0, 1}-matrix M, we first define the $\leq$ relation on matrix rows. We write $M_{i',*} \leq M_{i,*}$ to denote that for any column j of M, $M_{i',j} = 1$ implies $M_{i,j} = 1$. In other words, row i has ones in at least all the same columns as row i' and possibly more. Note how this captures the same notion expressed by $\Gamma\left(u'\right) \subseteq \Gamma(u)$ in the bipartite graph case. Defining disposable elements in the {0, 1}-matrix M is now rather straight-forward: a nonzero element $M_{i,j}$ of a row $M_{i,*}$ is called *disposable* if there is another row $M_{i',*} \leq M_{i,*}$ such that $M_{i',j}$ is also nonzero (element $M_{i',j}$ can be used as a partial pivot to clear element $M_{i,j}$). Fig. 3 illustrates the disposable elements of our example matrix. Clearly, disposable elements play an important role in the characterization of the matrices that allow perfect partial elimination schemes. We now come to the definition of the class of bipartite graphs associated with these matrices.

Definition 3.2. A bipartite graph $G = (U, V, E)$ with $|U| \geq |V| = n$ and $|E| = m$ (where $|X|$ denotes the cardinality of set X) is called *perfect partial elimination bipartite* if there exists an ordering $e_1, \ldots, e_m$ of the edges such that $\{e_{m-n+1}, \ldots, e_m\}$ together form a maximum matching of G covering V and for each $i \in \{1, \ldots, m-n\}$, e_i is a disposable edge of $G_i := \left(U, V, \left\{e_j \in E \mid j \geq i\right\}\right)$. We call this ordering a *perfect partial elimination scheme*.

In Fig. 4 we indicate an ordering of the nonzero matrix elements of the matrix in Fig. 2(a) that gives a perfect partial elimination scheme for the corresponding bipartite graph.

As our partial pivots only involve single edge operations, it is no longer required for the two vertex classes to be of equal size. Clearly, the notion of a perfect partial elimination scheme for G carries over readily to the corresponding {0, 1}-matrix, which also no longer has to be square.

Having defined the class of matrices and bipartite graphs that allow perfect partial elimination, the logical next question to ask is how hard it is to recognize members of this class. The corresponding decision problem can be stated as follows.

Perfect Partial Elimination
Instance: A {0, 1}-matrix M
Question: Does M have a perfect partial elimination scheme?

4. Perfect Partial Elimination is NP-complete

In this section we come to our main result. The proof is by reduction from Satisfiability [1]. We start by briefly defining this problem, using the terminology and notation from Garey and Johnson [2]. An instance S of Satisfiability consists of a set U of boolean variables and a set C of clauses defined over U. For every variable $u_i \in U$, u_i and $\bar{u}_i$ are called *literals* over U. A *truth assignment* (T, F) is a partition of the variables U. Under a given truth assignment, the literal u_i is *true* if and only if $u_i \in T$, otherwise it is *false*. Similarly, the literal $\bar{u}_i$ is *true* if and only if $u_i \in F$. The clauses in C are disjunctions of the literals over U. A given truth assignment is called *satisfying* for C if every clause in C contains at least one literal that is *true* under the truth assignment. The decision problem is now stated as follows.

Satisfiability
Instance: Set U of variables, collection C of clauses over U.
Question: Is there a satisfying truth assignment for C?

Using this we prove the NP-hardness of Perfect Partial Elimination.

Theorem 4.1. Perfect Partial Elimination *is NP-complete.*

Proof. As a given elimination sequence for perfect partial elimination can clearly be verified in polynomial time, we only show NP-hardness using a reduction from Satisfiability. For a given instance $S = (U, C)$ of Satisfiability, we construct a corresponding {0, 1}-matrix M_S such that M_S has a perfect partial elimination scheme if and only if S has a truth assignment for U that is satisfying for C. To simplify the reduction, we will assume w.l.o.g. that the instance S has at least two clauses, all clauses contain at least one literal, and C contains no tautologies, i.e., there is no i such that some clause contains both u_i and $\bar{u}_i$.

The matrix M_S has $4|U| + |C| + 1$ rows and $2|U| + |C| + 2$ columns. For the description of the construction procedure, it is convenient to label the rows and columns.

For every variable u_i in U we have two columns labeled u_i and $\bar{u}_i$. These columns are used to represent the variables, their truth assignments and their occurrences in the clauses. For every clause $c_i \in C$, we have a column labeled c_i. These columns are used to keep the individual clauses separated while linking them together in the overall satisfiability requirement. We also have two auxiliary columns labeled a and b, which are used to limit the possible subtractions between rows.

The rows of M_S are partitioned into five sets $\{V, W, D, K, R\}$. The sets V and W each contain one row per variable. Subtractions between rows of these sets are used to represent possible truth assignments for U. The set D contains 2 rows per variable and is used mainly to clear matrix elements that are no longer required for the elimination process themselves. The set K represents the clauses of S and links them to their literals. The final set R contains only a single row and encodes the requirement that all clauses must be satisfied by the truth assignment.

Having introduced the rows and columns that together form the constructed matrix M_S, we will now describe the values of the elements of M_S. The row set V contains a single row v_i for every variable u_i. Each such row contains a one in both the columns corresponding to u_i and $\bar{u}_i$ and zeros everywhere else. The rows w_i in W are identical to the rows v_i, except for an additional 1-entry in column a. The set D contains two rows for each variable u_i: one for each of the two corresponding literals u_i and $\bar{u}_i$. Each row in D has a one in the corresponding literal column and a one in column b and zeros everywhere else. The rows in K each correspond to a clause in C. Row k_i has a one in every column corresponding to a literal occurring in c_i, as well as a one in the column corresponding to c_i itself. All rows in K also have a one in the columns a and b and zeros elsewhere. Finally, the set R contains only a single row r with ones in all columns c_i as well as in the b column, and zeros everywhere else. An example of this construction for $S = (U, C)$ with $U = \{u_1, u_2, u_3, u_4\}$ and $C = \{\{u_1, u_2\}, \{u_1, \bar{u}_2, u_3, \bar{u}_4\}, \{u_1, \bar{u}_3, u_4\}\}$ is shown in Fig. 5 where the nonzero entries have been numbered according to the elimination scheme corresponding to $(T, F) = (\{u_1, u_4\}, \{u_2, u_3\})$ that will be described next. To complete the reduction, we have to show that M_S has a perfect partial elimination scheme if and only if S is satisfiable. We first show how a satisfying truth assignment for S leads to a perfect partial elimination scheme for M_S.

Let (T, F) be a satisfying truth assignment for S. For every $u_i \in T$, we use row v_i in V to clear the element in the $\bar{u}_i$ column of row w_i in W. Similarly, for every $u_i \in F$, we use row v_i in V to clear the element in the u_i column of row w_i in W. The modified rows in W now represent the truth assignment (T, F). As (T, F) is a satisfying truth assignment for S, we can find a row w_j for each clause row k_l such that $w_j \leq k_l$. We use these rows to clear all elements in the a column of K. Next, the rows from D are used to clear all the elements in the u_i and $\bar{u}_i$ columns of K. The only nonzero elements remaining in K are now the column b and the diagonal in the c_i columns. For every clause c_i we now have that $k_i \leq r$. The rows of K are now used to clear all c_i columns of r, so that the only nonzero element of r that remains is in column b. The remainder of the elimination scheme is rather straightforward: the current row r can be used to clear column b in row sets D and K. The resulting rows

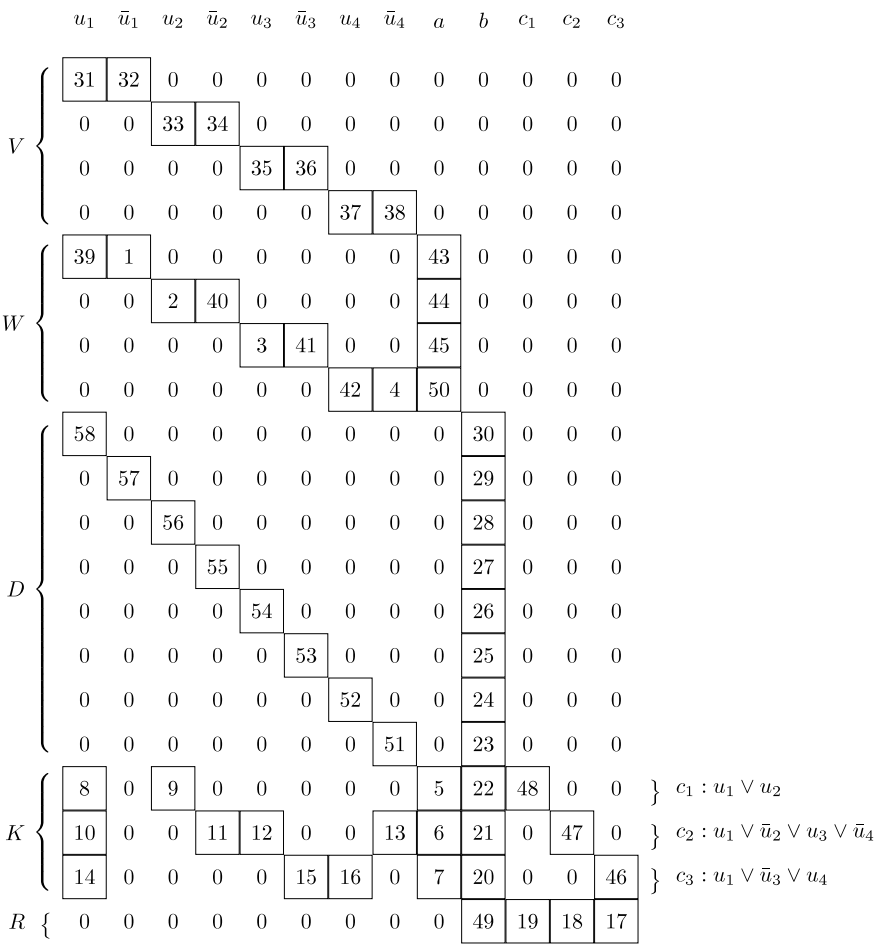

Fig. 5. An example perfect partial elimination scheme for the construction used in the NP-hardness proof of Perfect Partial Elimination (*nonzero entries emphasized by squares*).

in D can then zero all of the literal entries in other rows, leading to a matrix in which for each column there is a row with only a single 1-entry in exactly that column. After that, completion of the elimination scheme is a trivial task. Fig. 5 shows an example of such an elimination scheme where the numbers in the figure represent the ordering from Definition 3.2.

It remains to show that no perfect partial elimination scheme exists if S is not satisfiable. The construction of M_S guarantees that a perfect partial elimination scheme, if one exists, must proceed over three distinct phases that can be characterized as follows.

I During the first phase, only rows from the row sets V, W and D can be used as pivots.
II During the second phase, rows from the row sets R and K can also be used as pivots, but only on other rows from R and K.
III During the final phase, rows from the row sets R and K can also be used as pivots on other rows of M_S.

We proceed by showing why during each phase at least one pivot must be performed before the next phase is reached and why each perfect partial elimination scheme contains pivots in each phase.

Due to the structure of the c_i columns and the assumption that there are at least two clauses, the row sets R and K cannot immediately be used as a pivot on M_S, so initially we can use only rows from the row sets V, W and D as pivots. This is phase I. Using only these rows as pivots, no row with a single 1-entry can be obtained.

Rows of R and K each contain at least a single 1-entry in one of the c_i columns. This means that before they can be used as pivots on rows from V, W or D, at least one row of R and K must be used as a pivot on another row of R and K. When such pivots become possible, we have reached phase II.

To ultimately clear all but a single 1-entry in each of the rows of V, W and D as required for a perfect partial elimination scheme, at least one of the rows of R and K must be used as a pivot on the rows of V, W and D. When such pivots become possible, we have reached phase III.

Before a row from K can be used as a pivot on either another row of K or R, all of its 1-entries in the columns a, u_i and $\bar{u}_i$ must be cleared. The 1-entries in the u_i and $\bar{u}_i$ columns can be cleared using rows from D as pivots. The 1-entry in the a column can only be cleared by using a row from W after clearing a 1-entry in either the u_i or $\bar{u}_i$ column of that row. This

means that for each row k_i of K that we want to use as a pivot, we have to pick some row from W that, after clearing one of its own 1-entries, can be used to clear the 1-entry in the a column of k_i. If there is some way to do this for every row of K, the modified rows from W again represent a satisfying truth assignment, so we have to assume that for every possible usage of the rows from W to clear the a column of the rows of K, there is at least one row for which we cannot clear the 1-entry in the a column. If there is more than one row for which this holds, we can never reach phase III, as none of the rows of K can be applied as pivots to other rows of K and after using the other rows as pivots, row r will always have at least two different 1-entries left in the c_i columns, so it can also never be used as a pivot. So let us assume that there is exactly one row of K for which we cannot clear the 1-entry in the a column. We refer to this row as z. In this case we can use all the other rows of K to clear all but a single 1-entry in the c_i columns of row r. Row r can subsequently be used to clear the last 1-entry in the c_i columns of z. (Row r could also be used to clear the 1 entry in column b instead, but that would clearly block row z from being used as a pivot.) The a column of row z still contains a 1-entry, so in order to use it as a pivot on a row from V, W or D and reach phase III, we will have to subtract it from another row with a 1-entry in the a column. The only rows eligible as operand for such a subtraction are the rows of W. Note that up to this point no row with only a single 1-entry could have been created.

By our assumption, none of the rows of W could be subtracted from z, as this would have enabled us to clear the 1-entry in the a column. This implies that in order to subtract z from any row of W, we have to clear all but a single 1-entry from row z first. This is true as z must contain fewer 1-entries than any row of W we want to subtract it from, because if z contains the same 1-entries, then the subtraction could have been performed the other way round which is impossible by our assumption. Now assume that we have been able to clear all 1-entries of z except for the 1-entry in column a and one other 1-entry. To clear any of these two remaining 1-entries, we require another row with either a single 1-entry, none of which have been created yet, or a row with exactly the same two 1-entries as z, which again contradicts our assumption, as this row must be a row of W. It is therefore impossible to reach phase III if S is not satisfiable and therefore a perfect partial elimination scheme only exists if S is satisfiable. □

Remark 4.2. The problem does not become easier if we restrict ourselves to square matrices. Indeed, we can augment the matrix M_S with additional 'all ones' columns. A moment of reflection shows that these additional columns neither prevent a perfect partial elimination scheme if S has a satisfying truth assignment, nor do they allow such a scheme if S does not have a satisfying truth assignment.

5. Conclusion

When performing Gaussian elimination on sparse matrices, the choice of pivots is critical to preserving sparsity. Ideally, during elimination not a single zero element is turned into a nonzero. Previous work on the relation between Gaussian elimination and elimination schemes on bipartite graphs has led to recognition algorithms for the class of matrices and graphs that allow such perfect elimination schemes. However, by being restricted to a single pivot per column, some possibilities for preserving sparsity may be missed. By clearing single elements at a time instead of performing pivots on entire columns at once, a more fine-grained variant on Gaussian elimination can achieve perfect elimination on a larger class of matrices and their associated bipartite graphs. However, our main result shows that determining whether a matrix allows such a perfect partial elimination scheme is NP-hard.

As our analysis only treats the general case, it may be interesting to also investigate whether more restricted classes of matrices do admit a polynomial time algorithm for partial elimination. Another subject for possible further research could be parameterized versions of the Perfect Partial Elimination decision problem, for example bounding the degree of vertices in the bipartite graph.

Acknowledgment

We gratefully acknowledge the support of the Innovation-Oriented Research Programme 'Integral Product Creation and Realization (IOP IPCR)' of the Netherlands Ministry of Economic Affairs.

References

[1] S.A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC '71, ACM, New York, NY, USA, 1971, pp. 151–158.
[2] M.R. Garey, D.S. Johnson, Computers and Intractability, W.H. Freeman and Company, San Francisco, 1979.
[3] L. Goh, D. Rotem, Recognition of perfect elimination bipartite graphs, Information Processing Letters 15 (4) (1982) 179–182.
[4] M.C. Golumbic, C.F. Goss, Perfect elimination and chordal bipartite graphs, Journal of Graph Theory 2 (2) (1978) 155–163.
[5] D.J. Rose, R.E. Tarjan, Algorithmic aspects of vertex elimination on directed graphs, SIAM Journal on Applied Mathematics 34 (1) (1978) 176–197.
[6] D.J. Rose, R.E. Tarjan, G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, SIAM Journal of Computing 5 (2) (1976) 266–283.
[7] J.P. Spinrad, Recognizing quasi-triangulated graphs, Discrete Applied Mathematics 138 (1–2) (2004) 203–213.