Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

Matrix Robustness, with an Application to Power System Observability

Matthias Brosemann,[1] Jochen Alber, Falk Hüffner,[2] and Rolf Niedermeier

abstract. We initiate the study of the computational complexity of Matrix Robustness: Check whether deleting any k rows from a full-rank matrix does not change the matrix rank. This problem is motivated by applications in observability analysis of electrical power networks. We indicate the coNP-completeness of Matrix Robustness, provide linear programming based solutions (both exact and heuristic) and a problem-specific algorithm, and present encouraging experimental results.

1 Introduction

Observability analysis is a key issue in the management of power systems: Energy supply companies continuously monitor the power flow of their electrical network such that, e.g., in the event of an unexpected behavior emergency actions can be initiated. In order to retrieve the state of the power system in operation, various electrical quantities in the network are repeatedly measured. The network is said to be *observable* if all system states can be estimated based on the given measurements.

Checking whether the installed measurements suffice for estimating the entire state of the system is known as *observability analysis* in the electrical engineering literature (see [11] for an overview). In this paper, we follow a recently proposed approach for observability analysis [1]. There, it is shown that the network is observable iff the rank of a corresponding discrete *sensitivity matrix* that is derived from the electrical specification of the network is full. Based on this new correspondence, we can extend the classic observability analysis scenario to the practically important scenario where one asks whether the network states remain observable even if up to k arbitrary meters (corresponding to matrix rows) fail.

[1]Supported by the Deutsche Forschungsgemeinschaft (DFG), project PEAL (parameterized complexity and exact algorithms), NI 369/1.

[2]Supported by the DFG, Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4.

Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

Matrix Robustness
Instance: An $m \times n$ matrix M over an arbitrary field $\mathbb{F}$ with full rank n, $m \geq n$, and an integer $k > 0$.
Question: Is M robust against deletion of k rows, that is, is the rank of M preserved if any k rows are deleted?

For convenience, we sometimes consider Matrix Weakness, the complement of this decision problem, that is, the question whether we can find k rows such that when they are deleted, the matrix drops in rank.

2 Complexity of Matrix Robustness

In this section, we show that Matrix Weakness is many-one equivalent to the Generalized Minimum Distance problem from coding theory [13], while preserving the underlying fields. This implies that Matrix Robustness is coNP-complete for finite fields, and makes it plausible that it is also coNP-complete for infinite fields as they are used in our application.

We show the equivalence of Matrix Weakness and Generalized Minimum Distance via the Generalized Minimum Circuit problem from matroid theory [12] as an intermediate step.

Generalized Minimum Circuit
Instance: An $m \times n$ matrix M over an arbitrary field and a positive integer k.
Question: Is there a linearly dependent subset of the column vectors of M with at most k elements?

For this, we need some definitions and observations from matroid theory [12].

DEFINITION 1. Consider a ground set E and its power set $\mathcal{P}(E)$. A pair $M = (E, \mathcal{I})$ with $\mathcal{I} \subseteq \mathcal{P}(E)$ is called a *matroid* if

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

(I2) $A \in \mathcal{I}, B \subseteq A \Rightarrow B \in \mathcal{I}$, and

(I3) $A, B \in \mathcal{I}$, $|B| > |A| \Rightarrow \exists a \in B \setminus A : A \cup \{a\} \in \mathcal{I}$.

A set $A \in \mathcal{I}$ of a matroid $(E, \mathcal{I})$ is called *independent set*. Conversely, a set $A \subseteq E$ with $A \notin \mathcal{I}$ is called *dependent set*. A set $B \in \mathcal{I}$ is a *basis* if B is a maximal independent set of M, i. e., $\forall B' \in \mathcal{I} : B \subseteq B' \Rightarrow B' = B$. A dependent set C is called a *circuit* if C is a minimal dependent set of M,

Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

i. e., $\forall C' \subsetneq C : C' \in \mathcal{I}$. The cardinality of a basis of a matroid is called the *rank* of this matroid (it is easy to prove that all bases have the same size).

To get a connection to Matrix Weakness, we now focus on matroids that represent a set of vectors.

DEFINITION 2. Let A be an $m \times n$ matrix over a field $\mathbb{F}$, let E be a set of column labels of A, and let $\mathcal{I}$ be the set of subsets of E that represent sets of linearly independent column vectors of A. The tuple $(E, \mathcal{I})$ is then called the *vector matroid* $M[A]$ of A.

A central proof tool in matroid theory is the concept of duality.

DEFINITION 3. The *dual matroid* $M^* = (E, \mathcal{I}^*)$ of the matroid $M = (E, \mathcal{I})$ is defined by the set of its independent sets

$$\mathcal{I}^* = \{X \subseteq E : \exists \text{ basis } B \text{ of } M \text{ with } B \cap X = \emptyset\}.$$

Two results from matroid theory are that the dual matroid is a matroid, and that the duality is closed over the set of vector matroids [12]. With this we can now state the following theorem.

THEOREM 4. Matrix Weakness *on a field* $\mathbb{F}$ *is many-one equivalent to* Generalized Minimum Circuit *on* $\mathbb{F}$. *The matrices of both problems can be transformed into each other in polynomial time.*

Proof. At first we consider the input matrix M of Matrix Weakness in its transposed form $A := M^T$, asking whether there is a set of column vectors of A such that the matrix drops in rank when they are deleted.

Next we consider equivalently the vector matroid $M[A] = (E, \mathcal{I})$ and ask whether we are able to find k elements of E such that the matroid drops in rank when they are deleted. We can do this because the independent sets of A and $M[A]$ are equal (up to different labeling).

Now if we have such k elements—say as a set L—getting the matrix to drop in rank when they are deleted, then all bases of $M[A]$ have a nonempty intersection with L, since otherwise $M[A]$ is still of full rank after the deletion of the k elements. Conversely, if there is a set L with k elements of E having a nonempty intersection with each basis of $M[A]$, then these k elements can be deleted to get the matrix to drop in rank. Hence, the question whether there are k elements in E so that the matroid drops in rank when they are deleted is equivalent to the question whether there is a set L with k elements that has a nonempty intersection with each basis of $M[A]$.

Further, such a set L is a dependent set of the dual matroid of $M[A]$. So it is equivalent to ask whether there is a dependent set of the dual matroid of $M[A]$ containing k elements. This is in fact the problem Generalized Minimum Circuit if we consider the matrix form of the dual of $M[A]$.

Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

Besides the equivalence between both problems we have to consider the runtime of matrix operations used in the proof above. The transpose of a matrix can be calculated in polynomial time, and it is well-known that the Gaussian elimination of an $n \times m$ matrix of rank r to the form $[I_r|D]$, where I_r is the $r \times r$ identity matrix and D an arbitrary $r \times (m-r)$ matrix, can be done in polynomial time, too, and does not change the set of independent vectors. In addition, it is proved in matroid theory (see [12]) that the dual of the vector matroid $[I_r|D]$ is the vector matroid of $[-D^T|I_{m-r}]$. Thus, all the transformations used above can be done in polynomial time. ■

We now demonstrate that Generalized Minimum Circuit is equivalent to Generalized Minimum Distance, completing the chain of equivalences from Matrix Weakness to Generalized Minimum Distance. The Minimum Distance problem is a problem from coding theory: A *linear code* is a subspace of a vector space $\mathbb{F}^n$ over a field $\mathbb{F}$. The *Hamming weight* $\mathrm{wt}(x)$ of a vector $x \in \mathbb{F}^n$ is the number of nonzero entries of x. In traditional coding theory, only vector spaces over finite fields such as GF(2) are considered. However, the sensitivity matrix from our application is over the infinite field $\mathbb{R}$, and we defined Matrix Robustness for arbitrary fields. Therefore, we need a generalization of the Minimum Distance problem to arbitrary fields.

> Generalized Minimum Distance
> **Instance:** An $m \times n$ matrix H over an arbitrary field $\mathbb{F}$ and integers $n, k > 0$.
> **Question:** Is there a nonzero vector $x \in \mathbb{F}^n$ of weight $\mathrm{wt}(x) \leq k$ with $H \cdot x = 0$?

The equivalence of Generalized Minimum Circuit and Generalized Minimum Distance is now easy to see. Together with Theorem 4, we obtain the central result of this section.

THEOREM 5. Matrix Weakness *is equivalent to* Generalized Minimum Distance.

Since it is known that Generalized Minimum Distance is NP-complete for any finite field [13], we obtain the following for the complement of Matrix Weakness.

COROLLARY 6. Matrix Robustness *is coNP-complete for any finite field.*

Unless RP = NP, there is no polynomial-time constant factor approximation for Minimum Distance [6]. Since our reductions are approximation-

Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

preserving, the same holds for Matrix Robustness restricted to arbitrary finite fields.

McCormick [10] claims NP-completeness for Generalized Minimum Circuit on any infinite field. This would have settled the question for the coNP-completeness of Matrix Robustness on infinite fields, too. Unfortunately, although cited several times in the literature, the corresponding reduction from Clique contains an error that is not obvious how to fix [2].

For the field GF(2), there are several polynomial-time probabilistic algorithms [7, 9] for Minimum Distance, but as far as we know, there are no algorithms for Generalized Minimum Distance.

3 Algorithms for Matrix Robustness

Matrix Robustness is coNP-complete for finite fields (Corollary 6). It seems plausible that this still holds for infinite fields, over which the sensitivity matrix from our application is defined. Therefore, we develop algorithms based on mixed integer programming (Sect. 3.1), and present several speedups for it in Sect. 3.2. The employed ideas also lead to a linear programming formulation, which can solve Matrix Robustness in provably polynomial time, albeit without guarantees for the solution quality (Sect. 3.3). Further, a relaxation of the notion of rank called *pseudorank* that has been previously suggested in the context of observability analysis leads to another polynomial-time algorithm presented in Sect. 3.4.

3.1 Mixed Integer Formulation

It is convenient to reformulate our problem in terms of vector spaces. Because the input matrix is of full rank, its row vectors span an n-dimensional space. If the rank is reduced by one, then the row vectors span an $(n-1)$-dimensional subspace, that is, a hyperplane. Hence, it is equivalent to Matrix Weakness to ask whether there is a hyperplane containing $n-k$ row vectors. The other k vectors are those that have to be deleted to get the matrix to drop in rank.

THEOREM 7. Matrix Robustness *is solvable using Mixed Integer Programming (MIP).*

Proof. We are looking for a hyperplane H that includes as many row vectors of the matrix as possible. To model whether a row vector y_i lies in H, we introduce binary variables d_i with the following characterization: $d_i = 0$ iff y_i lies in the hyperplane H. Given this, the goal is to minimize the sum of the d_i's. The hyperplane H we are looking for is represented by its normal vector x. The only constraint x has to satisfy is that it must not be the null vector. This requirement is considered at the end of this proof.

Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

To test whether a vector y_i lies in H, we use the scalar product $\langle \cdot, \cdot \rangle$ between the row vectors and the normal vector: $\langle y_i, x \rangle = 0$ iff y_i lies in the hyperplane defined by x. This is implemented with the variables d_i and two constraints per row vector: $\langle y_i, x \rangle - d_i \leq 0$ and $-1 \cdot \langle y_i, x \rangle - d_i \leq 0$. With the two additional preconditions $||y_i|| \leq 1$ and $||x|| \leq 1$ it holds that $-1 \leq \langle y_i, x \rangle \leq 1$. It is now easy to show that the binary variables d_i indeed behave as desired.

The precondition $||y_i|| \leq 1$ can be obtained by dividing every y_i^j by $||y_i||$. In contrast, $||x|| \leq 1$ has to be demanded explicitly in the formulation of the MIP; it can be achieved with a simple constraint per component of x: $-1/n \leq x_i \leq 1/n$.

Finally, the precondition $||x|| > 0$ (that is, $x \neq 0$) remains to be implemented. But this cannot be achieved with simple inequalities, because it is not a convex characteristic. It could be implemented by using quadratic programming. However, solvers for quadratic programming are less readily available, and the added expressive power makes them much slower. Therefore, the approach we chose is to solve several MIPs. If $x \neq 0$, then there must be at least one entry $x_i \neq 0$ (w. l. o. g. $x_i > 0$), and we can solve the problem by solving n MIPs for $i = 1$ to n, each time adding the constraint $x_i \geq c$ for a sufficiently small constant c that does not contradict the constraint $-1/n \leq x_i \leq 1/n$ (e. g., $x_i \geq 1/n$). The solution for the input instance is then the best solution from the n MIPs. ■

3.2 Improvements for the MIP Formulation

The following improvements can speed up the overall solving process exploiting the fact that n very similar MIPs are solved in a loop for one input instance.

Improvement 1: Exploitation of partial solutions. We want to determine the global minimum over the solutions of all n MIPs and do this by solving them successively. In this process, solutions of previously solved MIPs can be used to constrain further MIPs, helping the solver to cut down the search space. For this a new constraint is introduced: $\sum_{j=1,\ldots,m} d_j \leq k_{\text{best}}$. Here, k_{best} is replaced by the currently best minimum from the previous subproblem (the first subproblem is solved without this constraint).

Improvement 2: Pairwise linear dependencies. Alber and Pöller suggested to exploit pairwise linear dependencies between rows of the sensitivity matrix [1]. We implement this idea in our context by a more invasive change to the MIPs. The pairwise linear dependence of vectors is an equivalence relation, and therefore partitions the input vectors into equivalence classes. If one vector of such a class is deleted, then all other vectors from this class have to be deleted, too, since otherwise the deletion cannot affect the ma-

Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

trix rank. Therefore, we can replace a class of pairwise linearly dependent vectors by a representative vector, and in the objective sum give a weight to each d_i equal to the cardinality of the corresponding class. Determination of equivalence classes and weights can be done with a simple polynomial-time preprocessing.

Improvement 3: Time limitation. To get a view at suboptimal results at an early stage of the overall solving process, the solver is called for the n MIPs with a defined runtime limit. All MIPs that were not solved are enqueued. After one pass, all enqueued problems are attempted to be solved again, but with a twice as large time-frame. This is repeated until all MIPs are solved. The time required with this technique is at most three times longer than without it, but quickly solvable subproblems are calculated earlier.

The time-frame technique has further advantages arising from the combination with the other speed-ups. For instance, the smallest reachable minimum could be found earlier. Also (suboptimal) results are calculated faster and can be used as bounds for other MIPs.

3.3 Linear Programming Formulation

Mixed integer programming is NP-hard, and therefore the method described in Sect. 3.1 probably takes exponential time. Since Matrix Robustness (over finite fields) is a coNP-complete problem, if we want a provably fast (polynomial) algorithm, we probably have to lower our expectations with respect to solution quality. To obtain a polynomial-time algorithm without guarantee on solution quality, we again consider the hyperplane formulation, where the goal is to find a hyperplane that contains as many row vectors of the input matrix as possible. The idea is to successively eliminate the "worst" vectors, that is, those most "unlikely" to be in the goal hyperplane, until all of the remaining vectors lie in a hyperplane.

For this, we iteratively consider the *consensus hyperplane* that minimizes the sum of the scalar products between each row vector of the input matrix and the normal vector of the hyperplane. The vector with the largest absolute value of its scalar product to the normal vector of the consensus hyperplane is eliminated. This is repeated until all remaining vectors lie in the consensus hyperplane.

To find the consensus hyperplane, we can use a linear program (LP) very similar to the MIP from Sect. 3.1. The only difference is that the variables d_i now are real instead of binary; they now measure the "degree of containedness" instead of deciding containedness in the hyperplane. We overcome the obstacle of the nonconvex constraint $||x|| > 0$ in the same way as for the MIP by solving n LPs. Since each LP is solvable within polynomial

Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

time, and we repeat the search for the worst vector only n times, the overall algorithm has a polynomial runtime.

The preprocessing of eliminating pairwise linear dependencies (Sect. 3.2, Improvement 2) can also be applied to this linear programming formulation.

3.4 Pseudorank-Based Algorithm

Alber and Pöller [1] introduced the concept of a *pseudorank* of a matrix, which is a cheap to calculate upper bound to the rank of the matrix. They observed that using pseudorank instead of rank for the observability of networks is sufficient in the practical settings they considered. We show that Matrix Robustness with respect to the pseudorank (that is, given a $m \times n$ matrix deciding whether the pseudorank remains at least n when deleting any k rows) can be solved in polynomial time.

The pseudorank is defined as the minimum of the number of rows and the number of columns after exhaustive elimination of pairwise linear dependencies (as described in Sect. 3.2 (Improvement 2)) both within rows and within columns. It can be calculated by eliminating all pairwise linear dependencies within the rows, then within the columns, and then repeating this until no further change takes place. Hence, the pseudorank is an upper bound to the rank. Therefore, if a matrix is robust against deletion of k rows, then it is pseudorank robust, too.

If we do not want to calculate the pseudorank, but just decide whether it is full, it suffices to do a single elimination pass over the rows followed by a single elimination pass over the columns: If columns were deleted, then the pseudorank is not full; otherwise, the situation of the rows remains unchanged and no further passes are required. The idea of our algorithm for pseudorank robustness is to follow this simple scheme, but take into account that any k rows are being deleted first, without trying all $\binom{m}{k}$ possibilities explicitly.

THEOREM 8. Matrix Robustness *with respect to the pseudorank can be solved in* $O(s \cdot m \log m)$ *time for an* $m \times n$*-matrix, where* s *is the number of nonzero matrix entries.*

Proof. As explained, if an $m \times n$-matrix M is to be not robust in terms of pseudorank, then one of three conditions must hold:

1. After deleting k rows and then eliminating pairwise linearly dependent rows, there are less than n rows left.
2. After deleting k rows, there are two dependent columns.
3. After deleting k rows, there is a zero column.

Proc. 2nd ACiD-06, Vol. 7 Algorithmics, pp. 37-48, College Publications, 2006

We delete zero rows upfront, since they do not affect the worst-case scenario, and then check the three conditions successively. Conditions 1 and 3 are easy to check. We check Condition 2 separately for all pairs (M_i, M_j) of columns, that is, we try to determine a factor c such that $M_j = c \cdot M_i$ after deleting k rows. For a row r, there are three cases:

- $M_{ri} \neq 0$ and $M_{rj} \neq 0$. Then $c = M_{rj}/M_{ri}$, or r needs to be deleted.
- Exactly one of M_{ri} and M_{rj} is 0. Then r needs to be deleted in any case.
- $M_{ri} = M_{rj} = 0$. Then it is never necessary to delete r.

Therefore, we determine all possible factors for rows r with $M_{ri} \neq 0$ and $M_{rj} \neq 0$. If the number of rows that require a different factor than the most frequent one plus the number of rows where exactly one of M_{ri} and M_{rj} is 0 exceeds k, then the columns i and j cannot be made linearly dependent by deleting at most k rows; otherwise M is not robust with respect to the pseudorank.

We omit the details of the runtime analysis. ■

4 Experimental Results and Conclusions

We implemented the MIP formulation from Sect. 3.1 and the LP formulation from Sect. 3.3, including the improvements from Sect. 3.2, and the pseudorank algorithm from Sect. 3.4, and tested them on real and synthetic instances. The MIPs and LPs themselves are implemented in the GNU MathProg modeling language and solved by the solver *glpsol* from the GNU Linear Programming Kit. The pseudorank heuristic is implemented as a Python script. The testing machine is an AMD Athlon 64 3400+ with 2.4 GHz, 512 KB cache, and 1 GB main memory, running under the Debian GNU/Linux 3.1 operating system.

Calculating Matrix Robustness of Electrical Networks. Four networks are mostly designed for testing purposes ("IEEE" is from a benchmark by the IEEE); the fifth is based on the Namibian power network operated by NamPower. The discretization of the resulting sensitivity matrix is to the integer interval $\{-9, \ldots, 9\}$, except for the Namibian network, where the sensitivity matrix is discretized to the interval $\{-999, \ldots, 999\}$. These intervals are chosen based on numerical properties of the matrices.

The results and runtimes of both the exact algorithm with different speedups and the heuristics are shown in Table 1. Clearly, exploiting upper bounds (Improvement 1) and exploiting equivalence classes (Improvement 2) is always worthwhile. With these improvements, the MIP can solve all instances optimally within reasonable time. For the MIP, the runtime is not

Table 1. Runtimes for some Matrix Robustness instances. Dimension is after elimination of all-zero rows, Dimension/EC (equivalence classes) after additionally subsuming pairwise dependent vectors. The result is the number of rows that have to be deleted to get the matrix to drop in rank. For the runtimes, simple means plain algorithm without speed-up techniques, EC means exploiting equivalence classes, and UB means exploiting upper bounds determined by previous MIPs in a pass. Finally, LP is the runtime for the linear programming based approach and PR the runtime for the pseudorank heuristic.

	Dimension		Result		Runtime in seconds					
		EC	opt.	LP	Simple	EC	UB	EC&UB	LP	PR
Treelike	18×8	8×8	2	2	0.07	0.05	0.08	0.05	0.04	0.02
MV/LV	78×12	23×12	2	2	1.25	0.18	0.40	0.15	0.14	0.04
Nine-Bus	40×12	28×12	4	12	982.29	26.09	33.26	17.61	1.15	0.03
IEEE	150×29	45×29	2	6	11.74	1.41	4.01	1.18	3.32	0.15
Namibia	411×164	306×164	1	7	—	—	687.02	477.09	1426.06	4.70

alone determined by the size of the input, as can be seen when comparing "Nine-Bus" and "IEEE". Although larger, the latter is solved much quicker, possibly because of the significantly higher sparsity (ratio of zero entries in the sensitivity matrix). The pseudorank based algorithm achieves an optimal result for all instances, while the LP method gives rather bad results for some instances.

Note that the column "Result" in Table 1 shows that for our current real-world instances the solution sizes are rather small and probably complete enumeration of row subsets in order of increasing size would work as well. However, larger solution sizes are conceivable, and therefore we study randomly generated instances with larger solution sizes in what follows.

Test Results with Synthetic Input Data. To get a better impression of the range of problems where we can employ the exact MIP based algorithm as opposed to the heuristics, and to estimate the solution quality of the heuristics, we tested the algorithms on random instances. For this, we generated matrices with random entries and dimension, sparsity, and entry range similar to that from our real-world instances. The results are shown in Fig. 1. Clearly, these random instances are harder than the real-world instances. We can also clearly see the expected exponential runtime of the exact MIP algorithm. For $n > 13$, the MIP formulation is no longer feasible and we have to use one of the heuristics, which can still solve problems with $n = 45$ within a few minutes. Moreover, runtimes are much more pre-

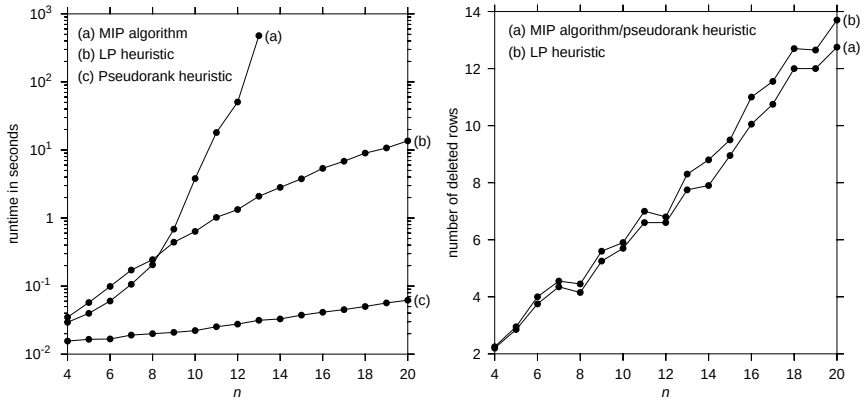

Figure 1. Runtime and solution quality for random matrices of size $5n \times n$, with entries from $\{-9, \ldots, 9\}$ and 80 % sparsity. Each point is an average over 20 random instances.

dictable for the heuristics, with the standard deviation being only 22 % for the LP heuristic and 7 % for the pseudorank heuristic as opposed to 128 % for the MIP for $n = 13$.

In all tested instances the pseudorank heuristic produced an optimal result. The LP heuristic was optimal for 81 % of the instances, but was sometimes off by up to 50 % (see Fig. 1 (right)).

Summary and Recommendations. It seems advisable to start with the MIP algorithm, which gives provably optimal results and can solve many practically relevant instances. If this takes too long, the pseudorank algorithm is the best choice, since it was superior in both runtime and quality to the LP based method for all instances. However, our method of generating random instances might favor the pseudorank algorithm, since sets of linearly dependent vectors that do not contain pairwise linearly dependent vectors are unlikely. Therefore, in some scenarios "consensus hypeplane"-based methods, possibly with improved efficiency as compared to the LP method, might be superior.

Future Challenges. An immediate open question is to find out whether Matrix Robustness remains coNP-complete in case of infinite fields. To investigate polynomial-time approximation algorithms with guaranteed approximation ratio is open. Downey et al. [5] conjectured that an equivalent problem is parameterized intractable with respect to parameter k. Thus, can W[1]-hardness [4] be shown for Matrix Robustness with respect to parameter k? From an application point of view, it would make sense to

further distinguish between meters with different degrees of reliability. Finally, it would be interesting to closer investigate the connection with meter placement problems such as studied in [3, 8].

Acknowledgment. We thank Geoff Whittle (Victoria University of Wellington, NZ) for helpful suggestions for the material in Sect. 2.

BIBLIOGRAPHY

[1] J. Alber and M. Pöller. Observability of power systems based on fast pseudorank calculation of sparse sensitivity matrices. In *Proc. IEEE Power Engineering Society: Transmission and Distribution Conference*, 2005.

[2] M. Brosemann. Matrix robustness: Algorithms, complexity and an application in network observability (in German). Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2006.

[3] D. J. Brueni and L. S. Heath. The PMU placement problem. *SIAM Journal on Discrete Mathematics*, 19(3):744–761, 2005.

[4] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[5] R. G. Downey, M. R. Fellows, A. Vardy, and G. Whittle. The parametrized complexity of some fundamental problems in coding theory. *SIAM Journal on Computing*, 29(2):545–570, 1999.

[6] I. Dumer, D. Micciancio, and M. Sudan. Hardness of approximating the minimum distance of a linear code. *IEEE Transactions on Information Theory*, 49(1):22–37, 2003.

[7] A. Foster. A polynomial-time probabilistic algorithm for the minimum distance of an arbitrary linear error-correcting code. Honors Thesis, United States Naval Academy, Annapolis, 2003.

[8] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, and M. A. Henning. Domination in graphs applied to electric power networks. *SIAM Journal on Discrete Mathematics*, 15(4):519–529, 2002.

[9] J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34:1354–1359, 1988.

[10] S. T. McCormick. *A combinatorial approach to some sparse matrix problems*. PhD thesis, Stanford University, Stanford, California, 1983. Published in SOL 83-5.

[11] A. Monticelli. Electric power system state estimation. *Proceedings of the IEEE*, 88(2):262–282, 2000.

[12] J. G. Oxley. *Matroid Theory*. Oxford University Press, 2004.

[13] A. Vardy. The intractability of computing the minimum distance of a code. *IEEE Transactions on Information Theory*, 43(6):1757–1766, 1997.

Matthias Brosemann, Falk Hüffner, Rolf Niedermeier

Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany

{matthias.brosemann,hueffner,niedermr}@minet.uni-jena.de

Jochen Alber

DIgSILENT GmbH, Power System Applications & Consulting, Heinrich-Hertz-Straße 9, D-72810 Gomaringen, Germany

j.alber@digsilent.de