Journal of Optimization Theory and Applications
https://doi.org/10.1007/s10957-023-02280-7

Sufficient Matrices: Properties, Generating and Testing

Marianna E.-Nagy[1] · Tibor Illés[1] · Janez Povh[2] · Anita Varga[3] · Janez Žerovnik[2]

Received: 12 December 2022 / Accepted: 18 July 2023

Abstract

This paper investigates various aspects of sufficient matrices, one of the most relevant matrix classes introduced in connection with linear complementarity problems. We summarize the most important theoretical results and properties related to sufficient matrices. Based on these, we propose different construction rules that can be used to generate new matrices that belong to this class. A nonnegative number can be assigned to each sufficient matrix, which is called its handicap and works as a measure of sufficiency. The handicap plays a crucial role in proving convergence and complexity results for interior point algorithms for linear complementarity problems. For a particular sufficient matrix, called Csizmadia's matrix, we give the exact value of the handicap, which is exponential in the size of the matrix. Another important topic that we address is deciding whether a matrix is sufficient. Tseng proved in 2000 that this decision problem is co-NP hard. We investigate three different algorithms for determining the sufficiency of a given matrix: Väliaho's algorithm, a linear programming-based

Communicated by Goran Lesaja.

✉ Marianna E.-Nagy
marianna.eisenberg-nagy@uni-corvinus.hu

Tibor Illés
tibor.illes@uni-corvinus.hu

Janez Povh
janez.povh@fs.uni-lj.si

Anita Varga
vanita@math.bme.hu

Janez Žerovnik
janez.zerovnik@fs.uni-lj.si

1 Corvinus Centre for Operations Research, Corvinus Institute of Advanced Studies, Corvinus University of Budapest, Fővám tér 8., Budapest 1093, Hungary

2 Faculty of Mechanical Engineering, University of Ljubljana, Aškerčeva ul. 6, 1000 Ljubljana, Slovenia

3 Department of Analysis and Operations Research, Institute of Mathematics, Budapest University of Technology and Economics, Müegyetem rkp. 3., Budapest 1111, Hungary

Published online: 17 August 2023

algorithm, and an algorithm that facilitates nonlinear programming reformulations of the definition of sufficiency. We tested the efficiency of these methods on our recently launched benchmark data set that consists of four different sets of matrices. In this paper, we give the description and most important properties of the benchmark set, which can be used in the future to compare the performance of different interior point algorithms for linear complementarity problems.

Keywords Sufficient matrices · $\mathcal{P}_*(\kappa)$-matrices · Linear complementarity problems · Principal pivot operations

1 Introduction

Several matrix classes have been introduced either because of the study of some properties of linear complementarity problems (LCPs), or have become known because they have important, influential properties on the solvability of LCPs.

The focus of our interest is the sufficient matrix class, since an LCP with such a coefficient matrix (sufficient LCP) can be solved with the criss-cross pivot algorithm (CCA) in finite steps [6–8, 14] or can be solved with an interior point algorithm (IPA) in polynomial time, see [2, 10–12, 17, 28, 35, 36, 38, 41, 48] and references in these papers. Good surveys on IPAs for sufficient LCPs can be found in the PhD theses of Nagy [47] and Rigó [50].

There is a need to test the practical efficiency of these solution methods. However, there is no real benchmark for sufficient matrices. There were already some attempts to propose sufficient matrix instances to study the numerical performance of IPAs. Numerical studies started with Gurtuna et al. [25], and the authors used block diagonal sufficient matrices built from sufficient matrices of size 2×2 and 3×3 with different handicap values. (The handicap, usually denoted κ, is a very important parameter of sufficient matrices, for the definition, see the next section.) They have already pointed out the necessity of building benchmark sufficient LCPs for testing IPAs. Another early attempt was by Kheirfam [38], who defined some very special, non-symmetric matrices with $\kappa = 0$, namely positive semidefinite (PSD) matrices and compared three different IPAs. All these restricted computational studies were related to the fact that only small, structured sufficient matrices were known.

An interesting question related to sufficient matrices was how to construct such sufficient matrices that have small bit length and very large (i.e., exponential) κ value. Csizmadia constructed a very simple sufficient matrix with ± 1 and 0 entries, which has exponential entries after a handicap-invariant transformation (the principal pivotal transformation). Therefore, de Klerk and E.-Nagy [13] considered it as a good candidate and indeed, gave a lower bound on its handicap, which exponentially depends on the size of the matrix. In this paper, we prove that the lower bound is exactly the handicap.

However, for a systematic study of generating sufficient matrices, we had to wait until 2018 for the paper by Illés and Morapitiye [27]. Although the theoretical basis of their study is well founded, there are some pitfalls, like the small values of κ parameters of the generated matrices and the numerical errors during constructions that caused

the loss of the sufficiency property for a few of the constructed matrices. More precise constructions and more careful computations would lead to a better set of sufficient matrices even in the near future. In this paper, we collected a wider range of tools and also completed proofs for the tools from [27] to generate sufficient matrices.

The sufficient matrices generated by Illés and Morapitiye [27] and the introduction of Csizmadia's matrix made it possible to study the computational performance of newly developed IPAs for sufficient LCPs and compare those with algorithms known from the literature. The first steps in this direction have been taken by Darvay et al. [10–12] in a series of papers. The numerical study of IPAs for sufficient LCPs became a standard, required section of new publications like in Illés et al. [35, 36] and E.-Nagy and Varga [17].

In order to build a benchmark set of sufficient matrices, it is important not only to construct such matrices but also to be able to test the sufficiency of a matrix. For a long time, up to our knowledge, the only method to decide whether a matrix is sufficient or not has been proposed by Väliaho in 1996 [57]. A year later, Väliaho [59] published an algorithm for computing the handicap of a given sufficient matrix. Tseng [56] proved that deciding whether a square matrix with rational entries is a column sufficient matrix is a co-NP-complete problem. Thus, it is not surprising that both of Väliaho's algorithms have exponential running time. In addition to Väliaho's algorithm, we implemented a linear programming based procedure [49], as well. We give a simplified description of the latter one. However, it was revealed in the meantime that the idea of the linear programming-based approach was already mentioned by Guu and Cottle [26]. Finally, we compare the running times of these MATLAB implementations with a third possibility: we reformulate the definition of sufficiency to a nonlinear programming problem, and we solve it using BARON.

Finally, we would like to discuss an interesting type of result, the so-called EP-theorems. In the setting of LCPs, EP-theorems showed up in the paper of Fukuda et al. [22] as generalizations of the LCP alternative theorem of Fukuda and Terlaky [23]. Shortly, the main idea of Fukuda, Tamura, and Namiki [22] was that the CCA can be modified to handle general LCPs. Namely, the modified CCA tries to solve a given general LCP starting from an arbitrary basis, and either (i) solves the LCP, or (ii) solves its dual (which proves that the LCP has no solution), or (iii) gives a certificate that the coefficient matrix is not sufficient. The three stopping criteria of the modified CCA are called the *solution of the general LCP in EP-sense*. The advantage of the modified CCA is that it can solve many general LCPs that have a solution before identifying that its matrix is not sufficient. However, this might depend on the basis from which the modified CCA has been started. An interesting example of LCPs with nonsufficient matrices is related to the Arrow–Debreu market exchange model [1]. For the connections of the Arrow–Debreu market exchange model to LCPs and IPAs, see Ye [60]. Csizmadia et al. [8] tested the modified CCA to solve such LCPs that possess the properties of the LCPs derived from the Arrow–Debreu market exchange model with Leontief utilities. This preliminary computational study showed that the modified CCA for larger size instances stops much more frequently with the certificate that the matrix is not sufficient than computes a solution, although it is known that these LCPs always have a solution.

Illés, Nagy, and Terlaky [29–31] investigated the possibility of replacing the modified CCA in the proof of an EP-theorem for LCPs. It turned out that the EP-theorem of Fukuda, Tamura, and Namiki for LCPs [22] could not be proved using IPAs. When IPAs are applied to obtain an EP-theorem for LCPs, instead of the whole sufficient matrix class, it should be restricted to a large, but fixed subclass of sufficient matrices, namely to $\mathcal{P}_*(\bar{\kappa})$-matrices, with an arbitrarily large, but given $\bar{\kappa} > 0$ parameter. However, the improvement in the application of modified IPAs instead of modified CCA was that the solution of the general LCP in EP-sense was computed in polynomial time [30, 31]. Nagy [47] tested the modified IPAs on LCPs similar to those used by Csizmadia et al. [8] in their computational studies of the modified CCA. Nagy obtained very similar results to Csizmadia and his coauthors, namely, when the size of the matrix is growing, it is less likely that the modified IPA for this type of problem will stop with a solution of the general LCP than with a certificate that the matrix is not a $\mathcal{P}_*(\bar{\kappa})$-matrix. Some further research would be necessary to solve the Arrow–Debreu market exchange model with Leontief utility functions more efficiently using modified IPAs. This paper can be considered one of the first important steps in this direction since we need to understand better the sufficient matrix class to achieve this aim.

The rest of our paper is organized as follows. In Sect. 2, we summarize the preliminary results about matrix classes. Known and new construction rules to get sufficient matrices are provided in Sect. 3. Here, we also deal with lower triangular matrices as a special case and prove the exact value of the handicap of Csizmadia's matrix. Section 4 describes three different approaches for deciding whether a given matrix is sufficient or not. First, we recall Väliaho's algorithm, then we propose an LP-based method, and finally, we give nonlinear programming reformulations of the definition of sufficiency. In Sect. 5, we provide implementation details and our numerical results.

Notation We use the following notation throughout the paper. Scalars and indices are denoted by lowercase Latin letters, vectors by lowercase boldface Latin letters, matrices by capital Latin letters, and sets by capital calligraphic letters. Let $\mathbb{R}^n_\oplus$ ($\mathbb{R}^n_+$) denote the nonnegative (positive) orthant of $\mathbb{R}^n$. I and O denote the identity and the all-zero matrix of appropriate dimension, respectively, and X is the diagonal matrix whose diagonal elements are the coordinates of the vector $\mathbf{x}$, i.e., $X = \operatorname{diag}(\mathbf{x})$. The vector $\mathbf{x} \circ \mathbf{s} = X\mathbf{s}$ is the componentwise product (Hadamard product) of the vectors $\mathbf{x}$ and $\mathbf{s}$. The ith entry of a vector $\mathbf{x}$ is denoted by x_i. If $A \in \mathbb{R}^{n\times n}$, $A_{\mathcal{J}\mathcal{K}}$ denotes the submatrix of A with rows indexed by the index set $\mathcal{J} \subset \{1, \ldots, n\}$ and columns by the index set $\mathcal{K} \subset \{1, \ldots, n\}$. We denote the vector of ones by $\mathbf{e}$ and the ith standard unit vector by $\mathbf{e}_i$.

2 Matrix Classes

Let us define some matrix classes that are important in the context of LCPs, where for a given matrix $M \in \mathbb{R}^{n\times n}$ and vector $\mathbf{q} \in \mathbb{R}^n$, we search for vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, such that

$$-M\mathbf{u} + \mathbf{v} = \mathbf{q}, \quad \mathbf{u}, \mathbf{v} \geq \mathbf{0}, \quad \mathbf{u} \circ \mathbf{v} = \mathbf{0}.$$

Since we do not require symmetry in the definition of any matrix class, the definition of positive (semi)definite matrices differs from the one known from linear algebra; therefore, these are also stated here:

Definition 2.1 A matrix $M \in \mathbb{R}^{n\times n}$ belongs to the class of *positive definite matrices* (PD), if $\mathbf{x}^T M\mathbf{x} > 0$ holds for all $\mathbf{x} \in \mathbb{R}^n \backslash \{\mathbf{0}\}$. Likewise, $M \in \mathbb{R}^{n\times n}$ belongs to the class of *positive semidefinite matrices* (PSD) if $\mathbf{x}^T M\mathbf{x} \geq 0$ holds for all $\mathbf{x} \in \mathbb{R}^n$.

Furthermore, $M \in \mathbb{R}^{n\times n}$ is a *skew-symmetric matrix* (SS), if $\mathbf{x}^T M\mathbf{x} = 0$ for all $\mathbf{x} \in \mathbb{R}^n$.

Clearly, the decision problem of whether the matrix is skew-symmetric is solvable in polynomial time. For matrices with integer entries, checking whether a matrix is PD or PSD belongs to the set of polynomially solvable decision problems (via Cholesky or LU factorization, see [45]).

Definition 2.2 A matrix $M \in \mathbb{R}^{n\times n}$ is a $\mathcal{P}$-matrix ($\mathcal{P}_0$-matrix) if all its principal minors are positive (nonnegative).

The class of $\mathcal{P}$- and $\mathcal{P}_0$-matrices were introduced and studied by Friedler and Pták [19, 20]. The importance of the $\mathcal{P}$-matrices in the theory of LCPs was observed in the 1960s, and the following result was proved:

Theorem 2.1 *Let $M \in \mathbb{R}^{n\times n}$ be given. The LCP has a unique solution for each $\mathbf{q} \in \mathbb{R}^n$ if and only if M is a $\mathcal{P}$-matrix.*

For details, see Cottle, Pang and Stone [4, 3.3.7 Theorem], or Murty [46, 3.15 Theorem]. Later in 1991, Kojima et al. [40, Lemma 4.1.] proved that the Newton-system occurring in each iteration of an IPA for solving LCPs has a unique solution if and only if M is a $\mathcal{P}_0$-matrix due to the fact that the matrix of the Newton-system is regular if and only if M is a $\mathcal{P}_0$-matrix. Tseng [56] showed that the complexity of the decision problem of whether a given matrix is $\mathcal{P}$ or $\mathcal{P}_0$, among several other matrix classes, is co-NP-complete.

Cottle, Pang, and Venkateswaran in their paper [5] defined new matrix classes related to LCPs.

Definition 2.3 A matrix $M \in \mathbb{R}^{n\times n}$ is called a column sufficient matrix (CS) if for all $\mathbf{x} \in \mathbb{R}^n$

$$X(M\mathbf{x}) \leq 0 \ \text{ implies } \ X(M\mathbf{x}) = 0,$$

and row sufficient (RS) if M^T is column sufficient. The matrix M is sufficient (SU) if it is both row and column sufficient.

We call a nonzero vector $\mathbf{u}$ a *certificate* for the non column (row) sufficiency of the matrix M if the vector $UM\mathbf{u}$ ($UM^T\mathbf{u}$) is nonpositive but not the zero vector. (Recall that $U = \text{diag}(\mathbf{u})$.)

Cottle et al. [5, Theorem 6] showed that for every vector $\mathbf{q} \in \mathbb{R}^n$ and matrix $M \in \mathbb{R}^{n\times n}$, a feasible LCP has convex solution set if and only if the matrix $M \in CS$. On the other hand, the solution set of a feasible LCP is identical to the set of the Karush–Kuhn–Tucker points of the natural quadratic program associated with the given LCP if and only if the matrix $M \in RS$ [5, Theorem 4]. Furthermore, as we have already mentioned in Introduction, the SU is the widest class of matrices where the finiteness of the CCA for LCPs can be proved, see for details Hertog et al. [14] and Csizmadia et al. [7, 8]. After these positive and valuable results, let us finish with a negative one. Tseng [56] proved that the decision problem of whether a matrix is column sufficient is co-NP-complete, as was expected from the previously stated results of Tseng.

The $\mathcal{P}_*(\kappa)$-matrices were introduced by Kojima et al. [40] as the matrix class where the polynomiality of IPAs for solving LCPs can be assured, however, the complexity depends also on κ. On the other hand, this matrix class can also be considered as a generalization of PSD matrices.

Definition 2.4 Let κ be a nonnegative number. A matrix $M \in \mathbb{R}^{n\times n}$ is a $\mathcal{P}_*(\kappa)$-matrix if for all $\mathbf{x} \in \mathbb{R}^n$

$$(1+4\kappa)\sum_{i\in\mathcal{I}_+(\mathbf{x})} x_i(M\mathbf{x})_i + \sum_{i\in\mathcal{I}_-(\mathbf{x})} x_i(M\mathbf{x})_i \geq 0, \tag{1}$$

where $\mathcal{I}_+(\mathbf{x}) = \{1 \leq i \leq n : x_i(M\mathbf{x})_i > 0\}$ and $\mathcal{I}_-(\mathbf{x}) = \{1 \leq i \leq n : x_i(M\mathbf{x})_i < 0\}$.

Therefore, naturally, $\mathcal{P}_*(0)$ is the class of PSD matrices.

The smallest $\kappa \geq 0$ for which M is $\mathcal{P}_*(\kappa)$ is called the *handicap* of M, and is denoted by $\widehat{\kappa}(M)$. Matrices with finite handicap are known as $\mathcal{P}_*$-matrices.

For the complete picture, it is worth mentioning some further related results for IPAs. First of all, the central path that has been introduced for the linear programming (LP) problem independently by Sonnevend [53] and Megiddo [42] plays an important role in defining IPAs. The existence and uniqueness of the central path for LCPs depends on the data of the problem. In case when M is SS-, PSD- or bisymmetric matrix (see Klafszky and Terlaky [39]), the existence and uniqueness of the central path was already known. Kojima et al. [40], based on some deeper results and under the assumption of the existence of a strictly feasible solution (interior point), proved the existence and the uniqueness of the central path for LCPs with $\mathcal{P}_*(\kappa)$-matrices. Illés at al. (1998) in a manuscript [33] gave an elementary proof of the existence and the uniqueness of the central path for LCPs with $\mathcal{P}_*(\kappa)$-matrices. Nagy (2009) in her PhD thesis (see [47, Chapter 3]) used the proof of Illés et al. to show some properties of the central path for sufficient LCPs. Furthermore, Nagy introduced an LCP [47, Problem (7.1)] such that the feasible set and the solution set are nonempty, but there is no unique central path. Naturally, the matrix of this LCP is not sufficient, and not even a $\mathcal{P}_0$-matrix.

IPAs for sufficient LCPs, similarly to IPAs for LPs, produce an ε-optimal solution. Even for LP problems, many operations research experts who are not specialists in linear optimization thought that with IPAs only ε-optimal solution could be produced

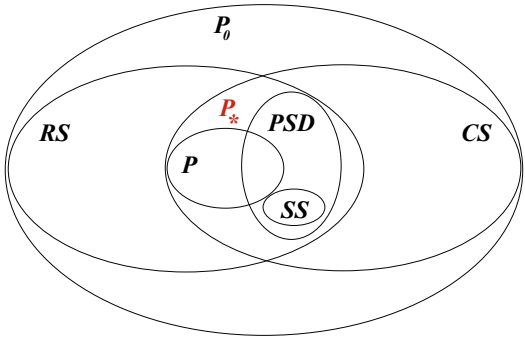

Fig. 1 The inclusion relations between matrix classes; CS = column sufficient, RS = row sufficient, SS = skew-symmetric, PSD = positive semidefinite

for LP problems, not exact, optimal solutions. Although the *rounding procedure* has been discussed in the well-known book of Roos et al. [51], still many thought that an exact solution could not be produced by IPAs for LP problems. The paper of Illés and Terlaky [34] helped a lot to dispel this misunderstanding. Therefore, it is not very surprising that they participated in the delivery of a theoretically efficient rounding procedure [32] for IPAs solving LCPs with $\mathcal{P}_*(\kappa)$-matrices. However, de Klerk and E.-Nagy [13] pointed out correctly that since the rounding procedure's performance strongly depends on the parameter κ, for sufficient matrices with large handicap values, the rounding procedure practically will be inefficient; thus, finding an exact solution for LCPs with $\mathcal{P}_*(\kappa)$-matrices, in case of large κ, from a practical point of view, is still a challenging question.

Note that $M \in CS$ if and only if the index set $\mathcal{I}_+(\mathbf{x})$ is nonempty for all $\mathbf{x}$ for which the index set $\mathcal{I}_-(\mathbf{x})$ is nonempty. In a sequence of papers [26, 40, 58] it was proven that the class of $\mathcal{P}_*$-matrices is equal to the class of sufficient matrices.

Figure 1 summarizes the relations among the previously discussed matrix classes [40].

According to the definitions, it is easy to see that the sets of PD, PSD, $\mathcal{P}$, $\mathcal{P}_*$, and $\mathcal{P}_0$-matrices are cones. But while the PSD and SS matrix classes are convex, the sets of $\mathcal{P}$, $\mathcal{P}_*$ and $\mathcal{P}_0$-matrices are not convex cones.[1]

2.1 Classification and Testing Sufficient Matrices

Väliaho [58] gave a full characterization of 2×2 matrices. Let $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, where $a, b, c, d \in \mathbb{R}$, then

$$\begin{aligned} M \in \mathcal{P} \quad &\Leftrightarrow \quad a > 0,\ d > 0,\ ad - bc > 0, \\ M \in \mathrm{PSD} \quad &\Leftrightarrow \quad a \geq 0,\ d \geq 0,\ (b + c)^2 \leq 4ad, \\ M \in \mathcal{P}_* \quad &\Leftrightarrow \quad a \geq 0,\ d \geq 0,\ (ad - bc > 0 \vee \\ &\qquad \vee\ (ad - bc = 0 \wedge ((a = 0 \vee d = 0) \Rightarrow b = 0, c = 0))). \end{aligned} \tag{2}$$

[1] Illés and Wenzel [37] gave a $\mathcal{P}$-matrix, $\left(\begin{smallmatrix} 1 & 4 \\ -1 & 1 \end{smallmatrix}\right)$, whose sum with its transpose has a negative determinant, so the sum is not even a $\mathcal{P}_0$-matrix.

No similar classification of sufficient matrices of larger size is known. Furthermore, non-trivial subclasses of $\mathcal{P}_*\backslash\mathcal{P}$ are not known. The handicap of 2×2-matrices M, $\widehat{\kappa}(M)$ can be computed in the following way (for details, see [59, Theorem 4.1])

$$\begin{aligned} M \in \text{PSD} \quad &\Rightarrow \quad \widehat{\kappa} = 0 \quad \text{(by definition)}, \\ M \in \mathcal{P}_*\backslash\text{PSD} \quad &\Rightarrow \quad 1 + 4\widehat{\kappa} = \frac{\max\{b^2, c^2\}}{(\sqrt{ad} + \sqrt{ad - bc})^2}, \\ M \in \mathcal{P}_* \setminus (\mathcal{P} \cup \text{PSD}) \quad &\Rightarrow \quad 1 + 4\widehat{\kappa} = \max\left\{\left|\frac{b}{c}\right|, \left|\frac{c}{b}\right|\right\}. \end{aligned}$$

Väliaho proved that the handicap is a continuous function of the elements for 2 by 2 sufficient matrices [59, Remark 4.2]. Furthermore, for any $M \in \mathcal{P}$, $\widehat{\kappa}(M)$ is a continuous function of the elements of M [59, Remark 2.1]. However, it is an open question whether the handicap $\widehat{\kappa}$ is a continuous function of elements of M when $M \in \mathcal{P}_* \setminus \mathcal{P}$ [59, Remark 6.1].

The inequality (1) gives the following lower bound on $\widehat{\kappa}(M)$ for any[2] matrix M:

$$\widehat{\kappa}(M) \geq \kappa_M(\mathbf{x}) := \begin{cases} 0 & \text{if } \mathbf{x}^T M\mathbf{x} \geq 0, \\ \dfrac{1}{4}\dfrac{-\mathbf{x}^T M\mathbf{x}}{\sum_{i\in\mathcal{I}_+} x_i(Mx)_i} & \text{if } \mathbf{x}^T M\mathbf{x} < 0 \text{ and } \mathcal{I}_+(\mathbf{x}) \neq \emptyset, \\ \infty & \text{otherwise.} \end{cases}$$

Furthermore,

$$\widehat{\kappa}(M) = \sup_{\mathbf{x}\in\mathbb{R}^n} \kappa_M(\mathbf{x}) = \sup_{\|\mathbf{x}\|_p \leq 1} \kappa_M(\mathbf{x}) = \sup_{\|\mathbf{x}\|_p = 1} \kappa_M(\mathbf{x}),$$

where $\|.\|_p$ is the p-norm. So it is enough to take the supremum on the surface of a unit sphere. Therefore, we call a nonzero vector $\mathbf{u}$ for a certificate for the fact $M \notin \mathcal{P}_*(\bar{\kappa})$, if $\kappa_M(\mathbf{u}) > \bar{\kappa}$.

Note that the function $\kappa_M(\mathbf{x})$ is not continuous even for 2 by 2 sufficient matrices.

As we have already mentioned, Väliaho developed two tests. One to decide whether a matrix is sufficient [58] and another to determine the handicap value of sufficient matrices [59]. Unfortunately, both methods are exponential, and there is no known polynomial algorithm for these problems. Tseng proved that the decision problem of whether a matrix is column (row) sufficient is co-NP-complete [56], therefore, a polynomial algorithm is not expected for deciding the sufficiency of a matrix. It is an open question whether there is a polynomial time algorithm to compute the handicap of a sufficient matrix.

All matrix classes discussed earlier, namely SS, PSD, $\mathcal{P}$, $\mathcal{P}_*(\kappa)$, $\mathcal{P}_*$, and $\mathcal{P}_0$, enjoy the nice property that if a matrix belongs to one of these classes, then any principal submatrix of the matrix and any principal pivotal transformation of it does, as well [4].

[2] We can extend the concept of the handicap for arbitrary matrices: let $\widehat{\kappa}(M)$ be infinity if $M \notin \mathcal{P}_*$, namely it is not sufficient.

A *principal pivotal transformation* (PPT) of the matrix $A = \begin{pmatrix} A_{\mathcal{J}\mathcal{J}} & A_{\mathcal{J}\mathcal{K}} \\ A_{\mathcal{K}\mathcal{J}} & A_{\mathcal{K}\mathcal{K}} \end{pmatrix}$ (where $\mathcal{J} \cup \mathcal{K} = \mathcal{I}$ and $\mathcal{J} \cap \mathcal{K} = \emptyset$) for nonsingular $A_{\mathcal{J}\mathcal{J}}$ is the matrix

$$\mathcal{P}_{\mathcal{J}}(A) = \begin{pmatrix} A_{\mathcal{J}\mathcal{J}}^{-1} & -A_{\mathcal{J}\mathcal{J}}^{-1} A_{\mathcal{J}\mathcal{K}} \\ A_{\mathcal{K}\mathcal{J}} A_{\mathcal{J}\mathcal{J}}^{-1} & A_{\mathcal{K}\mathcal{K}} - A_{\mathcal{K}\mathcal{J}} A_{\mathcal{J}\mathcal{J}}^{-1} A_{\mathcal{J}\mathcal{K}} \end{pmatrix}.$$

3 Properties and Construction of Sufficient Matrices

In this section, we first collect some known results, which give basic tools to generate sufficient matrices. Then, we prove some further properties of this matrix class which can help to build matrices belonging to this class. Finally, we combine these basic steps to show a few more complex ways to construct sufficient matrices. Some of the results were already published in the paper [27], but mostly without proofs.

Basic steps to construct a sufficient matrix:

1. **(Submatrix)** [5] *If* $M \in \mathcal{P}_*$, *then its every principal submatrix is also sufficient, namely* $M_{\mathcal{R}\mathcal{R}} \in \mathcal{P}_*$*for all* $\mathcal{R} \subseteq \mathcal{I}$.
2. **(Principal rearrangement)** [3] *If* $M \in \mathcal{P}_*$, *then for any permutation matrix* P *(in the same size as* M*), the rearranged matrix* $P^T M P \in \mathcal{P}_*$.
3. **(PPT)** [3] *If* $M \in \mathcal{P}_*$, *then any principal pivotal transformation of* M*is also sufficient, namely* $\mathcal{P}_{\mathcal{J}}(M) \in \mathcal{P}_*$ *for all* $\mathcal{J} \subseteq \mathcal{I}$.
4. **(Rank one matrix)** [57] *A rank one matrix* A *is sufficient if and only if it has a nonnegative diagonal and if* $A_{ii} = 0$*then the* i^{th} *row and column of the matrix* A *are all zero. In other words,* $A = \mathbf{u}\mathbf{v}^T \in \mathcal{P}_*$ $(\mathbf{u}, \mathbf{v} \in \mathbb{R}^n)$*if and only if* $u_i v_i > 0$ *or* $u_i = v_i = 0$*for all* i.
5. **(Scaling)** [40] *If* $M \in \mathcal{P}_*$, *then* $PMQ \in \mathcal{P}_*$ *for any diagonal matrices* P *and* Q, *where* $P_{ii} Q_{ii} > 0$ *for all indices* i. [3]
6. **(Block diagonal)** [59] *If* $M_1, M_2 \in \mathcal{P}_*$ *(possibly with different sizes), then* $\begin{pmatrix} M_1 & O \\ O & M_2 \end{pmatrix} \in \mathcal{P}_*$.
7. **(Blowing)** [59] *If* $M \in \mathcal{P}_*$, *then* $\begin{pmatrix} M & I \\ -I & D \end{pmatrix} \in \mathcal{P}_*$ *for any nonnegative diagonal matrix* D.
8. **(Shifting)** [59] *If* $M \in \mathcal{P}_*$, *then* $M + D \in \mathcal{P}_*$ *for any nonnegative diagonal matrix* D.
9. **(Duplication)** *If* $M \in \mathcal{P}_*$,*then* $\begin{pmatrix} M & M \\ M & M \end{pmatrix} \in \mathcal{P}_*$. Indeed, if the vector $(\mathbf{x}; \mathbf{y})$ certifies that the latter block matrix is not sufficient, then the vector $\mathbf{x} + \mathbf{y}$ proves that the matrix M cannot be sufficient by definition.

The next construction possibility is a combination of steps 9 and 1:

[3] Moreover, if $M \in \mathcal{P}_*(\kappa)$, then $PMQ \in \mathcal{P}_*(\kappa')$, where $(1+4\kappa') \min_i P_{ii}/Q_{ii} = (1+4\kappa) \max_i P_{ii}/Q_{ii}$.

10. **(Row-column duplication)** *If* $M \in \mathcal{P}_*$,*then* $\begin{pmatrix} M & \mathbf{m}_i \\ \mathbf{m}^{(i)} & m_{ii} \end{pmatrix} \in \mathcal{P}_*$ *for any row-column pairs, where* $\mathbf{m}_i$ and $\mathbf{m}^{(i)}$ *are the* i^{th}*column and row of the matrix* M,*respectively.*
11. **(Sign reverse)** *If the block matrix* $\begin{pmatrix} M_1 & M_2 \\ M_3 & M_4 \end{pmatrix} \in \mathcal{P}_*$,*then* $\begin{pmatrix} M_1 & -M_2 \\ -M_3 & M_4 \end{pmatrix} \in \mathcal{P}_*$.
The proof of the last implication again can be done by definition. Assume that the second matrix is not sufficient and $(\mathbf{x}; \mathbf{y})$ is a certificate. Then, $(\mathbf{x}; -\mathbf{y})$ verifies that the initial matrix is not sufficient.

Now let $M \in \mathbb{R}^{n\times n}$ be a sufficient matrix and use blowing (step 7) with $D = O$, then take the principal submatrix (step 1) for $\mathcal{R} = \{1, \ldots, n+1\}$. The resulting matrix $\begin{pmatrix} M & \mathbf{e}_1 \\ -\mathbf{e}_1^T & 0 \end{pmatrix}$ is sufficient. Using row-column duplication (step 10) on this matrix for the last index ($i = n + 1$) and then repeating it k times, we get the sufficient matrix $\begin{pmatrix} M & \mathbf{e}_1\mathbf{e}^T \\ -\mathbf{e}\mathbf{e}_1^T & O \end{pmatrix} \in \mathbb{R}^{(n+k)\times(n+k)}$, where $\mathbf{e} \in \mathbb{R}^k$ and $\mathbf{e}_1 \in \mathbb{R}^n$. If we choose not the first but the jth row and column of the identity matrix after the blowing step, then in the end, we get the following sufficient matrix: $\begin{pmatrix} M & \mathbf{e}_j\mathbf{e}^T \\ -\mathbf{e}\mathbf{e}_j^T & O \end{pmatrix} \in \mathbb{R}^{(n+k)\times(n+k)}$. Finally, we can rescale the matrix (step 5) and get the following result:

12. **(Composition)** *If* $M \in \mathcal{P}_*$ *and* $\mathbf{p}, \mathbf{q} \in \mathbb{R}^k$ such that $p_i q_i < 0$*for all indices* i,*then the matrix* $\begin{pmatrix} M & \mathbf{p}\mathbf{e}^T \\ \mathbf{e}\mathbf{q}^T & O \end{pmatrix} \in \mathbb{R}^{(n+k)\times(n+k)}$ *is sufficient.*
13. **(PSD gluing)** *Let* $A, B \in PSD$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$*and take the block diagonal matrix* $M = \mathrm{diag}(A, B)$. *Finally, let* $M_{n,n+1} = 1$ *and* $M_{n+1,n} = -1$. *Then* $M \in PSD$, *so it is sufficient.*

Indeed, PSD gluing preserves the PSD property of the matrices since $M + M^T$ is a symmetric block diagonal matrix, whose blocks are symmetric PSD matrices. Therefore, M is also a PSD matrix by definition.

Notice that if we glue two all-one matrices in the previously described way and apply scaling (step 5), then actually we glue two rank-one sufficient matrices with suitable values. More precisely

14. **(Rank-1 gluing)** *Let* $M = \mathrm{diag}(\mathbf{p}\mathbf{q}^T, \mathbf{r}\mathbf{s}^T)$, *where* $\mathbf{p}, \mathbf{q} \in \mathbb{R}^n$, $\mathbf{r}, \mathbf{s} \in \mathbb{R}^m$, $p_i q_i$ and $r_j s_j$ *are positive (or maybe* $p_i = q_i = 0$, $r_j = s_j = 0$*)for all* i *and* j, *and then modify two entries of the matrix* M: $M_{n,n+1} = p_n s_1$ *and* $M_{n+1,n} = -q_n r_1$.*Then* $M \in \mathcal{P}_*$.

Of course, we can make several other combinations of the aforementioned steps, for example, using step 12 and then shifting (step 8), we can put a nonnegative diagonal matrix in the low right corner instead of the all-zero matrix.

Based on some of these construction steps, Illés and Morapitiye [27] generated and collected several sufficient matrices [43]. However, there are some disadvantages of these matrices:

– During the constructions of these matrices, several times the principal pivotal transformation has been applied. The principal pivotal transformation is a numerically

unstable transformation due to the finite digit representation of the computed numbers, therefore, the resulting matrix is inaccurate and the sufficient matrix property may be lost.
– The constructions used are not suitable for paying more attention to the size of the handicap, so the $\widehat{\kappa}(M)$ of the constructed matrices is not large.

All these disadvantages still keep open the question of how we could construct medium and large size sufficient matrices.

Sufficient matrices of smaller size have been constructed in a different way. On the website [16], the reader can find the collection of these matrices, which are used in the computational part of our paper.

3.1 Lower Triangular Sufficient Matrices

It is a straightforward consequence of the definitions that a lower triangular matrix is in $\mathcal{P}$ ($\mathcal{P}_0$) if and only if its diagonal elements are positive (nonnegative). Indeed, all principal submatrices of a lower triangular matrix are also lower triangular matrices and the determinant of such a matrix is the product of the diagonal elements.

Furthermore, we can give a simple characterization of lower triangular sufficient matrices as well. Since a sufficient matrix is also in $\mathcal{P}_0$, all of the diagonal elements should be nonnegative. On the other hand, $\mathcal{P}$ is a subset of the sufficient matrix class, thus if all of the diagonal elements are positive, then the matrix is sufficient. Namely, we only need to investigate the case of the zero diagonal elements. For this, let us recall the following result of Cottle [3]. We present it in a similar way as Väliaho [58, Thm. 2.1].

Lemma 3.1 *Let $M \in \mathbb{R}^{n\times n}$ be a sufficient matrix with $m_{kk} = 0$ for some $k \in \{1, \ldots, n\}$. Then $m_{ik} = m_{ki} = 0$ or $m_{ik}m_{ki} < 0$ for all $i \neq k$.*

Based on Lemma 3.1, it follows easily that if a diagonal element is zero in a lower triangular sufficient matrix, then the corresponding row and column should be all zero. Summarizing the observations above:

Corollary 3.1 *A lower triangular matrix is sufficient if and only if its diagonal elements are nonnegative, and if a diagonal element is zero, then all the elements of the corresponding column and row are zero.*

Using the very strong structure of lower triangular matrices, the following constructions lead to new larger size $\mathcal{P}$- and $\mathcal{P}_*$-matrices.

15. **(Block structure)** *Let $M_1, M_2 \in \mathcal{P}$ (possibly with different sizes) be lower triangular matrices, then $\widehat{M} = \begin{pmatrix} M_1 & O \\ A & M_2 \end{pmatrix} \in \mathcal{P}$,where A is an arbitrary matrix with proper size. Therefore, $\widehat{M} \in \mathcal{P}_*$as well.*

3.2 Csizmadia's Matrix

In this section, we investigate a special lower triangular sufficient matrix, which was introduced in [13].

A natural question is how big the handicap of a matrix can be. It is easy to construct a sequence of matrices where an element of the matrix tends to infinity, and therefore, the handicap of the matrices tends also to infinity. But it is not trivial anymore what we can say if the encoding size of the matrix is bounded. In other words, is there a matrix whose handicap is exponential in its bit size? The answer was given by De Klerk and E.-Nagy [13].

Zsolt Csizmadia suggested the following matrix as an example where the components are small, but after a suitable principal pivotal transformation, the matrix has components that are exponential in size of the matrix:

$$C_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ -1 & -1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & 1 \end{pmatrix} \quad \text{and} \quad \mathcal{P}_{\{1,2,\ldots,n-1\}} C_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 2 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -2^{n-2} & -2^{n-3} & -2^{n-4} & \cdots & 1 \end{pmatrix}$$

This is a $\mathcal{P}$-matrix, as we discussed at the beginning of this section, so it is a sufficient matrix; namely, it has a finite handicap. Furthermore, the handicap is invariant under principal pivotal transformation; therefore, this matrix is a good candidate for a matrix with 'big' handicap. To that end, De Klerk and E.-Nagy [13] considered this matrix and proved that the handicap $\widehat{\kappa}$ of Csizmadia's matrix is exponential in the size n and the bit length L. They proved that $\widehat{\kappa}(C_n) \geq 2^{2n-8} - 0.25$.

We now give a constructive proof for this lower bound; moreover, we show that this is the exact value of the handicap. For the proof, we need some observations about parabolas:

Lemma 3.2 *Let a and b be two positive numbers, and let d be an arbitrary number. Consider the following two minimization problems*

$$\left.\begin{aligned} \min_z \; & az(z-d) - b(z+d)^2 \\ & z(z-d) \leq 0 \end{aligned}\right\} \; (P_{\leq}) \qquad \left.\begin{aligned} \min_z \; & az(z-d) - b(z+d)^2 \\ & z(z-d) \geq 0 \end{aligned}\right\} \; (P_{\geq})$$

1. *If $4b \leq a$, then $\frac{a+2b}{2(a-b)}d$ is an optimal solution of $(P_{\leq})$ and d is an optimal solution of $(P_{\geq})$.*
2. *If $b < a < 4b$, then d is an optimal solution of $(P_{\leq})$ and $\frac{a+2b}{2(a-b)}d$ is an optimal solution of $(P_{\geq})$.*
3. *If $a \leq b$, then d is an optimal solution of $(P_{\leq})$ and the problem $(P_{\geq})$ is unbounded or trivial (the case $a = b$ and $d = 0$).*

Proof The objective function of both problems is the same:

$$f(z) = az(z-d) - b(z+d)^2 = (a-b)z^2 - (a+2b)dz - bd^2.$$

This is a quadratic function of z; more precisely, it is a convex parabola if $a > b$, and a concave parabola if $a < b$.

Let us first consider the case when $a \geq 4b$, so f is convex. Then, the minimum point of this parabola is $\hat{z} = \frac{a+2b}{2(a-b)}d$. Since $\frac{1}{2} < \frac{a+2b}{2(a-b)} \leq 1$, $\hat{z}$ is feasible for the problem $(P_{\leq})$ and it is closer to d than to 0, so d is an optimal solution of $(P_{\geq})$.

In the second case, namely if $b < a < 4b$, the function f is still convex, and its minimum point is the same $\hat{z}$. However, now $\hat{z}$ is not feasible for the problem $(P_{\leq})$, but it is feasible for $(P_{\geq})$. Indeed, in this case $\frac{a+2b}{2(a-b)} > 1$, so the optimal solution of $(P_{\leq})$ is d and the optimal solution of $(P_{\geq})$ is $\hat{z}$.

When $a \leq b$, the function f is concave, so the problem $(P_{\geq})$ is unbounded except in the case when $a = b$ and $d = 0$. On the other hand, the minimum point of the problem $(P_{\leq})$ will be one of the endpoints of the feasibility interval. Since $-bd^2 = f(0) \geq f(d) = -4bd^2$, the optimal solution of $(P_{\leq})$ is d. □

Lemma 3.3 *Csizmadia's matrix is positive semidefinite for $n \leq 3$, but it is not positive semidefinite for $n \geq 4$. Furthermore,*

$$\widehat{\kappa}(C_n) = \begin{cases} 0 & \text{if } n \leq 3 \\ 2^{2n-8} - 0.25 & \text{if } n \geq 4. \end{cases}$$

Proof The positive semidefiniteness of the matrix in small dimensions can be shown easily. Now we assume that $n \geq 4$. Let the vector $\tilde{\mathbf{x}}$ be the following

$$\tilde{x}_1 = 1, \quad \tilde{x}_i = 2^{i-2} \text{ for } i = 2, 3, \ldots n-1, \quad \tilde{x}_n = 2^{n-3}.$$

Then for $\tilde{\mathbf{y}} = C_n\tilde{\mathbf{x}}$, we have

$$\tilde{y}_1 = 1, \quad \tilde{y}_i = 0 \text{ for } i = 2, 3, \ldots, n-1, \quad \tilde{y}_n = -2^{n-3}.$$

Thus

$$\tilde{x}_1\,\tilde{y}_1 = 1, \qquad \tilde{x}_n\,\tilde{y}_n = -2^{2n-6},$$

and we have $\tilde{x}_i\,\tilde{y}_i = 0$ for all other indices i. Namely, $\kappa_{C_n}(\tilde{x}) = 2^{2n-8} - 0.25$, which gives a lower bound on the handicap.

In the second part of the proof, we show that $C_n \in \mathcal{P}_*(2^{2n-8} - 0.25)$, so this value is also an upper bound on the handicap. We demonstrate it by definition, we verify the nonnegativity of the function $h(\mathbf{x}) = 2^{2n-6}\sum_{i\in\mathcal{I}_+(\mathbf{x})} x_i(C_n\mathbf{x})_i + \sum_{i\in\mathcal{I}_-(\mathbf{x})} x_i(C_n\mathbf{x})_i$ for all $x \in \mathbb{R}^n$. More precisely, we show that all of the minimizers of h have the form $\alpha\tilde{\mathbf{x}}$, where α is a nonzero number.

The function h is a weighted sum of the components of the vector

$$\mathbf{x} \circ C_n\mathbf{x} = \begin{pmatrix} x_1^2 \\ x_2(x_2 - x_1) \\ \vdots \\ x_n(x_n - \sum_{i=1}^{n-1} x_i) \end{pmatrix}.$$

Notice that only the last coordinate of this vector depends on x_n. Therefore, $\bar{\mathbf{x}}$ minimizes h if and only if $\bar{x}_n = -1/2 \sum_{i=1}^{n-1} \bar{x}_i$. Thus,

$$\bar{\mathbf{x}} \circ C_n \bar{\mathbf{x}} = \begin{pmatrix} \bar{x}_1^2 \\ \bar{x}_2(\bar{x}_2 - \bar{x}_1) \\ \vdots \\ \bar{x}_{n-1}(\bar{x}_{n-1} - \sum_{i=1}^{n-2} \bar{x}_i) \\ -1/4 \left(\sum_{i=1}^{n-1} \bar{x}_i \right)^2 \end{pmatrix}.$$

The latter vector has only two coordinates which depend on x_{n-1}; therefore, we should solve the following two minimization problems according to the sign of the $(n-1)^{th}$ coordinate (the last one is always nonpositive):

$$\left.\begin{aligned} \min_{x_{n-1}} \; & x_{n-1} \left(x_{n-1} - \sum_{i=1}^{n-2} x_i \right) - 1/4 \left(\sum_{i=1}^{n-1} x_i \right)^2 \\ & x_{n-1}(x_{n-1} - \textstyle\sum_{i=1}^{n-2} x_i) \leq 0 \end{aligned}\right\} (P_{\leq}^{n-1})$$

$$\left.\begin{aligned} \min_{x_{n-1}} \; & 2^{2n-6} x_{n-1} \left(x_{n-1} - \sum_{i=1}^{n-2} x_i \right) - 1/4 \left(\sum_{i=1}^{n-1} x_i \right)^2 \\ & x_{n-1}(x_{n-1} - \textstyle\sum_{i=1}^{n-2} x_i) \geq 0 \end{aligned}\right\} (P_{\geq}^{n-1})$$

Based on Lemma 3.2 for problems $(P_{\leq}^{n-1})$ and $(P_{\geq}^{n-1})$ with $z = x_{n-1}$ and $d = \sum_{i=1}^{n-2} x_i$, the optimal solution is $z = d$ in both cases, namely $\bar{x}_{n-1} = \sum_{i=1}^{n-2} \bar{x}_i$ and

$$\bar{\mathbf{x}} \circ C_n \bar{\mathbf{x}} = \begin{pmatrix} \bar{x}_1^2 \\ \bar{x}_2(\bar{x}_2 - \bar{x}_1) \\ \vdots \\ \bar{x}_{n-2}(\bar{x}_{n-2} - \sum_{i=1}^{n-3} \bar{x}_i) \\ 0 \\ -\left(\sum_{i=1}^{n-2} \bar{x}_i \right)^2 \end{pmatrix}.$$

Now only $n-2$ variables remain and again the largest index $(n-2)$ appears only in two coordinates (n^{th} and $(n-2)^{th}$) of the vector $\bar{\mathbf{x}} \circ C_n \bar{\mathbf{x}}$. Therefore, we show by induction that

$$\bar{\mathbf{x}} \circ C_n \bar{\mathbf{x}} = \begin{pmatrix} \bar{x}_1^2 \\ \bar{x}_2(\bar{x}_2 - \bar{x}_1) \\ \vdots \\ \bar{x}_{n-j}(\bar{x}_{n-j} - \sum_{i=1}^{n-j-1} \bar{x}_i) \\ 0 \\ \vdots \\ 0 \\ -2^{2(j-2)} \left(\sum_{i=1}^{n-j} \bar{x}_i \right)^2 \end{pmatrix}$$

for $2 \le j \le n-1$. Above we proved this statement for $j = 2$. Now we assume that it is true for a bigger index j ($j \le n-2$) and show that it holds for $j+1$ as well.

The vector $\bar{\mathbf{x}} \circ C_n \bar{\mathbf{x}}$ depends on x_{n-j} only in the coordinates n and $(n-j)$. So to solve the minimization problem of h in the variable x_{n-j}, we should solve the following two problems:

$$\left.\begin{array}{c} \min\limits_{x_{n-j}} x_{n-j} \left(x_{n-j} - \sum\limits_{i=1}^{n-j-1} x_i \right) - 2^{2(j-1)} \left(\sum\limits_{i=1}^{n-j} x_i \right)^2 \\ x_{n-j}(x_{n-j} - \sum_{i=1}^{n-j-1} x_i) \le 0 \end{array}\right\} (P_{\le}^{n-j})$$

$$\left.\begin{array}{c} \min\limits_{x_{n-j}} 2^{2n-6} x_{n-j} \left(x_{n-j} - \sum\limits_{i=1}^{n-j-1} x_i \right) - 2^{2(j-1)} \left(\sum\limits_{i=1}^{n-j} x_i \right)^2 \\ x_{n-j}(x_{n-j} - \sum_{i=1}^{n-j-1} x_i) \ge 0 \end{array}\right\} (P_{\ge}^{n-j}).$$

Using Lemma 3.2 for these problems with $z = x_{n-j}$ and $d = \sum_{i=1}^{n-j-1} x_i$, we again get that the optimal solution in both cases is $z = d$, namely $\bar{x}_{n-j} = \sum_{i=1}^{n-j-1} \bar{x}_i$ and $\bar{\mathbf{x}} \circ C_n \bar{\mathbf{x}}$ has the desired form. □

4 Testing Sufficiency

We have already seen in Introduction that the sufficient property of the coefficient matrix has a crucial role in analyzing algorithms for solving LCPs. This is why it would be important to find numerically efficient methods to test the sufficiency of arbitrary matrices. However, as we have already mentioned, Tseng proved that the decision problem of whether a matrix is column sufficient is co-NP-complete [56]; therefore, a polynomial algorithm cannot be expected.

In the rest of the paper, we investigate the following first three approaches and only mention the last possibility (remember, if the handicap of a matrix is finite, then the matrix is sufficient):

1. In 1996, Väliaho [58] proposed an exponential algorithm to examine the sufficiency of a matrix. We implemented his algorithm in MATLAB.
2. We propose a linear programming approach from [49], which turned out to be already a known result by Guu and Cottle [26]. However, our formulation is a bit different and simplified and we also implemented our method.
3. Nonlinear reformulations of the definition.
4. Väliaho also developed an exponential algorithm to determine the handicap of a sufficient matrix [59]. This is the only kind of such an algorithm in the literature. In 2011, de Klerk and E.-Nagy proposed an approximation hierarchy to compute the handicap [13].

4.1 Väliaho's Exponential Algorithm to Detect Matrix Sufficiency

Väliaho's algorithm is an iterative method. It checks the sufficiency of all of the principal submatrices with increasing size using the information that the previous ones are sufficient. Therefore, the core of the algorithm is the following procedure whose input is an n-by-n matrix which is sufficient in order $n-1$ (i.e. all of its principal submatrices of size $n-1$ are sufficient) and it decides whether the given matrix is sufficient or not.

We recall in Fig. 2 this core procedure ([57, Procedure 5.1]) keeping the original names S1-S5 of steps, but using our notations:

Now we are ready to give the full description of Väliaho's method, see Fig. 3.

Input: a matrix $A \in \mathbb{R}^{m\times m}$, A is $(m-1)$-sufficient.

S1: $B := A$, $k := m$, $\mathcal{K} := \{1,\ldots,m\}$.
S2: **If** $B_{\mathcal{K}\mathcal{K}}$ has at least one nonzero element go to S3.
If $k \geq 2$, **STOP**; A is sufficient.
If $k = 1$, **STOP**; let $\mathcal{K} = \{h\}$, A is sufficient if and only if $b_{ih} = 0, b_{hi} = 0$ or $b_{ih}b_{hi} < 0$ for all $i \in \{1,\ldots,n\}\backslash h$.
S3: **If** $k > 1$, go to S4. **If** $k = 1$, **STOP**; letting $\mathcal{K} = \{h\}$, A is sufficient if and only if $b_{hh} > 0$.
S4: **If** $B_{\mathcal{K}\mathcal{K}}$ has zero diagonal, go to S5. **Otherwise** choose an $i \in \mathcal{K}$ such that $b_{ii} > 0$, $B := \mathcal{P}_i(B)$, $\mathcal{K} := \mathcal{K} \setminus \{i\}$, $k := k-1$, and go to S2.
S5: **If** $k = 2$, **STOP**; A is sufficient if and only if $\det B_{\mathcal{K}\mathcal{K}} > 0$.
If $k > 2$, choose $(i,j) \in \mathcal{K}\times\mathcal{K}$ such that $b_{ij} \neq 0$, $B := \mathcal{P}_{i,j}B$, $\mathcal{K} := \mathcal{K} \setminus \{i,j\}$, $k := k-2$, and go to S2.

Fig. 2 `Valiaho_almost_SU(A)`, where $\mathcal{P}$ refers to the corresponding principal pivotal transformation, see on page 8

1. **Input:** a matrix $M \in \mathbb{R}^{n\times n}$.
2. **For** $k = 1\ldots n$
   For all $\mathcal{S} \subseteq \{1,\ldots,n\}$, $|\mathcal{S}| = k$
   If $\det M_{\mathcal{S}\mathcal{S}} < 0$, **STOP**: M is not sufficient.
   If $\det M_{\mathcal{S}\mathcal{S}} = 0$, **then** `Valiaho_almost_SU(`$M_{\mathcal{S}\mathcal{S}}$`)`
   If $M_{\mathcal{S}\mathcal{S}}$ is not sufficient, **STOP**: M is not sufficient.
3. **Output** M is sufficient.

Fig. 3 Väliaho's algorithm to detect sufficiency

4.2 Linear Programming Approach to Detect Matrix Sufficiency

In this subsection, we explain in detail the approach to detect matrix sufficiency based on solving linear programming feasibility problems.

Let $M \in \mathbb{R}^{n\times n}$ be a given matrix and $\mathcal{I} = \{1, 2, \ldots, n\}$ be the set of indices. Suppose that we partition the index set into three (disjoint) subsets, as follows: $\mathcal{I} = \mathcal{S}_{-1} \cup \mathcal{S}_0 \cup \mathcal{S}_1$, and $k = |\mathcal{S}_1 \cup \mathcal{S}_{-1}|$. There are 3^n such partitions of the index set. For each partition $\mathcal{S} = \mathcal{S}_{-1} \cup \mathcal{S}_0 \cup \mathcal{S}_1$, we define the *generalized orthant* $\mathbb{R}^n_{\mathcal{S}}$ in the following way:

$$\mathbb{R}^n_{\mathcal{S}} = \{\mathbf{x} \in \mathbb{R}^n : x_i > 0 \text{ for } i \in \mathcal{S}_1,\ x_i < 0 \text{ for } i \in \mathcal{S}_{-1},\ x_i = 0 \text{ for } i \in \mathcal{S}_0\}.$$

Let us define the matrix $\tilde{M}_{\mathcal{S}}$ corresponding to a given partition $\mathcal{S}$ as follows:

$$\left(\widetilde{M}^{\mathcal{S}}\right)_{ij} = \begin{cases} M_{ij} & \text{if } i, j \in \mathcal{S}_1 \text{ or } i, j \in \mathcal{S}_{-1}, \\ -M_{ij} & \text{if } (i \in \mathcal{S}_1 \text{ and } j \in \mathcal{S}_{-1}) \text{ or } (i \in \mathcal{S}_{-1} \text{ and } j \in \mathcal{S}_1), \\ 0 & \text{if } i \in \mathcal{S}_0 \text{ or } j \in \mathcal{S}_0. \end{cases}$$

The next step is to delete the zero rows and columns, so let us define the matrix $M^{\mathcal{S}}$ by deleting all rows and columns corresponding to the index set $\mathcal{S}_0$, namely $M^{\mathcal{S}} = \widetilde{M}_{\overline{\mathcal{S}}_0,\overline{\mathcal{S}}_0} \in \mathbb{R}^{k\times k}$, where $\overline{\mathcal{S}}_0 = \mathcal{I}\backslash\mathcal{S}_0$.

Theorem 4.1 *Let $M \in \mathbb{R}^{n\times n}$ be a given matrix and $\mathcal{I} = \{1, 2, \ldots, n\}$ be the set of indices. The following three statements are equivalent:*

1. *M is a column sufficient matrix.*
2. *There is no partition $\mathcal{S}$ of the index set $\mathcal{I}$ such that the system of linear inequalities*

$$\mathbf{x} \geq \mathbf{e}, \quad M^{\mathcal{S}}\mathbf{x} \leq \mathbf{0}, \quad \mathbf{e}^T M^{\mathcal{S}}\mathbf{x} \leq -1 \qquad (L_{\mathcal{S}})$$

   has a solution.
3. *For all partitions $\mathcal{S}$ of the index set $\mathcal{I}$ the system of linear inequalities*

$$\mathbf{y}^T M^{\mathcal{S}} \geq \mathbf{0}, \quad \mathbf{y} - \gamma\,\mathbf{e} \geq \mathbf{0}, \quad \gamma + \mathbf{y}^T M^{\mathcal{S}}\mathbf{e} = 1, \quad \gamma \geq 0 \qquad (D_{\mathcal{S}})$$

   has a solution.

Proof The matrix M is not column sufficient if and only if there is a vector $\bar{\mathbf{x}}$ such that the vector $\bar{X}M\bar{\mathbf{x}}$ is nonpositive but not the zero vector. If $\bar{\mathbf{x}} \in \mathbb{R}^n_{\mathcal{S}}$ then it means that $(M\bar{\mathbf{x}})_{\mathcal{S}_1} \leq \mathbf{0}$, $(-M\bar{\mathbf{x}})_{\mathcal{S}_{-1}} \leq \mathbf{0}$ and there is at least one nonzero coordinate. Namely,

$$M_{\mathcal{S}_1\mathcal{S}_1}\bar{\mathbf{x}}_{\mathcal{S}_1} - M_{\mathcal{S}_1\mathcal{S}_{-1}}(-\bar{\mathbf{x}}_{\mathcal{S}_{-1}}) \leq \mathbf{0} \quad \text{and} \quad -M_{\mathcal{S}_{-1}\mathcal{S}_1}\bar{\mathbf{x}}_{\mathcal{S}_1} + M_{\mathcal{S}_{-1}\mathcal{S}_{-1}}(-\bar{\mathbf{x}}_{\mathcal{S}_{-1}}) \leq \mathbf{0},$$

and there is at least one nonzero coordinate. Using the definition of the matrix $M^{\mathcal{S}}$ and introducing the positive vector $\mathbf{z} = (sgn(\bar{\mathbf{x}}) \circ \bar{\mathbf{x}})_{\overline{\mathcal{S}}_0}$, we get that $M^{\mathcal{S}}\mathbf{z} \leq \mathbf{0}$ and there is at least one nonzero coordinate.

Summarizing, M is not column sufficient if and only if there is a partition $\mathcal{S}$ and a vector $\mathbf{z}$ such that

$$\mathbf{z} > \mathbf{0}, \quad M^{\mathcal{S}}\mathbf{z} \leq \mathbf{0}, \quad M^{\mathcal{S}}\mathbf{z} \neq \mathbf{0},$$

equivalently

$$\mathbf{z} > \mathbf{0}, \quad M^{\mathcal{S}}\mathbf{z} \leq \mathbf{0}, \quad \mathbf{e}^T M^{\mathcal{S}}\mathbf{z} < 0. \tag{3}$$

Note that if $\mathbf{z}$ is a solution of system (3) then $\lambda\mathbf{z}$ is also a solution for all positive λ. Therefore, the system (3) is feasible if and only if the problem $(L_{\mathcal{S}})$ has a solution.

Finally, based on Farkas' lemma, system $(L_{\mathcal{S}})$ has no solution if and only if $(D_{\mathcal{S}})$ is solvable. □

Based on the result of Theorem 4.1, we can develop a simple, but computationally very demanding algorithm for testing if a given matrix is sufficient. It has to check whether $(L_{\mathcal{S}})$ is feasible for all possible partitions $\mathcal{I} = \mathcal{S}_{-1} \cup \mathcal{S}_0 \cup \mathcal{S}_1$ (there are 3^n of them). The infeasibility of all $(L_{\mathcal{S}})$ problems means that the given matrix M is column sufficient. Thus, if we find an $\hat{\mathcal{S}}$ partition of the index set $\mathcal{I}$ such that the corresponding $(L_{\hat{\mathcal{S}}})$ is feasible, then the given matrix M is not column sufficient, and the feasible solution $\mathbf{x}$ is a certificate for it.

The linear feasibility problems $(L_{\mathcal{S}})$ could be solved using state-of-the-art solvers like MOSEK [44], or FICO Xpress Optimizer [18].

Notice that this procedure inspects for each principal submatrix of M whether it is column sufficient or not. We can reduce significantly the number of LP problems to be solved based on Väliaho's following result [57, Theorem 4.2]:

Lemma 4.1 *If $M \in \mathbb{R}^{n\times n}$ is (row, column) sufficient of order $n-1$ and* det $M > 0$*, then M is (row, column) sufficient.*

It means that we need to solve the linear feasibility problem $(L_{\mathcal{S}})$ only if $\det M_{\bar{\mathcal{S}}_0,\bar{\mathcal{S}}_0} = 0$. Indeed, if the determinant is negative, then the matrix is not $\mathcal{P}_0$, namely, it is not sufficient. On the other hand, based on Lemma 4.1, if the determinant is positive, then the investigated partition $\mathcal{S}$ can give a certificate for non-column-sufficiency if and only if there is another partition $\mathcal{S}'$ such that $\mathcal{S}_0 \subset \mathcal{S}'_0$ and $\det M_{\bar{\mathcal{S}}'_0\bar{\mathcal{S}}'_0} < 0$ or $\det M_{\bar{\mathcal{S}}'_0\bar{\mathcal{S}}'_0} = 0$ and $(L_{\mathcal{S}'})$ has a solution.

The algorithm based on solving a sequence of linear feasibility problems of type $(L_{\mathcal{S}})$ is presented in Fig. 4.

The LP-B-Alg algorithm can be built in a slightly different way by using the system of linear inequalities presented in $(D_{\mathcal{S}})$. Numerical results obtained by LP-B-Alg method are presented in Sect. 5.5.

4.3 Nonlinear Programming Approach to Detect Matrix Sufficiency

Using the definition of column sufficiency, we introduce three different nonlinear programming models to test sufficiency using a nonlinear solver.

1. **Input:** matrix $M \in \mathbb{R}^{n\times n}$.
2. **For all** $\mathcal{S}_0 \subset \mathcal{I}$, where $|\mathcal{S}_0| = n-2$ apply Väliaho test (2).
   If $M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0}$ does not satisfy (2) **then** M is not column sufficient, **STOP**.
3. **For all** $k = n-3, n-4, \ldots, 0$,
   For all $|\mathcal{S}_0| = k$, compute $\det M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0}$.
   3.1 **If** $\det M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0} < 0$ **then** M is not column sufficient, **STOP**.
   3.2 **If** $\det M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0} = 0$ **then**
   For all $\mathcal{S}_{-1}, \mathcal{S}_1 \subset \mathcal{I}$ such that $\bar{\mathcal{S}}_0 = \mathcal{S}_{-1} \cup \mathcal{S}_1$ **solve** feasibility problem $(L_{\mathcal{S}})$.
   If $(L_{\mathcal{S}})$ has a solution, **then** M is not column sufficient, **STOP**.
4. **Output** M is column sufficient.

Fig. 4 LP-based algorithm (LP-B-Alg) to detecting column sufficiency

As we have already discussed, a vector $\mathbf{x}$ is a certificate for non-column-sufficiency of a matrix M if and only if $\mathbf{x} \circ M\mathbf{x} \lneq \mathbf{0}$, which is equivalent to

$$\mathbf{x} \circ M\mathbf{x} \leq \mathbf{0} \quad \text{and} \quad \mathbf{x}^T M\mathbf{x} < \mathbf{0}. \tag{4}$$

Since this is a homogeneous feasibility problem, we can add a norm constraint on $\mathbf{x}$. Moreover, the second constraint can be considered as the objective function of the problem. In this way, we get the first reformulation of the definition of column sufficiency:

1. The matrix M is column sufficient if and only if the optimal objective function value of the following problem is 0. (Note that the optimal value is nonpositive since the all-zero vector is always feasible.)

$$\min\ \mathbf{x}^T M\mathbf{x} \quad \text{s.t.} \qquad \mathbf{x} \circ (M\mathbf{x}) \leq \mathbf{0}, \quad \|\mathbf{x}\|_p \leq 1. \tag{5}$$

We get another approach if the homogeneous feasibility problem (4) is scaled in a different way. We can add a negative upper bound on the quadratic term $\mathbf{x}^T(M\mathbf{x})$ and search for a feasible solution where the Hadamard product $\mathbf{x}\circ(M\mathbf{x})$ gives a nonpositive vector:

2. The matrix M is column sufficient if and only if the following problem is either infeasible or its optimal value is positive.

$$\min\ t \quad \text{s.t.} \qquad \mathbf{x} \circ (M\mathbf{x}) \leq t\mathbf{e}, \quad \mathbf{x}^T M\mathbf{x} \leq -1. \tag{6}$$

In the case of the last mentioned possibility, to analyze the feasibility problem (4), we introduce a new variable $\mathbf{z} = \mathbf{x} \circ M\mathbf{x}$ and put these defining equations into the objective function with a proper penalty parameter λ:

3. The matrix M is column sufficient if and only if the optimal objective function value of the following problem is 0 for all positive λ. (Note that the optimal value is again nonpositive.)

$$\min \sum_{i=1}^{n} z_i + \lambda \sum_{i=1}^{n} (z_i - x_i(M\mathbf{x})_i)^2 \quad \text{s.t.} \qquad \mathbf{z} \leq \mathbf{0}. \tag{7}$$

5 Implementation Details and Numerical Results

In the previous sections, we proposed three different approaches to check the sufficiency of a matrix. We have performed preliminary numerical tests using these procedures. We implemented Väliaho's algorithm from Fig. 3 and the LP-B-Alg method from Fig. 4 in MATLAB (version 2019a), using the parallel computing toolbox. At the very beginning of both algorithms, we check if the matrix M is positive semidefinite by computing the smallest eigenvalue of $M + M^\top$. Since the dimensions of M are usually rather small in practical computations (less than 30×30), we compute this value by symbolic computations, i.e., we compute $\lambda_{\min} =$ `min(eig(sym(M+M')))`. If $\lambda_{\min} > -10^{-12}$, we consider the matrix M to be positive semidefinite, hence sufficient.

The NLP-based approach was implemented by the global nonlinear solver BARON. In the following subsections, we describe the implementation details, the data sets that we used and explain the numerical results.

5.1 Implementation Details about Väliaho's Algorithm

We made separate implementations for 2-by-2 and 3-by-3 matrices to increase efficiency. For higher dimensions, we coded exactly the recursive algorithm from Fig. 3, but we considered every number from the interval $(-10^{-8}, 10^{-8})$ as zero for numerical reasons. We note that Väliaho, in his paper [57], mentioned some further possibilities to improve the core procedure. It is future work to integrate these ideas into the code.

5.2 Implementation Details about the LP-Based Algorithm

In the sequel, we explain details about the implementation for testing the column sufficiency of a given square matrix M. The implementation for testing row sufficiency follows the same steps applied to $M^\top$.

The feasibility problems $(L_{\mathcal{S}})$ considered during the algorithm were solved by the MOSEK linear programming solver [44], which we run through MATLAB [55].

The LP-B-Alg algorithm has two types of stopping criteria: Steps 2, 3.1, and 3.2 (the matrix is not column sufficient) and Step 4 (the matrix is column sufficient). Step 4 could occur only after we check 3^n linear feasibility problems $(L_{\mathcal{S}})$, where $\mathcal{S}$ is a partition of the indices. We implemented it hierarchically, which means that the size of S_0 is decreasing from $n-2$ to 0 and for each size $|S_0| < n-2$, if $\det M_{\bar{S}_0\bar{S}_0}$ is zero, we perform step 3.2 for all possible sub-partitions $\mathcal{S}_{-1} \cup \mathcal{S}_1$.

Note that the computation of $\det M_{\bar{S}_0\bar{S}_0}$ is numerically very sensitive. However, it is a more critical error if we wrongly detect $\det M_{\bar{S}_0\bar{S}_0} < 0$, since this can imply the wrong conclusion that M is not sufficient. Therefore, we make this test in two steps: we first check if $\det M_{\bar{S}_0\bar{S}_0} < -\varepsilon$, where we set $\varepsilon = 10^{-12}$. If this is true, we again compute $\det M_{\bar{S}_0\bar{S}_0}$, but we use the symbolic version of $M_{\bar{S}_0\bar{S}_0}$, hence the determinant is computed with very high precision. Now, we repeat the test with the new value of the determinant and if it is again smaller than $-\varepsilon$, we conclude that M is not sufficient.

The test whether $\det M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0}$ is positive is performed by using the default script for calculating the determinant in MATLAB and verifying if $\det M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0} > \varepsilon$.

Therefore, we perform step 3.2 only when $\det M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0} \in [-\varepsilon, \varepsilon]$. The only remaining situation when the LP-B-Alg algorithm may fail is when the determinant of some $M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0}$ is actually zero, but our algorithm mistakenly numerically detects that the determinant is larger than ε and therefore skips step 3.2 and consequently fails to find a certificate for non-sufficiency.

The inner part of the algorithm, i.e., the for-loop in step 3.2, can be parallelized. We carried out the parallelization in MATLAB using a `parfor`-loop. The communication between the parallel processes performed by `parfor` is very limited. In particular, if one of the parallel processes finds a certificate that the matrix is not (column) sufficient, we have to accomplish the whole loop (check all possible partitions $\mathcal{S}_{-1} \cup \mathcal{S}_1$ for a given S_0) before we can exit. This takes a bit longer than necessary, e.g., if we have coded the algorithm using C and OpenMP or MPI, but overall, parallelization makes a difference and increases the reach of our algorithm.

5.3 Implementation Details about the NLP Algorithm

To compare the different model formulations, we implemented the models described in Sect. 4.3 in the AMPL modeling language [21] and used the BARON [52, 54] global nonlinear solver with precision 10^{-4} to find solutions. Note that for our purposes, it is very important to obtain the global optimal value of the NLP problems, since otherwise, we cannot decide the sufficiency of the matrix. Therefore, we cannot use a local solver.

After comparing the different model formulations in practice, we found that model (5) gave the best results and chose this model to carry out the numerical tests. We compared the results for different vector norms, and the best running times could be achieved by limiting the infinity norm of the variable vector. This restriction is necessary since BARON requires nonlinear expressions to be bounded in the model descriptions.

5.4 Data Sets

We tested the algorithms on three data sets. The first was constructed by E.-Nagy and is denoted by ENM [16], the second by Illés and Morapitiye [27] (denoted by IM), while the third consists of 7 Csizmadia's matrices (Csizmadia) of size $n = 10, 15, 20, 22, 25, 27, 30$, which are all sufficient, as described in Sect. 3.2 and in the references therein.

The first data set (ENM) contains

- 82 sufficient matrices of sizes $n = 3, 4, \ldots, 10$, there are 10 matrices for sizes $n = 3, \ldots, 8$ and 11 matrices for $n = 9, 10$;
- 80 non-sufficient (neither row nor column) matrices of sizes $n = 3, 4, \ldots, 10$, there are 10 matrices for each size.

Table 1 The number of matrices in the second data set (IM) for each size n

Size n	10	20	50	100	200	500	700
Number of matrices	**10 (1)**	**10 (2)**	10	10	10	10	1

In parentheses, we write how many matrices out of the total are non-sufficient

The ENM data set was constructed as follows: First random matrices with integer entries between -10 and 10 were generated. Then to get a sufficient matrix, an appropriately large integer multiple of the identity matrix was added to them. (In almost all cases, these are $\mathcal{P}$ matrices.) Finally, some entries were modified at random (but still kept them integer) and got different sufficient and non-sufficient matrices.

The second data set (IM) contains 61 matrices. It has been built using constructions that theoretically lead to sufficient matrices, see [27]. However, they applied *principal pivotal transformation* (PPT) on integer matrices, as well. After the PPT, the resulting matrices are usually no longer integer matrices, thus the values of the entries strongly depend on the number representation. It could happen, due to the numerical errors, that some matrices become non-sufficient. Indeed, our numerical algorithms (Väliaho's algorithm and LP-B-Alg algorithm) have revealed and confirmed that 3 out of these 61 matrices are non-sufficient (they are denoted by `IM_NSU_10_07, IM_NSU_20_07, IM_NSU_20_08`). Current versions of our implementations allow us to numerically check the sufficient property of matrices only up to size $n = 30$ (Table 1).

5.5 Numerical Results

In this subsection, we report the numerical results obtained by all three algorithms on the data sets introduced in the previous subsection. Due to the exponential worst-case time complexity of all algorithms, we were able to run them only for matrices from data sets ENM and IM of size $n \leq 20$ for Väliaho's and LP-B-Alg algorithms and for $n \leq 10$ in the case of the nonlinear programming algorithm, however for Csizmadia's matrices, all three algorithms could go up to $n = 30$.

All the computations with Väliaho's and LP-B-Alg algorithms were done on the HPC system at the University of Ljubljana, Faculty of Mechanical Engineering. We used the E5-2680 V3 (1008 hyper-cores) DP cluster, with IB QDR interconnection, 164 TB of LUSTRE storage, 4.6 TB RAM, supplemented by GPU accelerators.

Since our implementations of Väliaho's and LP-B-Alg methods were done in MATLAB, we were able to use only one compute node for a computation, which on this cluster means that two processors with a total of 24 compute cores are available, since scaling the MATLAB code over more than one compute node requires the use of the MPI library and C coding, which was beyond our capacity. Therefore, we report scaling of the code speed-ups by parallelization using only 24 compute cores within one compute node, see Tables 5 and 6.

Tables 2 and 3 contain aggregated results obtained on the first two test data sets introduced in Sect. 5.4. More precisely, we ran Väliaho's and the LP-B-Alg algorithms

from Fig. 4 on all instances of size $n \leq 20$ from the ENM and IM data sets. For the matrices of size $n = 20$ from the IM data set, we were capable of accomplishing both algorithms in the time limit of 3 h only for three out of the 10 instances: for `IM_SU_20_02`, `IM_NSU_20_07`, and `IM_NSU_20_08`. Note that the last two matrices are the only non-sufficient matrices of order $n = 20$ from the IM data set, while the first matrix is PSD and this is the reason we could detect its status so easily. For the remaining 7 matrices of size $n = 20$ from the IM data set, which are all sufficient (but not PSD), Väliaho's algorithm could detect their status, while the LP-B-Alg procedure was run only on the leading principal submatrices for which it could finish computations in the time frame of 3 hours, see the results in Table 4.

The first and the second columns in Tables 2 and 3 contain the size (n) and the number of the input matrices (# mtx) aggregated in that row. The third column (# LP) contains the average number of feasibility problems ($L_{\mathcal{S}}$) that the LP-B-Alg algorithm needed to solve. The last two columns contain the average computing times needed by Väliaho's and the LP-B-Alg algorithms. When for a given dimension the table has two rows, in the first one we aggregated the instances where the value of the third column (# LP) is 0, namely only determinants were tested during the run (i.e., the LP-B-Alg method ended in step 2 or 3.1). These are the easiest matrices regarding the LP-B-Alg procedure since the hardest part (solving a large number of feasibility problems ($L_{\mathcal{S}}$)) was skipped, i.e., for these instances, we had $\det M_{\bar{\mathcal{S}}_0\bar{\mathcal{S}}_0} > 0$, for all $\mathcal{S}_0$. In the second part of the table, we report the results where the third column is positive, so these are aggregated results for the instances where we detected at least once a zero principal minor, namely, we had to solve the corresponding ($L_{\mathcal{S}}$) problems. The last two columns in both tables contain the average computing times needed by Väliaho's (time$_V$) and by the LP-B-Alg algorithms (time$_{\mathrm{LP}}$) in seconds.

Table 2 shows that very often no feasibility problem ($L_{\mathcal{S}}$) needed to be solved. When this was not the case, on average only a few instances of ($L_{\mathcal{S}}$) were needed to find a certificate for non-sufficiency.

Table 3 contains the results for sufficient matrices. Väliaho's algorithm solved all these instances in the time frame of 3 h, while for the LP-B-Alg algorithm, the situation was different. When the status can be revealed without solving ($L_{\mathcal{S}}$) (this happens if in the third column (#LP) we report zero), then this is done very efficiently, since in these cases only the determinant tests are needed. However, if the matrix is sufficient and the determinant tests are not enough, then the LP-B-Alg algorithm needs to solve (exponentially) many instances of ($L_{\mathcal{S}}$) and for $n = 20$ there were too many of them, so this algorithm could not solve them within the time limit. For example, the second row of Table 5 shows that for the 17×17 principal submatrix of `IM_SU_20_01`, the LP-B-Alg method needed to solve more than 10^8 instances of ($L_{\mathcal{S}}$), which took more than 137 h.

Table 4 contains the results for the 7 sufficient matrices for which the LP-B-Alg algorithm could not finish computations in the time frame of 3 h; therefore, we considered only the leading principal submatrices, which were still sufficient. We report in this table the largest sizes k_{LP}, for which the LP-B-Alg method produced correct results for the leading principal submatrices of order k_{LP}, and the number of instances of ($L_{\mathcal{S}}$) that the LP-B-Alg procedure had to solve. Note that all these submatrices are column and row sufficient and the LP-B-Alg algorithm gave correct answers for all

Table 2 Summary of numerical results for **non-sufficient matrices** from the first and the second data sets (one 10 and two 20 dimensional non-sufficient matrices from the data set IM and the remaining sixty matrices from the data set ENM)

n	# mtx	# LP	Time $_V$	Time$_{\mathrm{LP}}$
3	2	0	0	0.0008
4	6	0	0.0002	0.0009
5	8	0	0.0005	0.0015
6	8	0	0.0015	0.0027
7	10	0	0.0037	0.006
8	10	0	0.0076	0.0065
9	8	0	0.0035	0.0042
10	9	0	0.0090	0.0020
3	8	4.5	0.0008	0.2667
4	4	4	0.0003	0.0318
5	2	6	0.0006	0.0474
6	2	12	0.0010	0.0935
9	2	8	0.0054	0.0638
10	2	28	0.0053	0.2306
20	2	38	0.6788	2.2797

The upper part of the table contains results for instances where the determinant tests were sufficient to decide the status of the matrix. In the lower part of the table, we report results for the instances where the LP-B-Alg algorithm needed to solve (very few) instances of $(L_{\mathcal{S}})$

Table 3 Summary of numerical results for **sufficient matrices** from the data sets ENM and IM (for $n = 20$ Väliaho's algorithm solved all instances from the data set IM, while the LP-B-Alg algorithm could solve only the instance `IM_SU_20_02`, for which the determinant test was enough)

n	# mtx	# LP	Time $_V$	Time$_{\mathrm{LP}}$
3	10	0	0.0001	0.0047
4	10	0	0.0005	0.0034
5	10	0	0.0010	0.0040
6	10	0	0.0020	0.0036
7	10	0	0.0036	0.0030
8	10	0	0.0084	0.0070
9	11	0	0.0141	0.0056
10	12	0	0.0265	0.0097
20	8(1)	0	556.97	0.4032
10	8	31,415	0.2996	197.6019

of the examined (sub)matrices. It is interesting that for `IM_SU_20_05` and $k = 15$, the LP-B-Alg method did not need to solve any instance of $(L_{\mathcal{S}})$, while for $k = 16$, the computing time has already exceeded the time limit. We can see that on these matrices, Väliaho's algorithm strongly outperforms the LP-B-Alg method, since in the time frame of 3 h it can detect their status correctly for the whole matrices (see Table 3, while the LP-B-Alg algorithm only for the submatrices of order k_{LP}.

Tables 5 and 6 show how the parallel versions of Väliaho's and the LP-B-Alg algorithms scale. Computations in Table 5 were performed on the sufficient matrices of order $n = 20$ from the IM data set. As demonstrated in Table 4, checking sufficiency

Table 4 Numerical results for 7 out of 10 matrices of size $n = 20$, for which the LP-B-Alg algorithm could not detect the correct status in the time frame of 3 h, so we report for each instance the size of the leading principal submatrix for which the correct status was revealed in this time frame

Instance	n	k_{LP}	# LP
IM_SU_20_01	20	13	790,400
IM_SU_20_03	20	15	371,744
IM_SU_20_04	20	13	1,438,448
IM_SU_20_05	20	15	0
IM_SU_20_06	20	13	1,529,272
IM_SU_20_09	20	15	297,352
IM_SU_20_10	20	15	297,352

by the LP-B-Alg algorithm is very time consuming for 7 out of 8 sufficient matrices from this data set, so we did these computations with the LP-B-Alg method only for the 14×14 leading principal submatrices, and we did not impose any time limit. Additionally, we add results for 17×17 leading principal submatrix of IM_SU_20_01, which demonstrates that the LP-B-Alg algorithm may need to solve a huge number of instances of $(L_{\mathcal{S}})$ even in the case of a rather small matrix, which results in huge computing times of the LP-B-Alg algorithm. Therefore, even the parallel version of LP-B-Alg algorithm is not able to check the sufficiency of the 17×17 leading submatrix of IM_SU_20_01 in 3 h.

We can observe that the sequential and parallel versions of Väliaho's algorithm are significantly faster compared to the LP-B-Alg algorithm, which is already known from Table 4. Both algorithms scale quite well. The average scaling factor for the instances in Table 5 is 18.7 for Väliaho's algorithm, which is very good, and 6.0 for the LP-B-Alg algorithm, which is moderate. (It should be recalled that the parallel computations were performed on computing nodes with 24 available CPU cores.)

Table 6 contains the results for Csizmadia's matrices of sizes 10, 15, 20, 25, 30, which are quite different compared to results in Table 5. These matrices are sufficient and both algorithms detect this correctly, the LP-B-Alg algorithm even without solving any instance of $(L_{\mathcal{S}})$. This is not surprising since these matrices are $\mathcal{P}$-matrices (see Sect. 3.2), so all of their principal minors are positive. We can see that the computing times for the LP-B-Alg method are significantly smaller, compared to Väliaho's algorithm. Table 5 additionally demonstrates that for small n and short computation times, there is not much difference between the sequential and parallel versions of the two algorithms. In some cases, the parallel version may even require slightly more time. The reason behind this is the fact that setting up the parallel environment takes some time. If the computation time is very short, which means that the status of the matrix was revealed without starting the parallel part of the code, then parallelization is not worth it.

For larger values of n, we can observe from Table 6 that the parallel version of Väliaho's algorithm again scales well with the average scaling factor of 10.9, while the LP-B-Alg algorithm shows no scaling. The reasons for this lies again in the fact that we parallelized only one part of each algorithm:

– for Väliaho's algorithm, we parallelized only the search for a certificate for non-sufficiency over all possible submatrices of given order $s \in \{3, \ldots, n\}$, which

Table 5 Scaling property for Väliaho's and the LP-B-Alg algorithm on the seven **hard** IM **sufficient (sub)matrices** of order $n_V = 20$ and $n_{LP} = 14$ and 17, respectively

Instance	n_V	Seq.-time $_V$	Par.-time $_V$	n_{LP}	Seq.-time$_{\text{LP}}$	Par.-time$_{\text{LP}}$	# LP
IM_SU_20_01	20	677.24	34.92	14	15543.99	2433.45	2,371,400
IM_SU_20_01	20	–	–	17	493204.79	50334.09	111,102,304
IM_SU_20_03	20	669.56	58.22	14	851.47	175.58	135,552
IM_SU_20_04	20	622.27	28.93	14	27479.31	4486.70	4,315,656
IM_SU_20_05	20	662.97	30.92	14	0.05	0.05	0
IM_SU_20_06	20	591.68	27.75	14	28565.00	4775.04	4,588,136
IM_SU_20_09	20	657.46	31.07	14	1442.33	303.05	218,624
IM_SU_20_10	20	659.80	30.92	14	1463.12	303.78	218,624

Table 6 Results of Väliaho's and the LP-B-Alg algorithms on **Csizmadia's matrices**

n	Seq.-time$_V$	Par.-time$_V$	Seq.-time$_{\mathrm{LP}}$	Par.-time$_{\mathrm{LP}}$	# LP
10	0.117	1.018	0.117	0.013	0
15	0.601	1.343	0.105	0.101	0
20	18.018	2.444	3.207	3.236	0
25	682.370	33.784	117.2567	118.855	0
30	12803.490	1203.770	4882.666	4982.708	0

The LP-B-Alg algorithm never needed to solve any instance of (L_S)

means that we parallelized the internal for-loop using `parfor` option from MATLAB;

– for the LP-B-Alg algorithm, we parallelized only the internal for-loop, i.e., step 3.2, using again the `parfor` option from MATLAB.

In the case of Csizmadia's matrices, the parallelized part of Väliaho's algorithm was activated and made the difference, while in the case of the LP-B-Alg method, the parallelized part was skipped since no instance of (L_S) had to be solved.

In general, parallelization brings a difference when these internal for-loops contain many computations, otherwise not. The most difficult cases are exactly those where the internal for-loops are computationally demanding, so it makes no sense to parallelize also the outer loops as long as we work only on one compute node, regardless of how many compute cores it has, since this compute node is already fully utilized with this variant of parallelization (intra-node parallelization). However, it would be worthwhile to also implement inter-node parallelization in combination with GPUs, but this would require rewriting some parts of the code in C, using MPI and CUDA. This is beyond the scope and capacity of this current research.

5.6 Numerical Results for the NLP Approach

To test the efficiency of the NLP-based approach, we used the NEOS server [9, 15, 24], where BARON is available for scientific purposes. The proposed models are suitable for deciding whether a matrix is column sufficient or not. Since a matrix is sufficient if it is both row and column sufficient, we would have to solve the nonlinear programming problems twice (for both M and M^T) to test the sufficiency of a matrix. The numerical results presented in this section only show the time needed for deciding whether the given matrix is column sufficient or not.

For the numerical test, we first used the 3×3 to 10×10 sufficient and non-sufficient matrices from the first data set, and the 10×10 sufficient instances of the second data set, described in Sect. 5.4.

We also tried to solve the 20×20 problems from the second data set, but the running times exceeded the 8-hour time limit of the NEOS server, except for the second 20×20 matrix where the total solving time was only 0.4 s. (This instance is indeed an easy one since it is a positive semidefinite matrix.)

Table 7 Average running times of BARON for **sufficient matrices**

Data set	n	# mtx	Time
ENM	3	10	0.194
ENM	4	10	1.520
ENM	5	10	4.265
ENM	6	10	27.223
ENM	7	10	114.688
ENM	8	10	434.841
ENM	9	10	950.340
ENM	10	11	5640.118
IM	10	9	1797.335

Table 8 Average running times of BARON for **non-sufficient matrices**

n	# mtx	Time
3	10	0.230
4	10	0.332
5	10	1.145
6	10	5.749
7	10	7.740
8	10	36.350
9	10	40.561
10	10	297.689

The average running times in seconds for the sufficient matrices are shown in Table 7, and the results for the non-sufficient instances can be seen in Table 8. The first 8 rows describe the results for the first data set in both tables, and the ninth row of the first table shows the average time for the 10×10 instances of the second data set.

For the 10×10 ENM sufficient instances, the average running time was more than 1.5 h; in the worst case, it almost reached 3 h. The average running time for the IM data set was more than half an hour. However, the results for the particular problems show a considerable variation in this latter case. Six out of the 10×10 IM problems could be solved in less than 0.03 seconds, one in 0.5 seconds, but for one instance, the solution time was more than 4 hours.

The running times for the non-sufficient instances were significantly lower than for the sufficient instances. Based on these results, a possible application of the NLP approach is to run it with a time limit in the hope of finding a certificate for non-sufficiency.

We also tested the sufficiency of Csizmadia's matrices with BARON. These instances are $\mathcal{P}$-matrices; therefore, in this case the nonlinear programming problem (5) has only one feasible solution, the all-zero vector (meaning that this is the optimal solution as well). As can be seen in Table 9, BARON could solve these optimization problems very efficiently for smaller dimensions. The all-zero optimal solution vector was found in all cases shown in the table in less than 0.06 s.

Table 9 Running times of BARON for **Csizmadia's matrices**

n	10	15	20	22	25	27	30
Time	0.016	0.026	0.039	0.037	0.051	0.057	0.061

However, when we increased the dimension, numerical errors occurred. The smallest size where the exact solution is not found is 35×35. If we further increase the dimension, BARON converges to an infeasible solution.

6 Conclusion

In this paper, we discussed different results related to sufficient matrices. One of our main goals was to construct benchmark data sets with matrices for which we know whether they are sufficient or not. For this purpose, we collected already known construction rules for sufficient matrices and proposed new transformations that preserve sufficiency. We presented three data sets that can be used in the future to test the efficiency of IPAs proposed for sufficient LCPs by the operations research community. The data sets can be downloaded from the website [16]. In the future, we plan to extend our benchmark with further sufficient matrices using these construction rules and even adding right-hand side vectors of the LCPs since this choice can also affect the difficulty of solving the problem with IPAs.

Csizmadia's matrix is a good example of a sufficient matrix with a small encoding size but an exponentially big handicap. One of our research plans is to construct other such matrices. Moreover, it is still an open question what is the smallest encoding size of a given matrix that can be realized using only PPT transformations on the matrix. This result could also help answer the question raised in [13]: whether the handicap can be doubly exponential in the encoding size.

We also investigated three different algorithms to detect whether a given matrix is sufficient or not: Väliaho's algorithm, the LP-B-Alg algorithm, and a method that facilitates nonlinear programming reformulations of the definition of sufficiency. We implemented the first two algorithms in MATLAB, and in the third case, we used BARON to solve the nonlinear programming reformulations. We reported the details of their implementation and the obtained numerical results.

The numerical results demonstrate that Väliaho's algorithm is often the best choice. Usually, we can process the largest matrices with this algorithm, and when this is not the case, like for Csizmadia's matrices, then parallelization of Väliaho's algorithm gives very good speed-up factors. The LP-B-Alg algorithm is competitive when it can make the decision without solving too many linear programming subproblems ($L_{\mathcal{S}}$), which is the case for the non-sufficient instances or the instances where the determinant tests are enough. However, this algorithm takes large computing times for the sufficient instances where we need to process a very large (exponential) number of instances of ($L_{\mathcal{S}}$). In such situations, parallelization makes sense. We demonstrated how the intra-node parallelization speeds up the computations. It would also be worth implementing inter-node parallelization in combination with GPUs. This was beyond

the scope and capacity of our current research since we would need to partially re-write the code into C, using MPI and CUDA. However, to further our research, we would like to investigate the effect of these modifications.

As a summary of our numerical studies leaves a number of challenging open questions left for future research, we cannot provide a clear answer to a practitioner who needs a tool to decide whether his matrices are sufficient or not. The approaches applied in this work are complementary in the sense that some very hard instances for one algorithm were easily decided by another one. Therefore, in a situation where we have no additional information on the matrices, the best strategy based on current knowledge would be to apply all the algorithms in parallel and grasp the result that is obtained first.

Finally, the numerical results of the nonlinear approach prove that deciding whether a matrix is sufficient is a difficult nonconvex quadratic problem. The other two investigated algorithms have better performance because they take advantage of the structure of the problem. It would be valuable to investigate whether it is also possible to do so in the case of using a nonlinear solver.

Funding Open access funding provided by Corvinus University of Budapest.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Data availability The datasets generated during and/or analyzed during the current study are available on webpage [16].

References

1. Arrow, K., Debreu, G.: Existence of an equilibrium for competitive economy. Econometrica **22**, 265–290 (1954). https://doi.org/10.2307/1907353
2. Asadi, S., Mansouri, H.: Polynomial interior-point algorithm for $P_*(\kappa)$ horizontal linear complementarity problems. Numer. Algorithms **63**(2), 385–398 (2013). https://doi.org/10.1007/s11075-012-9628-0
3. Cottle, R.W.: The principal pivoting method revisited. Math. Program. **48**, 369–385 (1990). https://doi.org/10.1007/BF01582264
4. Cottle, R.W., Pang, J.-S., Stone, R.E.: The Linear Complementarity Problem. Computer Science and Scientific Computing. Academic Press, Boston (1992)
5. Cottle, R.W., Pang, J.-S., Venkateswaran, V.: Sufficient matrices and the linear complementarity problem. Linear Algebra Appl. **114**, 231–249 (1989). https://doi.org/10.1016/0024-3795(89)90463-1
6. Csizmadia, Zs.: New pivot based methods in linear optimization, and an application in petroleum industry. Ph.D. thesis, Eötvös Loránd University of Sciences, Institute of Mathematics (2007)

7. Csizmadia, Zs., Illés, T.: New criss-cross type algorithms for linear complementarity problems with sufficient matrices. Optim. Methods Softw. **21**(2), 247–266 (2006). https://doi.org/10.1080/10556780500095009
8. Csizmadia, Zs., Illés, T., Nagy, A.: The s-monotone index selection rule for criss-cross algorithms of linear complementarity problems. Acta Univ. Sapientiae Inform. **5**(1), 103–139 (2013). https://doi.org/10.2478/ausi-2014-0007
9. Czyzyk, J., Mesnier, M., Moré, J.: The NEOS server. IEEE Comput. Sci. Eng. **5**(3), 68–75 (1998). https://doi.org/10.1109/99.714603
10. Darvay, Zs., Illés, T., Majoros, Cs.: Interior-point algorithm for sufficient LCPs based on the technique of algebraically equivalent transformation. Opt. Lett. **15**, 357–376 (2021). https://doi.org/10.1007/s11590-020-01612-0
11. Darvay, Zs., Illés, T., Povh, J., Rigó, P.: Feasible corrector-predictor interior-point algorithm for $P_*(\kappa)$-linear complementarity problems based on a new search direction. SIAM J. Optim. **30**(3), 2628–2658 (2020). https://doi.org/10.1137/19M1248972
12. Darvay, Zs., Illés, T., Rigó, P.: Predictor-corrector interior-point algorithm for $P_*(\kappa)$-linear complementarity problems based on a new type of algebraic equivalent transformation technique. Eur. J. Oper. Res. **298**, 25–35 (2022)
13. de Klerk, E., E.-Nagy, M.: On the complexitiy of computing the handicap of a sufficient matrix. Math. Program. **129**, 383–402 (2011). https://doi.org/10.1007/s10107-011-0465-z
14. den Hertog, D., Roos, C., Terlaky, T.: The linear complementarity problem, sufficient matrices, and the criss-cross method. Linear Algebra Appl. **187**, 1–14 (1993). https://doi.org/10.1016/0024-3795(93)90124-7
15. Dolan, E.: The NEOS server 4.0 administrative guide. Technical Memorandum ANL/ MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory (2001). arXiv.cs/0107034
16. E.-Nagy, M.: A collection of small size sufficient matrices. https://sites.google.com/view/menagy/research/sufficient-matrices (2019)
17. E.-Nagy, M., Varga, A.: A new Ai-Zhang type interior point algorithm for sufficient linear complementarity problems. J. Optim. Theory Appl. (2022). https://doi.org/10.1007/s10957-022-02121-z
18. FICO Xpress-Optimizer. http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx
19. Fiedler, M., Pták, V.: On matrices with non-positive off-diagonal elements and positive principal minors. Czechoslov. Math. J. **12**, 382–400 (1962). https://doi.org/10.21136/CMJ.1962.100526
20. Fiedler, M., Pták, V.: Some generalizations of positive definiteness and monotonicity. Numer. Math. **9**, 163–172 (1966). https://doi.org/10.1007/BF02166034
21. Fourer, R.: AMPL: A Modeling Language for Mathematical Programming, 2nd edn. Scientific Pr., San Francisco (1996)
22. Fukuda, K., Namiki, M., Tamura, A.: EP theorems and linear complementarity problems. Discrete Appl. Math. **84**(1–3), 107–119 (1998). https://doi.org/10.1016/S0166-218X(97)00143-1
23. Fukuda, K., Terlaky, T.: Linear complementarity and oriented matroids. J. Oper. Res. Soc. Jpn. **35**, 45–61 (1992)
24. Gropp, W., Moré, J.J.: Optimization environments and the NEOS server. In: Buhman, M.D., Iserles, A. (eds.) Approximation Theory and Optimization, pp. 167–182. Cambridge University Press, Cambridge (1997)
25. Gurtuna, F., Petra, C., Potra, F.A., Shevchenko, O., Vancea, A.: Corrector-predictor methods for sufficient linear complementarity problems. Comput. Optim. Appl. **48**(3), 453–485 (2011). https://doi.org/10.1007/s10589-009-9263-4
26. Guu, S.-M., Cottle, R.W.: On a subclass of P_0. Linear Algebra Appl. **223–224**, 325–335 (1995). https://doi.org/10.1016/0024-3795(93)00271-Z
27. Illés, T., Morapitiye, S.: Generating sufficient matrices. In: Friedler, F. (ed.) Short Papers of the 8th VOCAL Optimization Conference: Advanced Algorithms, pp. 56–61. Published by Pázmány Péter Catholic University, Budapest (2018)
28. Illés, T., Nagy, M.: A Mizuno-Todd-Ye type predictor-corrector algorithm for sufficient linear complementarity problems. Eur. J. Oper. Res. **181**(3), 1097–1111 (2007). https://doi.org/10.1016/j.ejor.2005.08.031
29. Illés, T., Nagy, M., Terlaky, T.: EP theorem for dual linear complementarity problems. J. Optim. Theory Appl. **140**(2), 233–238 (2009). https://doi.org/10.1007/s10957-008-9440-0

30. Illés, T., Nagy, M., Terlaky, T.: Polynomial interior point algorithms for general linear complementarity problems. Algorithmic Oper. Res. **5**(1), 1–12 (2010). (http://journals.hil.unb.ca/index.php/AOR/article/view/11067)
31. Illés, T., Nagy, M., Terlaky, T.: A polynomial path-following interior point algorithm for general linear complementarity problems. J. Glob. Optim. **47**(3), 329–342 (2010). https://doi.org/10.1007/s10898-008-9348-0
32. Illés, T., Peng, J., Roos, C., Terlaky, T.: A strongly polynomial rounding procedure yielding a maximally complementary solution for $P_*(\kappa)$ linear complementarity problems. SIAM J. Opt. **11**(2), 320–340 (2000). https://doi.org/10.1137/S1052623498336590
33. Illés, T., Roos, C., Terlaky, T.: Simple approach for the interior point theory of linear complementarity problems with $\mathcal{P}_*$ matrices. Unpublished manuscript (1998)
34. Illés, T., Terlaky, T.: Pivot versus interior point methods: pros and cons. Eur. J. Oper. Res. **140**, 6–26 (2002). https://doi.org/10.1016/S0377-2217(02)00061-9
35. Illés, T., Török, R., Rigó, P.: Predictor-corrector interior-point algorithm based on a new search direction working in a wide neighbourhood of the central path. Corvinus Economics Working Papers, Corvinus University of Budapest **CWEP 02/2021**, pp. 1–24 (2021)
36. Illés, T., Török, R., Rigó, P.: Unified approach of interior-point algorithms for $\mathcal{P}_*(\kappa)$-lcps using a new class of algebraically equivalent transformations. J. Optim. Theory Appl. (2023). https://doi.org/10.1007/s10957-023-02232-1
37. Illés, T., Wenzel, M.: A survey on properties of P- and P_*-matrices. Unpublished manuscript (1999)
38. Kheirfam, B.: A predictor-corrector interior-point algorithm for $P_*(\kappa)$-horizontal linear complementarity problem. Numer. Algor. **66**(2), 349–361 (2014). https://doi.org/10.1007/s11075-013-9738-3
39. Klafszky, E., Terlaky, T.: Some generalization of the criss-cross method for quadratic programming. Math. Oper. Stat. Ser. Optim. **24**, 127–139 (1992)
40. Kojima, M., Megiddo, N., Noma, T., Yoshise, A.: A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems. Lecture Notes in Computer Science, vol. 538. Springer Verlag, Berlin (1991)
41. Lešaja, G., Roos, C.: Unified analysis of kernel-based interior-point methods for $P_*(\kappa)$-linear complementarity problems. SIAM J. Opt. **20**(6), 3014–3039 (2010). https://doi.org/10.1137/090766735
42. Megiddo, N.: Pathways to the optimal set in linear programming. In: Megiddo, N. (ed.) Progress in Mathematical Programming: Interior Point and Related Methods, pp. 131–158. Springer Verlag, New York (1989)
43. Morapitiye, S.: A collection of sufficient matrices. https://math.bme.hu/~sunil/su-matrices/ (2018)
44. MOSEK ApS. MOSEK optimization toolbox for MATLAB. Release 9.08.89. (available at https://docs.mosek.com/9.0/toolbox.pdf, last accessed on August 19, 2022)
45. Murty, K.: Linear and Combinatorial Programming. John Wiley & Sons. INC, New York (1976)
46. Murty, K., Yu, F.: Linear Complementarity Problems. Linear and Nonlinear Programming. Internet Edition, Ann Arbor (1997)
47. Nagy, M.: Interior point algorithms for general linear complementarity problems. Ph.D. thesis, Eötvös Loránd University of Sciences, Institute of Mathematics (2009)
48. Potra, F., Liu, X.: Predictor-corrector methods for sufficient linear complementarity problems in a wide neighborhood of the central path. Optim. Methods Softw. **20**(1), 145–168 (2005). https://doi.org/10.1080/10556780512331318038
49. Povh, J., Žerovnik, J.: On sufficient properties of sufficient matrices. Cent. Eur. J. Oper. Res. **29**(3), 809–822 (2021). https://doi.org/10.1007/s10100-021-00747-4
50. Rigó, P.: New trends in algebraic equivalent transformation of the central path and its applications. Ph.D. thesis, Budapest University of Technology and Economics, Institute of Mathematics (2020)
51. Roos, C., Terlaky, T., Vial, J.P.: Interior Point Methods for Linear Optimization. Springer Science+Business Media, New York (2005)
52. Sahinidis, N.V.: BARON 21.1.13: Global Optimization of Mixed-Integer Nonlinear Programs, User's Manual (2017). Available at http://www.minlp.com/downloads/docs/baron%20manual.pdf
53. Sonnevend, G.: An "analytic center" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. In: Prékopa, A., Szelezsán, J., Strazicky, B. (eds.) System Modelling and Optimization: Proceedings of the 12th IFIP-Conference held in Budapest, Hungary, September 1985, Lecture Notes in Control and Information Sciences, vol. 84, pp. 866–876. Springer Verlag, Berlin (1986)

54. Tawarmalani, M., Sahinidis, N.V.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**, 225–249 (2005). https://doi.org/10.1007/s10107-005-0581-8
55. The Mathworks, Inc., Natick, Massachusetts: MATLAB version 9.10.0.1613233 (R2019a) (2019)
56. Tseng, P.: Co-NP-completeness of some matrix classification problems. Math. Program. **88**, 183–192 (2000). https://doi.org/10.1007/s101070000159
57. Väliaho, H.: Criteria for sufficient matrices. Linear Algebra Appl. **233**, 109–129 (1996). https://doi.org/10.1016/0024-3795(94)00058-1
58. Väliaho, H.: P_*-matrices are just sufficient. Linear Algebra Appl. **239**, 103–108 (1996). https://doi.org/10.1016/S0024-3795(96)90005-1
59. Väliaho, H.: Determining the handicap of a sufficient matrices. Linear Algebra Appl. **253**, 279–298 (1997). https://doi.org/10.1016/0024-3795(95)00703-2
60. Ye, Y.: A path to the Arrow–Debreu competitive market equilibrium. Math. Program. **111**(1–2), 315–348 (2008). https://doi.org/10.1007/s10107-006-0065-5