A complexity analysis of Policy Iteration
through combinatorial matrices arising from Unique Sink Orientations[*]

Balázs Gerencsér, Romain Hollanders, Jean-Charles Delvenne and Raphaël M. Jungers[†]

Abstract

Unique Sink Orientations (USOs) are an appealing abstraction of several major optimization problems of applied mathematics such as for instance Linear Programming (LP), Markov Decision Processes (MDPs) or 2-player Turn Based Stochastic Games (2TBSGs). A polynomial time algorithm to find the sink of a USO would translate into a strongly polynomial time algorithm to solve the aforementioned problems—a major quest for all three cases. In addition, we may translate MDPs and 2TBSGs into the problem of finding the sink of an acyclic USO of a cube, which can be done using the well-known Policy Iteration algorithm (PI). The study of its complexity is the object of this work. Despite its exponential worst case complexity, the principle of PI is a powerful source of inspiration for other methods.

As our first contribution, we disprove Hansen and Zwick's conjecture claiming that the number of steps of PI should follow the Fibonacci sequence in the worst case. Our analysis relies on a new combinatorial formulation of the problem—the so-called Order-Regularity formulation (OR). Then, for our second contribution, we (exponentially) improve the $\Omega(1.4142^n)$ lower bound on the number of steps of PI from Schurr and Szabó in the case of the OR formulation and obtain an $\Omega(1.4269^n)$ bound.

1 Introduction

Three problems. Optimizing a linear function under a set of linear constraints is one of the most successful problems in engineering well known as *Linear Programming* (LP). Decision making in a stochastic environment is conveniently modeled using *Markov Decision Processes* (MDPs). Finding an optimal strategy for the Backgammon board game can be modeled as a *2-player Turn-Based Stochastic Game* (2TBSG). These three vastly studied problems share an important common point: they can all be seen as special families of instances of the problem of finding the *sink* of a *Unique Sink Orientation* (USO).

Unique Sink Orientations, a rich structure. Introduced by Szabó and Welzl [SW01], Unique Sink Orientations are an appealing extension for many frameworks including LP, MDPs, 2TBSGs [HPZ14, AD74, Con92, Lud95], but also *Linear Complementarity Problems* [GMR05, CPS09, SW78] or the problem of finding the *smallest enclosing ball* to a set of points [SW01, GW01] for instance. Algorithms that find the sink of USOs can also be used to solve any of the aforementioned problems. In general, a USO of a polytope is an orientation of its edges such that any face at any dimension has a unique sink (that is, a unique vertex with only incoming links). In particular, this implies that the whole polytope allows a unique sink. Polytopes arising from LP naturally exhibit a USO structure, where edges are directed towards better objective values (we here exclude degeneracies, such as when two vertices have the same objective value). In this case, the polytopes have the additional property of being *acyclic*. We then talk about Acyclic USOs, or AUSOs, and the vertices of the polytope form a partial order. The global sink corresponds to the optimal solution of the corresponding LP. MDPs and 2TBSGs exhibit an even more special structure as they can be represented as the AUSO of a *hypercube* (or Cube AUSO). Cube (A)USOs are convenient for algorithmic purposes because any vertex can be queried at any time, unlike Simplex-like methods that do only allow queries from neighbor to neighbor. Interestingly, Gärtner and Schurr showed that an LP can always be formulated as a Cube USO whose solution translates either to the solution of the

LP, or into a certificate of unboundedness or infeasibility [GS06]. However they do not mention whether the corresponding USO is acyclic or not.

Which algorithms to find the sink? A major goal in the study of (A)USOs is to find a polynomial time algorithm to find the sink. More precisely, such an algorithm should make no more than a polynomial number of vertex evaluations in the dimension or the number of facets of the (A)USO. By "vertex evaluation", we mean a request of the orientations of the edges adjacent to the vertex. In the case of LP, answering this question for Cube USOs would imply the first strongly polynomial time algorithm, a long lasting quest since the first weakly polynomial time algorithms were found about 30 years ago (namely *interior point* methods [Kar84] and *ellipsoids* methods [Kha80]). Regarding MDPs and 2TBSGs, it would be enough to find a polynomial time algorithm for Cube AUSOs to obtain the same consequence. Note that an MDP can always be formulated as an LP so it is not surprising that the problem at hand is simpler. It is more surprising for 2TBSGs though, as today no polynomial time algorithms are known to solve them in general [HMZ13].

Most algorithms to find the sink of a Cube (A)USO can be categorized along two axes:

- they can be *deterministic* or *randomized* when choosing the next vertex to query;

- at two successive steps, they can perform *local hops* (like the Simplex algorithm, from neighbor to neighbor) or large *jumps* in the cube.

A successful example of a jumping deterministic algorithm is the so-called *Fibonacci Seesaw* that applies to both USOs and AUSOs and is guaranteed to converge in at most around $O(1.61^n)$ steps, n being the dimension of the cube [SW01]. This is today the best known upper bound for deterministic (A)USO algorithms. On the other hand, *Random-Facet* for AUSOs is a local-hop randomized algorithm that is currently the only known method to solve AUSOs in sub-exponential time, namely in at most $e^{O(\sqrt{n})}$ steps [Gär02]. However, this bound was shown to be essentially tight [Mat94]. Apart from these two, many other methods have received quite a lot of attention, such as the *Product* Algorithm, *Random-Edge* or *Random-Jump* to mention only a few [SW01, HPZ14, GHZ98, MJ02, MS99].

Policy Iteration, inspiring dynamics. An interesting competitor to the Fibonacci Seesaw—and the focus of this paper—is the *Policy Iteration* algorithm (PI), a deterministic jumping algorithm specialized for AUSOs. First introduced by Howard for MDPs [How60], it was later adapted for *Parity Games*—a variant of 2TBSGs [RCN73]—and AUSOs [SS05]. PI's update rule is based on the following property: let v be a vertex of an AUSO and let u be any vertex in the sub-cube rooted in v and spanned by the out-links of v, then there is always a directed path from v to u. Intuitively, because of the acyclicity of the orientation, this means that any vertex u is "closer" to the sink than v. From v, choosing any vertex $u \neq v$ in the sub-cube as the next iterate would therefore provide an update rule that is guaranteed to converge to the global sink. PI's choice is to jump to the vertex u that is *antipodal* to v in the sub-cube and iterate from there. Therefore, it is sometimes referred to as *Bottom-Antipodal* or *Jump*. Note that Random-Jump mentioned above is based on the same principle except that it chooses any vertex of the sub-cube with uniform probability [MS99].

Complexity issues. Today's best upper bound on the complexity of PI for general MDPs and 2TBSGs is due to Hollanders et al. with a $(2 + o(1)) \cdot 2^n/n$ bound on the number of steps [HGDJ14], improving on the $O(2^n/n)$ bound from Mansour and Singh [MS99]. In some cases such as *discounted* MDPs and 2TBSGs with a fixed discount factor, PI was shown to run in strongly polynomial time [Tse90, Ye11]. A close variant of PI (namely a Simplex-like version) was also shown to run in strongly polynomial time for *deterministic* MDPs and 2TBSGs [PY13], leaving open the question for the usual PI. (The polynomial bounds in the two results were later improved in [HMZ13] and [Sch13].) Regarding lower bounds, Friedmann exhibited a construction where PI requires at least $\Omega(\sqrt[9]{2}^n)$ steps to converge for Parity Games [Fri09]. This bound was then adapted to MDPs by Fearnley and Hollanders et al. as an $\Omega(\sqrt[7]{2}^n)$ lower bound [Fea10, HDJ12]. (Fearnley translated Friedmann's construction to total- and average-cost MDPs and Hollanders et al. to discounted-cost MDPs.) For AUSOs, Schurr and Szabó provided an $\Omega(\sqrt{2}^n)$ bound [SS05]. Nevertheless, PI remains a very efficient algorithm in practice. Unfortunately, no convergence guarantees for cyclic USOs exist for PI.

A new formulation, towards new bounds. In this work, we investigate a new relaxation on the maximal number of steps taken by PI in a Cube AUSO known as the *Order-Regularity* problem (OR). First introduced by Hansen [Han12], the idea of this formulation is the following. Suppose that PI explores a sequence of vertices $v_1, v_2, ..., v_m$ in an AUSO. Here we represent vertices of the n-dimensional cube by an n-dimensional binary tuple. In the OR formulation, we record all the vertices into an $m \times n$ binary matrix so that each row corresponds to an iteration. We then translate the AUSO property into a combinatorial condition on this matrix, as defined in Section 2, Definition 1.

Using the OR formulation, Hansen and Zwick performed an exhaustive search on all OR matrices with up to 6 columns and reported the maximum number of rows (that is, iterations for PI) each time: $2, 3, 5, 8, 13, 21$. Based on these empirical observation, they conjectured that the maximum number of steps of PI should follow the *Fibonacci sequence* [Han12]. Confirming this conjecture for $n = 7$ has been claimed to be a hard computational challenge. It was introduced as January 2014's *IBM Ponder This* Challenge. Proving the conjecture in general would provide an $O(1.618^n)$ upper bound on the number of iterations of PI, a quasi-identical bound as that for Fibonacci Seesaw. Regarding lower bounds, no better bound than the one from Schurr and Szabó for the AUSO framework was known prior to our work.

Results. In this paper, our first contribution is to disprove Hansen and Zwick's conjecture by performing an exhaustive search for $n = 7$. We obtained a maximum number of rows that is lower than the expected Fibonacci number, which suggests that matching or even improving the $O(1.61^n)$ bound of Fibonacci Seesaw may be possible. Our second contribution is to (exponentially) improve Schurr and Szabó's $\Omega(1.4142^n)$ lower bound to $\Omega(1.4269^n)$, yet only in the framework of OR matrices. The key ideas behind our results both rely on the construction of large matrices satisfying OR-like conditions, which required substantial computational refinements.

Structure. The paper is organized as follows. In Section 2 we formulate the OR condition together with some key elements for its analysis. In Section 3, we discuss Hansen and Zwick's conjecture and formulate our first result. Section 4 establishes our new lower bound step by step on the number of rows of OR matrices, starting from Schurr and Szabó's construction. Then, since our results heavily rely on our ability to build large matrices, we describe in Section 5 the different ideas that we combined in order to speed up the existing computational methods. Finally, we conclude with some perspectives in Section 6.

2 Problem setting and preliminaries

Definition 1 (Order-Regularity [Han12])**.** We say that $A \in \{0,1\}^{m \times n}$ is Order-Regular (OR) whenever for every pair of rows i, j of A with $1 \leq i < j \leq m$, there exists a column k such that

$$A_{i,k} \neq A_{i+1,k} = A_{j,k} = A_{j+1,k}. \tag{1}$$

We may have $j + 1 = m + 1$. In that case, we use the convention that $A_{m+1,k} = A_{m,k}$. Furthermore, the last two rows (labeled $m - 1$ and m) are required to be distinct.

With other words, for all pairs (i, j), there exists a column k in A such that at the entries $i, i+1, j, j+1$ in this column we see either $0, 1, 1, 1$ or $1, 0, 0, 0$. Another possible reading of the OR condition in terms of Policy Iteration is as follows: at any iteration, some changes are made to the entries of the current vertex. It must always be assumed in future iterations that at least one of these changes was "right".

Note that the OR condition is invariant under permutation or negation of its columns. By *negating* a column, we mean changing all 0 entries to 1 and vice versa. Also observe that each pair (i, j) can be considered as a constraint to be satisfied by some column of the matrix. We will say that A *satisfies* a constraint (i, j) if it satisfies condition (1) for some column k.

Our aim is to find bounds on the number of rows of Order-Regular matrices.

Definition 2 (Constraint space)**.** We introduce the *constraint space* as a visualization tool to relate a binary matrix with the Order-Regularity condition. To any pair (i, j) for which condition (1) is required (i.e. for all i, j such that $1 \leq i < j \leq m$), we associate a unit square of the Cartesian grid centered at coordinate (i, j). Whenever we want to emphasize which part of the matrix satisfies which part of the

constraint space, we use a matching coloring on some subset of entries of the matrix and the corresponding squares of the constraint space.

The constraint space can be used in several ways. When considering a matrix that is not Order-Regular, it allows to visualize which constraints (i, j) are satisfied by the matrix and which ones are not, and possibly detect patterns. It also allows to visualize how each column of a matrix (or even any given part of it) contributes in achieving Order-Regularity as illustrated by Figure 1.
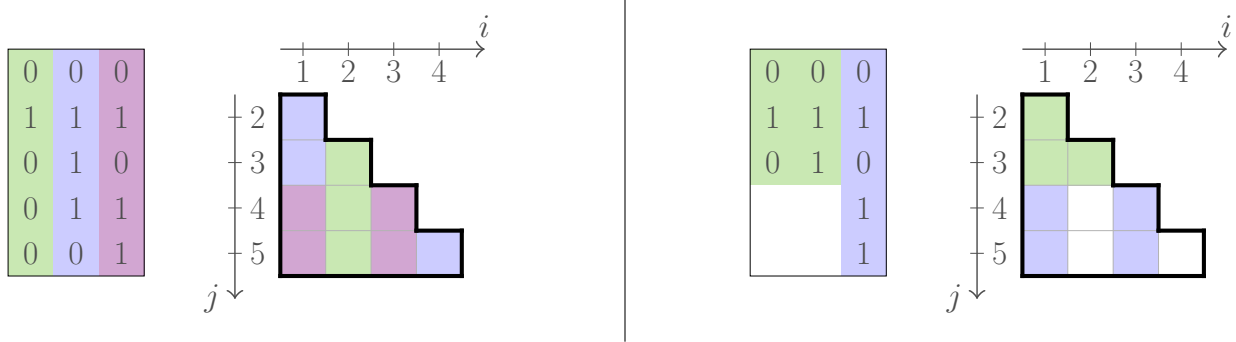

Figure 1: (Left) Each column of the matrix and the constraints it satisfies are associated with a color. (Right) The contribution of the red and blue blocks are indicated while the unfilled part of the matrix is disregarded.

Most binary vectors encountered in our constructions are repetitions of simple patterns, for which we introduce a compact notation.

Definition 3 (Patterns). A *pattern* $\left\{ {A \atop B} \right\}_M$ is a matrix composed of M copies of A or B put below each other in alternance, starting from A. Here M is called the size of the pattern. The matrices A and B are assumed to have the same number of columns. For example, $\left\{ {0 \atop 1} \right\}_5 = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T$. We sometimes omit the size of pattern if clear from the context.

The following operation is also frequently used in our constructions.

Definition 4 (Gluing). Let A be a binary matrix and let $\widetilde{A}$ be obtained from A by negating some of its columns such that the first row of $\widetilde{A}$ is identical to the last row of A. We call "the M-*gluing* of A" the construction of a matrix A' composed of M alternating copies of A and $\widetilde{A}$ such that:

$$
A' = \left\{ {A \atop \widetilde{A}} \right\}_M = \begin{bmatrix} A \\ \widetilde{A} \\ \vdots \\ A \\ \widetilde{A} \\ A \end{bmatrix}.
$$

Notice that if A is OR, then $\widetilde{A}$ is OR too. Furthermore, the first row of A is also identical to the last row of $\widetilde{A}$. The effect of gluing is illustrated in Figure 2 with A an OR matrix.

The following straightforward lemma will be of key importance for our constructions.

Lemma 1. *Let A' be the M-gluing of an OR matrix A with m rows. Then any constraint (i, j) such that $(s - 1) \cdot m < i < j \le s \cdot m$ for some integer $1 \le s \le M$ is satisfied by A'.*

Notice that if the appropriate columns of $A \in \{0, 1\}^{m \times n}$ had not been negated to obtain $\widetilde{A}$, then Lemma 1 would not have been guaranteed when $j = s \cdot m$ for any $1 \le s < M$. This is because the convention $A'_{s \cdot m, k} = A'_{s \cdot m + 1, k}$ from Definition 1 must be ensured for all columns at the connection between the different A and $\widetilde{A}$ blocks.
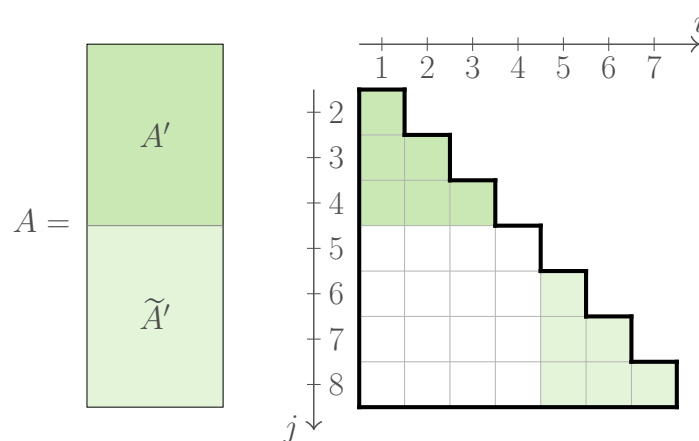
Figure 2: An illustration of the effect of a 2-gluing with a 4-rows matrix A. If the last row of A had been different from the first row of $\widetilde{A}$, then the row of $j = 4$ of the constraint space would not have been completely colored.

3 Refuting the Fibonacci Conjecture

In their works, Hansen and Zwick have performed an exhaustive search on every possible Order-Regular matrix with up to $n = 6$ columns and proposed the following conjecture that matches their observations perfectly.

Conjecture 1 (Hansen and Zwick, 2012 [Han12])**.** *The maximum number of rows of an n-column Order-Regular matrix is given by F_{n+2}, the $(n+2)^{nd}$ Fibonacci number.*

Note that except for $n = 5$, the extremal matrix found (that is, the matrix with the most rows) was always unique, up to symmetry. In Table 1, we show these extremal matrices for $n = 3$ and 4.

0	0	0	0
1	1	1	1
0	0	0	1
0	1	1	1
0	0	1	0
0	1	1	0
0	1	0	0
1	1	0	0

0	0	0
1	1	1
0	0	1
0	1	1
0	1	0

Table 1: The unique extremal Order-Regular matrices for 3 and 4 columns.

Whereas Conjecture 1 suggests that extremal OR matrices should have at most $O(\phi^n)$ rows, with $\phi = (\sqrt{5} + 1)/2 \approx 1.618$ the golden ratio, the best proven upper bound on their number of rows is presently much higher.

Proposition 1. *Let $A \in \{0, 1\}^{m \times n}$ be an Order-Regular matrix. Then $m \leq 2^{n-1} + 1$.*

Proof. Let us relax the Order-Regularity condition (1) and require the following condition instead:

$$A_{i,k} \neq A_{i+1,k} = A_{j,k}, \tag{2}$$

that is, we no longer require that $A_{j,k} = A_{j+1,k}$. Let $A' \in \{0, 1\}^{m' \times n}$ be an extremal matrix for this relaxed OR condition for a given number of columns n. Clearly m' is at least as large as the number of rows of any OR matrix with n columns since A' is strictly less constrained. We now show that m' is bounded from above by $2^{n-1} + 1$ to obtain the result. Indeed, for any rows i, j, $1 \leq i < j \leq m'$, we can never have $A'_{i+1,k} \neq A'_{j,k}$ for all columns k. Therefore, A' can never contain both a row and its negation except if one of the two is the first row. Hence it cannot have more than $2^{n-1} + 1$ rows. $\qquad\square$

Note that the above bound is optimal for the relaxation mentioned in the proof. However, the proof of this statement is beyond the scope of this paper.

In Section 5, we develop an algorithm to search for large OR matrices, with a special attention given to speed. Using this algorithm, we were able to perform an exhaustive search on all OR matrices with $n = 7$ columns. The largest matrices we found had only 33 rows, hence the following Theorem.

Theorem 1. *For $n = 7$, there exist no Order-Regular matrix with $34 = F_{n+2}$ rows and therefore Conjecture 1 fails.*

The proof of Theorem 1 is given by the computer-aided exhaustive search for $n = 7$. Our implementation of the algorithm described in Section 5 is available at `http://perso.uclouvain.be/romain.hollanders/docs/GoCode/ORsearch.zip`.

4 New lower bounds on the size of Order-Regular matrices

This section is organized as follows. First, we show a simple construction that allows to build OR matrices with n columns and $\Omega(\sqrt{2}^n)$ rows. Then we detail our construction to beat this bound as follows:

1. we introduce Strongly Order-Regular matrices, a refinement of OR matrices that we will need for our construction and improve on the $\sqrt{2}$ growth rate of the number of rows from the simple construction;

2. we describe and prove the heart of our construction and obtain a first improvement over Schurr and Szabó's bound in the setting of OR matrices;

3. we finally add an additional refinement to our construction that allows to improve our bound even a bit further to our final bound.

4.1 A family of Order-Regular matrices with $\Omega(\sqrt{2}^n)$ rows

The following construction provides Order-Regular matrices for arbitrarily large n with $m = \Omega(\sqrt{2}^n)$.

Construction 1. We recursively build a matrix $A^{(\ell)}$ as follows:

$$
A^{(\ell)} = \left[\begin{array}{c|cc} A^{(\ell-1)} & \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\} & \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\} \\ \hline \widetilde{A}^{(\ell-1)} & \left\{ \begin{smallmatrix} 1 \\ 1 \end{smallmatrix} \right\} & \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\} \end{array} \right]
$$

where $A^{(1)} = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right]$ and where $\widetilde{A}^{(\ell-1)}$ is obtained from $A^{(\ell-1)}$ by negating some of its columns such that the first row of $\widetilde{A}^{(\ell-1)}$ is identical to the last row of $A^{(\ell-1)}$. The size of all the patterns used in the construction is $2^{\ell-1}$ and the resulting matrix $A^{(\ell)}$ has $n_\ell = 2\ell - 1$ columns and $m_\ell = 2^\ell$ rows. A matching construction is given in [SS05] in the framework of Acyclic Unique Sink Orientations.

Lemma 2. *Matrices obtained from Construction 1 are Order-Regular and satisfy $m = \Omega(\sqrt{2}^n)$ with m and n its number of rows and columns respectively.*

Proof. We prove the lemma by induction on ℓ. Clearly $A^{(1)}$ is OR (only one (i, j) pair to check). We show that if $A^{(\ell-1)}$ is OR, then Order-Regularity follows for $A^{(\ell)}$.

First, observe that the left part of $A^{(\ell)}$ is a 2-gluing of $A^{(\ell-1)}$. Using Lemma 1, we get all constraints (i, j) satisfied when either $1 \leq i < j \leq 2^{\ell-1}$ or $2^{\ell-1} + 1 \leq i < j \leq 2^\ell$. The remaining constraints, i.e. those such that $1 \leq i \leq 2^{\ell-1}$ and $2^{\ell-1} + 1 \leq j \leq 2^\ell$, are satisfied by the two last columns of $A^{(\ell)}$. Indeed, if i is odd, then choosing k to be the first of the two extra columns ensures condition (1) for all $2^{\ell-1} + 1 \leq j \leq 2^\ell$. The same goes with even i's and the second of the two columns. This reasoning is illustrated by Figure 3. Furthermore, the matrix $A^{(\ell)}$ satisfies $m_\ell = \sqrt{2}^{n_\ell + 1}$. $\qquad \square$
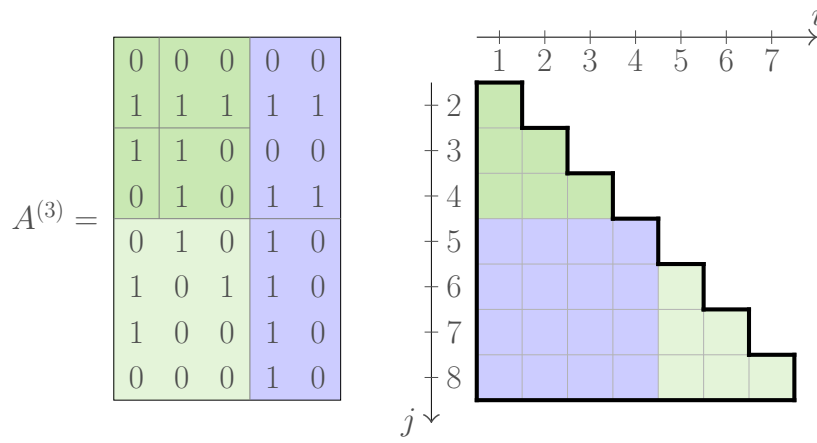
Figure 3: One step from Construction 1 is illustrated. The green matrices are glued together and ensure the top and right constraints in the constraint space. The two additional blue columns take care of the remaining square.

Using Construction 1, we have a way of building OR matrices satisfying $m = \Omega(\sqrt{2^n})$. In the next subsections, we show how we can improve this bound.

4.2 Building blocks

Similarly to the above Construction 1, our construction starts with a building block that we will use to trigger the recursion. We require the following Strong Order-Regularity condition on the building block which is a restriction of the Order-Regularity condition.

Definition 5 (Strong Order-Regularity). We say that $B \in \{0,1\}^{m \times n}$ is Strongly Order-Regular (SOR) whenever

(1) for every pair of rows i, j of B with $1 \leq i < j \leq m$, there exists a column k_1 such that:

$$B_{i,k_1} \neq B_{i+1,k_1} = B_{j,k_1} = B_{j+1,k_1} \tag{3}$$

(the original Order-Regularity condition);

(2) for every pair of rows i, j of B with $1 \leq i$ and $i + 1 < j \leq m$, there exists a column k_2 (necessarily different from k_1) such that:

$$B_{i,k_2} \neq B_{i+1,k_2} \neq B_{j,k_2} = B_{j+1,k_2}. \tag{4}$$

Again, we choose the convention that $B_{m+1,k} = B_{m,k}$ and the last two rows are required to be distinct.

In other words, at the entries $i, i + 1, j, j + 1$ we now ask for one column k_1 at which we observe either $0, 1, 1, 1$ or $1, 0, 0, 0$ and for another column k_2 at which we observe either $0, 1, 0, 0$ or $1, 0, 1, 1$. Clearly, this second column cannot exist when $j = i + 1$, hence we do not ask for its existence in that case. We say that a matrix *doubly-satisfies* a constraint (i, j) if both k_1 and k_2 exist for that constraint (that is, for an SOR matrix, every constraints such that $1 \leq i$ and $i + 1 < j \leq m$). An SOR matrix with 8 columns and 33 rows is given in Figure 4.

4.3 Blowing up

We now provide our main construction that enables us to improve the bound. We start by describing the components of each iterate of the construction. Then we show that it indeed generates Order-Regular matrices and conclude with the resulting new lower bound.

$$B = B^{(1)} = \begin{array}{|cccccccc|}
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
\hline
\end{array}$$

Figure 4: The 33×8 Strongly Order-Regular building block that we use to obtain our first improvement to the $\Omega\big(\sqrt{2}^n\big)$ lower bound.

Construction 2. Let $B = B^{(1)} \in \{0,1\}^{M \times N}$ be an SOR matrix (the building block). We inductively build a matrix $B^{(\ell)} \in \{0,1\}^{m_\ell \times n_\ell}$ as the merging of three blocks:

$$B^{(\ell)} = \begin{bmatrix} C^{(\ell)} & D^{(\ell)} & E^{(\ell)} \end{bmatrix}.$$

The blocks $C^{(\ell)}, D^{(\ell)}$ and $E^{(\ell)}$, $\ell \geq 2$, are defined as follows.

- The $C^{(\ell)}$ block is composed of M copies of the previous iterate glued together:

$$C^{(\ell)} \triangleq \left\{ \begin{matrix} B^{(\ell-1)} \\ \widetilde{B}^{(\ell-1)} \end{matrix} \right\}_M = \begin{bmatrix} B^{(\ell-1)} \\ \widetilde{B}^{(\ell-1)} \\ \vdots \\ B^{(\ell-1)} \\ \widetilde{B}^{(\ell-1)} \\ B^{(\ell-1)} \end{bmatrix}.$$

- The $D^{(\ell)}$ block expands the building block B in the following way:

$$D^{(\ell)} \triangleq \begin{bmatrix} d^{1,1} & d^{1,2} & \dots & d^{1,N} \\ d^{2,1} & d^{2,2} & \dots & d^{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ d^{M,1} & d^{M,2} & \dots & d^{M,N} \end{bmatrix}$$

with:

$$d^{i,k} \triangleq \left\{ \begin{matrix} B_{i,k} \\ B_{i+1,k} \end{matrix} \right\}_{m_{\ell-1}} = \begin{cases} \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} & \text{if } B_{i,k} = 0 \text{ and } B_{i+1,k} = 0 \\ \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} & \text{if } B_{i,k} = 0 \text{ and } B_{i+1,k} = 1 \\ \left\{ \begin{smallmatrix} 1 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} & \text{if } B_{i,k} = 1 \text{ and } B_{i+1,k} = 0 \\ \left\{ \begin{smallmatrix} 1 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} & \text{if } B_{i,k} = 1 \text{ and } B_{i+1,k} = 1 \end{cases}$$

again with the convention that $B_{M+1,k} = B_{M,k}$ for all k.

- The $E^{(\ell)}$ block is composed of two extra columns that will ensure the Order-Regularity of the whole:

$$E^{(\ell)} \triangleq \left[\left\{ \begin{matrix} \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} \\ \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} \end{matrix} \right\}_M \quad \left\{ \begin{matrix} \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} \\ \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} \end{matrix} \right\}_M \right] = \begin{bmatrix} \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} & \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} \\ \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} & \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} \\ \vdots & \vdots \\ \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} & \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} \\ \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} & \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} \\ \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}_{m_{\ell-1}} & \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}_{m_{\ell-1}} \end{bmatrix}.$$

We call $e^{i,k}$ the ith pattern encountered in the kth columns of $E^{(\ell)}$.

Given this construction, it follows that $m_\ell = M \cdot m_{\ell-1} = M^\ell$ and that $n_\ell = n_{\ell-1} + N + 2 = \ell \cdot N + 2(\ell - 1)$. Figure 5 helps visualizing the role of each block.

Definition 6 (Slices). All three blocks $C^{(\ell)}, D^{(\ell)}$ and $E^{(\ell)}$ from construction 2 are divided into M *slices* (that is, blocks of consecutive rows) of size $m_{\ell-1}$ each. We say that a row index i *belongs to* a slice s, $1 \le s \le M$, if $(s-1) \cdot m_{\ell-1} < i \le s \cdot m_{\ell-1}$. We also say that i corresponds to an *odd* (or *even*) index of s if its relative index within s is odd (or even).

We now prove the central lemma of this section.

Lemma 3. *Matrices obtained from Construction 2 using a Strongly Order-Regular matrix B with an odd number of rows as building block are Order-Regular.*

Proof. Clearly, $B^{(1)} = B$ is OR because it is also Strongly OR. Assuming that $B^{(\ell-1)}$ is OR, let us show that $B^{(\ell)}$ is also OR. Therefore, we show that each block of the construction is designed to satisfy complementary subsets of the constraints space. Figure 5 graphically illustrates on a particular case how each block contributes in filling in the constraint space.

Claim 1. *The $C^{(\ell)}$ block satisfies every constraint (i,j) where i and j belong to the same slice s of $B^{(\ell)}$.*

Claim 1 follows directly from Lemma 1 since $C^{(\ell)}$ is simply an M-gluing of $B^{(\ell-1)}$ which has $m_{\ell-1}$ rows by definition.

Claim 2. *The $D^{(\ell)}$ block satisfies every constraint (i,j) where i and j belong to two different and non-adjacent slices s_i and s_j.*

Let (i,j) be such a constraint for some integers s_i and s_j. From the Strong Order-Regularity of B and the fact that s_i and s_j are non adjacent (that is, $s_i + 1 < s_j$), we know that B doubly-satisfies the constraint (s_i, s_j). Therefore, from the definition of $D^{(\ell)}$, we know that there exist two columns k_1 and k_2 of $D^{(\ell)}$ such that the patterns that appear in the slices s_i and s_j for these columns are of the form:

$$d^{s_i,k_1} = \left\{ \begin{smallmatrix} \overline{\alpha} \\ \alpha \end{smallmatrix} \right\} \qquad\qquad d^{s_i,k_2} = \left\{ \begin{smallmatrix} \beta \\ \overline{\beta} \end{smallmatrix} \right\}$$

$$d^{s_j,k_1} = \left\{ \begin{smallmatrix} \alpha \\ \alpha \end{smallmatrix} \right\} \qquad\qquad d^{s_j,k_2} = \left\{ \begin{smallmatrix} \beta \\ \beta \end{smallmatrix} \right\}$$

for some $\alpha, \beta \in \{0,1\}$ where $\overline{\alpha} = 1 - \alpha$ and $\overline{\beta} = 1 - \beta$. Let $I(i,j) \triangleq \begin{bmatrix} i & i+1 & j & j+1 \end{bmatrix}$ be the vector of row indices needed when checking the OR condition (1) for the constraint (i,j). Then, using Matlab-like notations, two cases are possible:
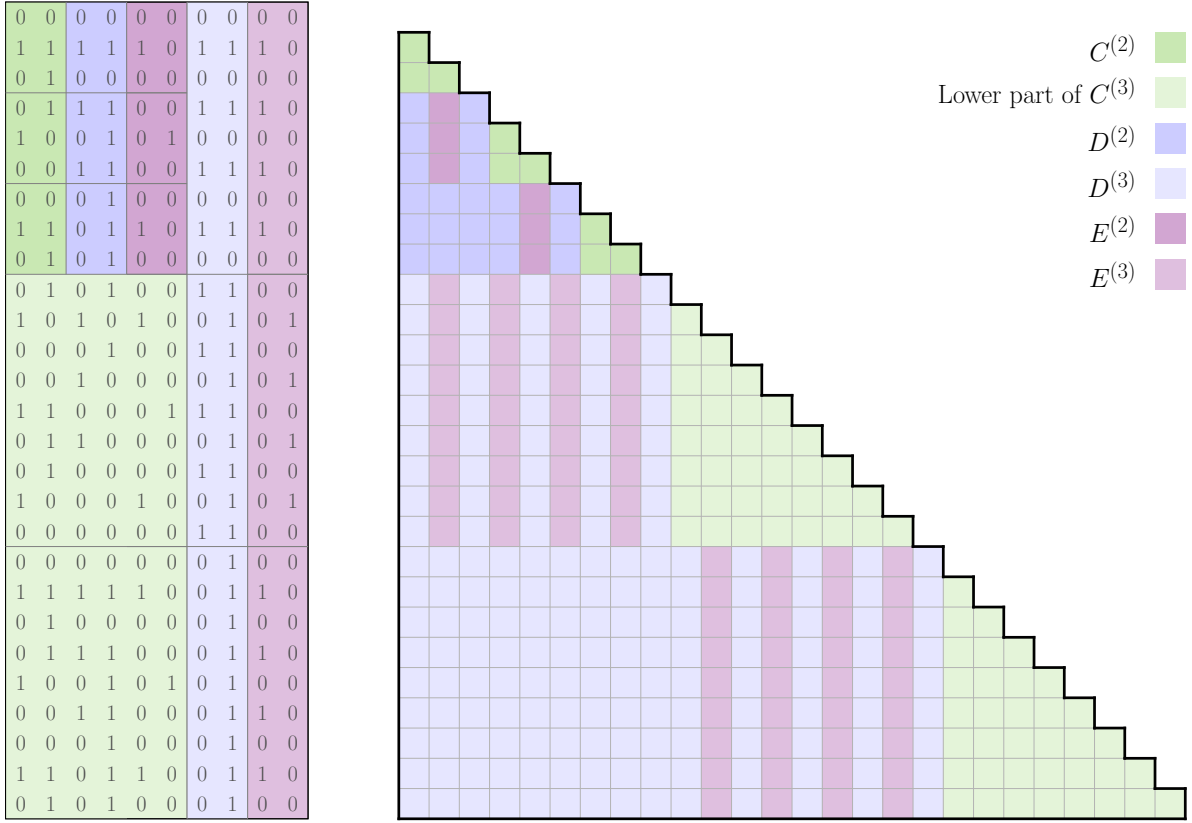
9

Figure 5: An example of two blowup steps of Construction 2 with a 3×2 SOR building block. Notice how each part of the construction contributes in filling the constraint space. The C blocks (green) fill in triangles of the size of the previous iterate along the diagonal. The D blocks (blue) almost fill in the rest of the space. The two last rows (violet) of each step of the construction aim to fill in the remaining holes.

- either $D^{(\ell)}_{I(i,j),k_1} = \begin{bmatrix} \overline{\alpha} & \alpha & \alpha & \alpha \end{bmatrix}$ and $D^{(\ell)}_{I(i,j),k_2} = \begin{bmatrix} \beta & \overline{\beta} & \beta & \beta \end{bmatrix}$;

- or $D^{(\ell)}_{I(i,j),k_1} = \begin{bmatrix} \alpha & \overline{\alpha} & \alpha & \alpha \end{bmatrix}$ and $D^{(\ell)}_{I(i,j),k_2} = \begin{bmatrix} \overline{\beta} & \beta & \beta & \beta \end{bmatrix}$.

In one or the other case there will always be a column k, either k_1 or k_2, such that condition (1) is verified. This will be true even if $i + 1$ or $j + 1$ belong to the next slice (respectively $s_i + 1$ or $s_j + 1$) thanks to the assumption that the building block B, and therefore also every iterate of Construction 2, have an odd number of rows. Indeed, because of this parity, any pattern $d^{i,k}$, of the form $\left\{ \begin{smallmatrix} \alpha \\ \beta \end{smallmatrix} \right\}$, ends with an α and the next pattern below it starts over with a β, thereby continuing the alternation of α and β for one more row.

Claim 3. *The $D^{(\ell)}$ block also satisfies every constraint (i, j) where i and j belong to two adjacent slices s and $s + 1$ and i corresponds to an* odd *index of s.*

In the case of adjacent slices, condition (4) is no longer ensured for B. However, the original Order-Regularity still holds and there exists a column k of $D^{(\ell)}$ such that $d^{s,k} = \left\{ \begin{smallmatrix} \overline{\alpha} \\ \alpha \end{smallmatrix} \right\}$ and $d^{s+1,k} = \left\{ \begin{smallmatrix} \alpha \\ \alpha \end{smallmatrix} \right\}$ for some $\alpha \in \{0, 1\}$. Since i corresponds to an odd index of s, we must have $D^{(\ell)}_{i,k} = \overline{\alpha}$ and therefore we have $D^{(\ell)}_{I(i,j),k} = \begin{bmatrix} \overline{\alpha} & \alpha & \alpha & \alpha \end{bmatrix}$ which confirms Claim 3.

Claim 4. *The $E^{(\ell)}$ block satisfies every constraint (i, j) where i and j belong to two adjacent slices s and $s + 1$ and i corresponds to an* even *index of s.*

From the definition of $E^{(\ell)}$, we know that there is always one of the two columns, say k, such that $e^{s,k} = \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}$ and $e^{s+1,k} = \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}$. Since i corresponds to an even index of s, it means that $E^{(\ell)}_{i,k} = 1$ and therefore we have $E^{(\ell)}_{I(i,j),k} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$ which confirms Claim 4.

Summary. Given any constraint (i, j):

- if i and j belong to the same slice, then the Order-Regularity condition holds for the constraint from Claim 1;

- if they belong to different slices that are non-adjacent to each other, then the condition holds from Claim 2;

- if they belong to adjacent slices, then the condition holds from Claims 3 and 4 together;

Therefore, all constraints are satisfied by $B^{(\ell)}$. □

Proposition 2 (A first improvement on the lower bound). *For all n there exists an n-column Order-Regular matrix with at least $m = \left(\sqrt[10]{33}\right)^{n-7} = \Omega(1.4186^n)$ rows.*

Proof. We use Construction 2 with the 33×8 building block from Figure 4. After ℓ steps of the construction, we get a matrix $B^{(\ell)}$ with $m = 33^\ell$ rows and $n = 10\ell - 2$ columns and therefore $m = 33^{(n+2)/10} = \Omega(\sqrt[10]{33}^n)$ when $n = 8, 18, 28, \ldots$ From Lemma 3, this matrix is Order-Regular. For a value of n such that $10\ell - 2 < n < 10(\ell+1) - 2$ for some integer ℓ, the same construction as for $n = 10\ell - 2$ applies (simply add up to 9 dummy columns to the construction to match the required number of columns). Here the worst case is when $n = 10(\ell+1) - 3$ and this is why we subtracted 9 in the exponent of the bound such that it holds for any value of n, with no incidence on the rate of growth. □

4.4 One step further: modified building block constraints

Definition 7 (Partially-Strong Order-Regularity). We say that $B \in \{0,1\}^{m \times n}$ is Partially-Strongly Order-Regular (PSOR) whenever

(1) for every pair of rows i, j of B with $1 \le i < j \le m$, there exists a column k_1 such that:

$$B_{i,k_1} \neq B_{i+1,k_1} = B_{j,k_1} = B_{j+1,k_1}$$

(the original Order-Regularity condition);

(2) for every pair of rows i, j of B with $1 < i < j < m$ and for which $j - i$ is even, there exists a column k_2 (necessarily different from k_1) such that:

$$B_{i,k_2} \neq B_{i+1,k_2} \neq B_{j,k_2} = B_{j+1,k_2}.$$

Once again, we choose the convention that $B_{m+1,k} = B_{m,k}$ and the last two rows are required to be distinct.

The difference with the Strong Order-Regularity lies in the second condition. We now no longer require the existence of the column k_2 when $i = 1$, when $j = m$ or when $j - i$ is an odd number, hence the constraints that are doubly-satisfied by a PSOR matrix are those such that $1 < i < j < m$ and $j - i$ is even. As illustrated by Figure 7, we are allowed to do this relaxation because the $E^{(\ell)}$ block from Construction 2 actually satisfies more constraints than the sole ones it was designed to satisfy initially (as referred to in Claim 4 of the proof of Lemma 3) making it possible to reduce the set of constraints that the $D^{(\ell)}$ block has to satisfy and hence to soften the SOR condition. The softened condition allows us to find a larger building block which in turn results in an improved lower bound.

Before we show why using PSOR building blocks results in OR matrices, we need to slightly adapt Construction 2, and more precisely the definition of the $E^{(\ell)}$ block.

Construction 2*. Let $\hat{B} = \hat{B}^{(1)} \in \{0,1\}^{M \times N}$ be a PSOR building block matrix. In the same spirit as Construction 2, we inductively build a matrix $\hat{B}^{(\ell)} \in \{0,1\}^{m_\ell \times n_\ell}$ as the merging of three blocks:

$$\hat{B}^{(\ell)} = \begin{bmatrix} \hat{C}^{(\ell)} & \hat{D}^{(\ell)} & \hat{E}^{(\ell)} \end{bmatrix},$$

$$\hat{B} = \hat{B}^{(1)} = \begin{array}{|cccccccc|}
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
\hline
\end{array}$$

Figure 6: The 35×8 Partially-Strongly Order-Regular building block that we use to obtain our final lower bound.

where the definitions of $\hat{C}^{(\ell)}$ and $\hat{D}^{(\ell)}$ are the same as those of $C^{(\ell)}$ and $D^{(\ell)}$ in Construction 2 with $\hat{B}^{(\ell)}$ and $\hat{B}$ taking the role of $B^{(\ell)}$ and B respectively and where

$$\hat{E}^{(\ell)} = \begin{bmatrix} \hat{e}^{1,1} & \hat{e}^{1,2} \\ \hat{e}^{2,1} & \hat{e}^{2,2} \\ \vdots & \vdots \\ \hat{e}^{M,1} & \hat{e}^{M,2} \end{bmatrix}$$

is a slight modification of $E^{(\ell)}$ such that:

$$\hat{e}^{i,k} = \begin{cases} \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\} & \text{if } i = 1 \text{ and } k = 2, \\ \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\} & \text{if } i = M \text{ and } k = 1, \\ e^{i,k} & \text{otherwise.} \end{cases}$$

The only changes compared to Construction 2 is that the second column of $\hat{E}^{(\ell)}$ now starts with $\left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}$ instead of $\left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}$ and its first column now ends with $\left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}$ instead of $\left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}$. Clearly, the modification can only help to satisfy more constraints.

Lemma 4. *Matrices obtained from Construction 2* * *using Partially-Strongly Order-Regular building blocks with an odd number of rows are Order-Regular.*

Proof. The proof is inductive, in the same flavor as the proof of Lemma 3. Knowing that $\hat{B}^{(1)}$ is OR and assuming that $\hat{B}^{(\ell-1)}$ is OR too, we show that $\hat{B}^{(\ell)}$ must also be OR.
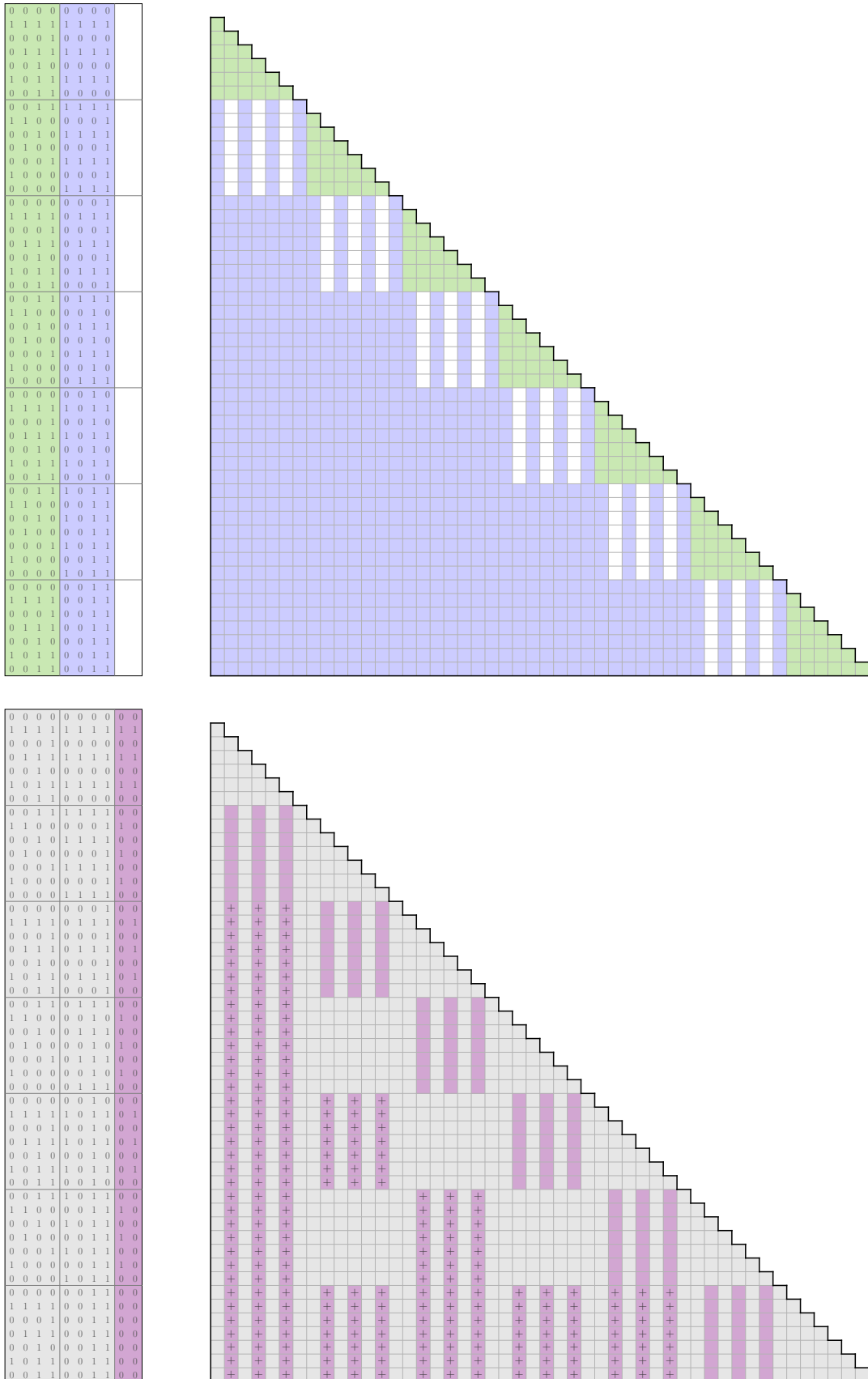
Figure 7: If we only used the $C^{(\ell)}$ and $D^{(\ell)}$ blocks (respectively in green and blue) in Construction 2 with an SOR building block, there would remain a few holes in the constraint space (top). The $E^{(\ell)}$ block is designed to fill these holes. But a slightly improved version of the $E^{(\ell)}$ block (violet) actually fills much more that just the required holes (bottom). This fact can be exploited to soften the constraints on the building block and further improve the lower bound to obtain our final bound.

Claim 1. *The $\hat{C}^{(\ell)}$ block satisfies every constraint (i,j) where i and j belong to the same slice s of $\hat{B}^{(\ell)}$.*

The argument is the same as for Claim 1 in the proof of Lemma 3.

Claim 2. *The $\hat{D}^{(\ell)}$ block satisfies every constraint (i,j) where i and j belong to two different slices s_i and s_j such that $s_i \neq 1$, $s_j \neq M$ and $s_j - s_i$ is even.*

From the Partially-Strong Order-Regularity of the building block $\hat{B}$ and the conditions on s_i and s_j, we know that the constraints (s_i, s_j) is doubly-satisfied by $\hat{B}$. Therefore, the same reasoning as the one of Claim 2 in the proof of Lemma 3 applies.

Claim 3. *The $\hat{D}^{(\ell)}$ block also satisfies every constraint (i,j) where i and j belong to two different slices s_i and s_j such that $s_i = 1$, $s_j = M$ or $s_j - s_i$ is odd and such that i corresponds to an odd index of s_i.*

Here, the constraint (s_i, s_j) is not doubly-satisfied by $\hat{B}$ but i corresponds to an *odd* index of s_i. Again, the same argument as for Claim 3 in the proof of Lemma 3 applies here.

Claim 4. *The $\hat{B}^{(\ell)}$ block satisfies every constraint (i,j) where i and j belong to two different slices s_i and s_j such that $s_i = 1$, $s_j = M$ or $s_j - s_i$ is odd and such that i corresponds to an even index of s_i.*

We evaluate the three possible cases when i corresponds to an even index of s_i.

(1) If $s_i = 1$, we have $\hat{E}^{(\ell)}_{[i\ i+1],k} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ for both columns $k = 1$ and 2 (since i corresponds to an even index of s_i), and we have $\hat{E}^{(\ell)}_{[j\ j+1],k} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ for either $k = 1$ or $k = 2$.

(2) When $s_j = M$, we have $\hat{E}^{(\ell)}_{[j\ j+1],k} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ for both $k = 1$ and 2, and we have $\hat{E}^{(\ell)}_{[i\ i+1],k} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ for either $k = 1$ or $k = 2$.

(3) If $s_i \neq 1, s_j \neq M$ and $s_j - s_i$ is an odd number, then $\hat{e}^{s_i,k_1} = \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}$ and $\hat{e}^{s_i,k_2} = \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}$ or vice versa. Furthermore, $\hat{e}^{s_i,k}$ and $\hat{e}^{s_j,k}$ are different patterns for both $k = 1$ and 2 (either $\left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}$ and $\left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}$ or vice versa). Therefore, there will always be one of the two columns, say k', such that $\hat{e}^{s_i,k'} = \left\{ \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \right\}$ and $\hat{e}^{s_j,k'} = \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\}$ and hence such that $\hat{E}^{(\ell)}_{I(i,j),k'} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$.

Summary. A constraint (i,j) such that i and j belong to the slices s_i and s_j is satisfied by:

- the $\hat{C}^{(\ell)}$ block if $s_i = s_j$;

- the $\hat{D}^{(\ell)}$ block if $s_i \neq s_j$ and the constraint (s_i, s_j) is doubly-satisfied by B;

- either the $\hat{D}^{(\ell)}$ block or the $\hat{E}^{(\ell)}$ block if $s_i \neq s_j$ and the constraint (s_i, s_j) is not doubly-satisfied by B (which is the case when $s_i = 1$, $s_j = M$ or $s_j - s_i$ is odd). □

Theorem 2. *Given a number of columns n, there exists an Order-Regular matrix with at least $m = \left(\sqrt[10]{35} \right)^{n-7} = \Omega(1.4269^n)$ rows.*

Proof. The proof is analog to the one of Proposition 2 with the PSOR building block from Figure 6 and using Lemma 4 to guarantee that the construction indeed provides OR matrices. □

5 Techniques for building large matrices

Our results heavily rely on our ability to build large (PS)OR matrices efficiently. First, to disprove Conjecture 1, we performed an exhaustive search on the massive set of OR matrices with $n = 7$ and found no matrix with $34 = F_{n+2}$ rows. Then, to obtain our lower bounds in Proposition 2 and Theorem 2, we searched for large enough matrices in the even huger set of (P)SOR matrices with $n = 8$.

Let us illustrate the size of the search space. First regarding the exhaustive search, 3×10^{11} is a conservative lower bound on the total number of OR matrices with $n = 7$, excluding symmetrical cases[1]. Therefore,

---

[1]We extrapolate the exact number to be around 3×10^{16} using a doubly exponential regression from the number of branches for $n = 1$ to 6.

we cannot afford to examine each of these matrices individually and performing an exhaustive search requires to come up with some additional tricks. Furthermore, the size of the search space grows doubly exponentially with n hence stepping from n to $n + 1$ columns makes a big difference. Including all the tricks and optimization described below, we were able to reduce the execution time to 1 month for $n = 7$ (using 10 Intel® Xeon® X5670 cores). As a comparison, the final code took less than 10 seconds for $n = 6$. This time increase when incrementing n by one suggests that the exhaustive search for $n = 8$ is very challenging. Regarding the search for building blocks, the total number of (P)SOR matrices with $n = 8$ is significantly larger than that of OR matrices with $n = 7$. However in that case, we only need to find one matrix that is as large as possible, which we achieve through the design of an efficient search strategy.

In the rest of this section, we present the techniques that we used to search the space of OR matrices without having to scan every candidate matrix and provide a pseudo-code of our algorithm. We also present the specific ideas that we used to perform an exhaustive search on the space and describe our search strategy to look for large matrices when an exhaustive search is neither within reach, nor necessary.

5.1   General principles

The steps below focus on OR matrices but an equivalent procedure applies for (P)SOR matrices as well.

Symmetry. OR matrices stay OR when a permutation or a negation is applied to some of their columns. Therefore we always assume that the columns follow each other in a lexicographical order and that the first row is composed of all 0 entries. We can also assume that the second row is composed of all 1 entries since starting a column with, e.g., 00, can only satisfy less constraints than the same (negated) column that would start with 01 instead. This way we remove redundancy in the search space.

Branching. If the first block of d rows of a matrix is infeasible itself there is no need to check the rest of the matrix. On the other hand, if the first d rows of several matrices are the same, it is unnecessary to recheck this part every time. We exploit these observations by using a depth first search on the matrices. If we have an initial block of d rows that is feasible, we try every extension to $d + 1$ rows and only continue with those that do not violate the OR condition. We are thus exploring a huge search tree whose root is by default the empty matrix and for which any node at *depth d*, that is at distance d from the root, corresponds to an OR matrix of size $d \times n$.

Remark 1 (Order-Regularity*). In this section, we use a variation of the OR condition which we refer to as OR*: we require that there exists a column k such that identity (1) is verified for all $1 \leq i < j < m$ but not for $j = m$ and we allow the last two rows to be equal. Both conditions are equivalent. Indeed, from an OR* matrix, remove the last row and it becomes OR. On the other hand, take an OR matrix and copy its last row to obtain an OR* matrix. Therefore, there exists an OR matrix with m rows iff there also exists an OR* matrix with $m + 1$ rows. Similarly, we refer to the same variation of the (P)SOR condition by (P)SOR*.

Filtering. During the branch search, assume we are investigating a branch with the first d rows fixed. In any extension, any pair of rows that we encounter later has to be compatible with the same first d rows. We can see these pairs as the rows labeled j and $j + 1$ in the order-regularity condition, to be compared with the pairs labeled i and $i + 1$ with $i < d$.

Definition 8 (Compatible pairs). Let A be some $d \times n$ Order-Regular* matrix. We define P_A, the *set of compatible pairs* of A, as:

$$P_A = \left\{ (r, q) : r, q \in \{0, 1\}^n \text{ and } \forall i, 1 \leq i < d, \exists k, 1 \leq k \leq n \text{ such that } A_{i,k} \neq A_{i+1,k} = r_k = q_k \right\}. \quad (5)$$

We also define R_A and Q_A, the projections of P_A on the set of rows that respectively appear as the first and second entry of a pair:

$$R_A = \left\{ r \in \{0, 1\}^n : \exists q \in \{0, 1\}^n \text{ for which } (r, q) \in P_A \right\}, \quad (6)$$

$$Q_A(r) = \left\{ q \in \{0, 1\}^n : (r, q) \in P_A \right\}. \quad (7)$$

Figure 8 illustrates how P_A, R_A and Q_A relate to each other on an example matrix.
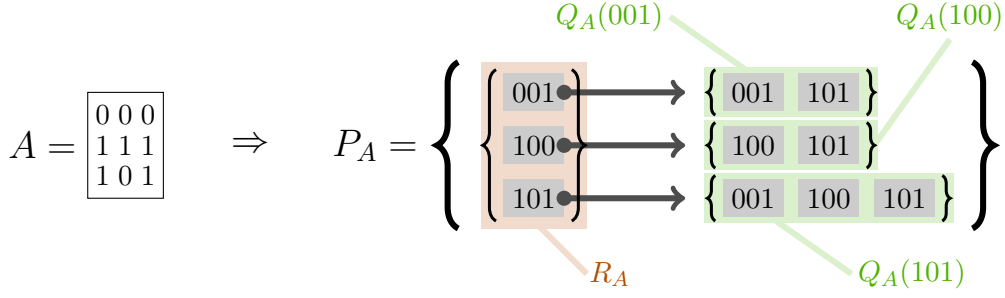
Figure 8: For this example matrix A, any pair of rows $(r, q) \notin P_A$ will never allow a valid extension of A. We encode the set of compatible pairs P_A as the set R_A where each entry r relates to a set $Q_A(r)$. In this example, notice that even though A is OR*, it does not satisfy the symmetry rules.

Given an OR* matrix A, the set of possible extension rows q such that $\begin{bmatrix} A \\ q \end{bmatrix}$ is OR* can be easily identified using P_A. Moreover, the P_A set can only shrink as we add rows to A hence the following lemma.

Lemma 5. *Let A be some $d \times n$ Order-Regular* matrix of the form $\begin{bmatrix} A^- \\ r \end{bmatrix}$ and let $A^+ = \begin{bmatrix} A \\ q \end{bmatrix}$ for some row q. Then A^+ is Order-Regular* iff $q \in Q_A(r)$. Furthermore for any r, q, it holds that $(r, q) \in P_{A^+}$ iff both $(r, q) \in P_A$ and there exists a column k such that $A_{d,k}^+ \neq A_{d+1,k}^+ = r_k = q_k$. Therefore $P_{A^+} \subseteq P_A$.*

Proof. First we observe that:

$$q \in Q_A(r) \Leftrightarrow (r, q) \in P_A,$$
$$\Leftrightarrow \forall i, 1 \leq i < d, \exists k: \quad A_{i,k} \neq A_{i+1,k} = r_k = q_k,$$
$$\Leftrightarrow \forall i, 1 \leq i < j = d, \exists k: A_{i,k}^+ \neq A_{i+1,k}^+ = A_{j,k}^+ = A_{j+1,k}^+,$$

since $A_{i,k} = A_{i,k}^+$ for all $i, 1 \leq i \leq d$. Furthermore, using the fact that A is OR*, we also have that for all $i, j, 1 \leq i < j < d$, there exists a column k such that $A_{i,k}^+ \neq A_{i+1,k}^+ = A_{j,k}^+ = A_{j+1,k}^+$. Therefore, condition (1) is verified for all $i, j, 1 \leq i < j < d + 1$ and we have that $q \in Q_A(r)$ iff A^+ is OR*.

The fact that $(r, q) \in P_{A^+}$ iff both $(r, q) \in P_A$ and there exists a column k such that $A_{d,k}^+ \neq A_{d+1,k}^+ = r_k = q_k$ follows directly from the definitions of P_A and P_{A^+}. $\square$

Direct cutting. Storing and maintaining the sets of compatible pairs of rows during the search has an additional advantage. Assume we are looking at a branch corresponding to a $d \times n$ matrix A. Then we have $|R_A|$ distinct rows appearing as r in the set of compatible pairs of rows P_A. There is clearly no way of getting more than $d + |R_A| + 1$ rows by extending this particular branch. Consequently, when searching for an $(m^* + 1) \times n$ OR* matrix, if $d + |R_A| + 1 < m^* + 1$, then we discard the node right away and make a step back in the search tree. This idea is formalized by the following lemma.

Lemma 6. *Let A be some $d \times n$ Order-Regular* matrix, let R_A be defined by equation (6) and let $m^* = d + |R_A|$. Then there exists no Order-Regular* matrix with more than $m^* + 1$ rows such that the first d rows equal A.*

Proof. First we observe that:

$$q \in Q_A(r) \Leftrightarrow (r, q) \in P_A,$$
$$\Leftrightarrow \forall i, 1 \leq i < d, \exists k: \quad A_{i,k} \neq A_{i+1,k} = r_k = q_k,$$
$$\Leftrightarrow \forall i, 1 \leq i < j = d, \exists k: A_{i,k}^+ \neq A_{i+1,k}^+ = A_{j,k}^+ = A_{j+1,k}^+,$$

since $A_{i,k} = A_{i,k}^+$ for all $i, 1 \leq i \leq d$. Furthermore, using the fact that A is OR*, we also have that for all $i, j, 1 \leq i < j < d$, there exists a column k such that $A_{i,k}^+ \neq A_{i+1,k}^+ = A_{j,k}^+ = A_{j+1,k}^+$. Therefore, condition (1) is verified for all $i, j, 1 \leq i < j < d + 1$ and we have that $q \in Q_A(r)$ iff A^+ is OR*.

The fact that $(r, q) \in P_{A^+}$ iff both $(r, q) \in P_A$ and there exists a column k such that $A_{d,k}^+ \neq A_{d+1,k}^+ = r_k = q_k$ follows directly from the definitions of P_A and P_{A^+}. $\square$

Using this trick, we are able to spot poor branches early on and hence to significantly reduce the size of the search tree without missing any OR* matrix with $34 + 1$ rows or more.

5.2 General implementation

Combining the ideas from Section 5.1, we sketch the *branch search* strategy in Algorithm 1. Notice that the starting branch needs not necessarily be the empty matrix. Though, choosing a $d \times n$ OR* matrix A as the root in Algorithm 1 will result in an OR* matrix whose d first rows correspond to the rows of A. As we will see, this option will be useful later, but then this also means Algorithm 1 only performs an exhaustive search on a restricted portion of the tree.

Algorithm 1 Branch search

Input: A, the (PS)OR* matrix of size $d \times n$ at the root of the search tree (optional, [ ] by default).
$\qquad m^{\text{target}}$, the target number of rows for the solution A^* (optional, 0 by default).

Initialization: Precompute P_A using equation (5).

Output: $A^* = \text{branchsearch}(d, A, P_A, A)$, a (PS)OR* matrix with n columns and the maximum
$\qquad$ (or the target) number of rows such that the d first rows of A^* are given by A.

1. **function** branchsearch($\ell, A^{(\ell)}, P^{(\ell)}, A^*$)
2. $\qquad$ **if** $\ell > \#\text{rows}(A^*)$ **then**
3. $\qquad\qquad$ $A^* := A^{(\ell)}$.
4. $\qquad$ **end if**
5. $\qquad$ **if** $\#\text{rows}(A^*) = m^{\text{target}}$ **then**
6. $\qquad\qquad$ **return** A^*.
7. $\qquad$ **end if**
8. $\qquad$ Extract $R^{(\ell)} := R_{A^{(\ell)}}$ from $P^{(\ell)}$ using equation (6).
9. $\qquad$ Extract $Q^{(\ell)} := Q_{A^{(\ell)}}(r)$ from $P^{(\ell)}$ using equation (7) with r being the last row of $A^{(\ell)}$.
10. $\qquad$ **if** $\ell + |R^{(\ell)}| < m^{\text{target}}$ **then**
11. $\qquad\qquad$ **return** A^*.
12. $\qquad$ **end if**
13. $\qquad$ **for** $q \in Q^{(\ell)}$ **do**
14. $\qquad\qquad$ $A^{(\ell+1)} := \begin{bmatrix} A^{(\ell)} \\ q \end{bmatrix}$.
15. $\qquad\qquad$ **if** $A^{(\ell+1)}$ satisfies the *symmetry rules* **then**
16. $\qquad\qquad\qquad$ Compute $P^{(\ell+1)} := P_{A^{(\ell+1)}}$ using equation (5).
17. $\qquad\qquad\qquad$ $A^* := \text{branchsearch}(\ell + 1, A^{(\ell+1)}, P^{(\ell+1)}, A^*)$.
18. $\qquad\qquad$ **end if**
19. $\qquad$ **end for**
20. $\qquad$ **return** A^*.
21. **end function**

Symmetry rules: the columns of the matrix must be lexicographically sorted and
$\qquad\qquad\qquad$ the first and second rows must be respectively all zeros and all ones.

Complexity issues. The steps 8 and 9 can be performed efficiently using, e.g., a two dimensional array to encode the $P_{A^{(\ell)}}$ sets. Moreover, the step 10 encodes the direct cutting according to Lemma 6. Regarding step 13, the rows q can be taken in any order. By adding randomness in the order, we allow the algorithm to return any matrix with the target size. Finally, Lemma 5 ensures that step 16 requires at most $|P_{A^{(\ell)}}|$ OR*-checks which is still the most expensive operation of each step of the recursion. Observe that since $Q_A(r) \subseteq R_A$ (as shown in the proof of Lemma 6), it holds that $|P_A| \le |R_A|^2$ and hence that the cardinality of both sets decrease together when ℓ increases.

5.3 For extremal matrices: we need exhaustive search

To further speed up our code in order to perform an exhaustive search on all OR* matrices with 7 columns, we develop a code capable of parallel processing.

Parallelization. In Algorithm 1, it is possible to perform the search in parallel on different branches of the tree. For this purpose, we first fix some depth d and precompute every possible non-symmetrical $d \times 7$ OR* matrix. These matrices act as the roots of several independent subtrees that together span the complete search tree. We then launch Algorithm 1 in parallel each time with a different root matrix as input. It finishes with the answer whenever every subtree has been completely searched.

In our case, we chose $d = 9$ which resulted in 106 million distinct subtrees of variable size. We obtained 35 subtrees that ended up with an OR* matrix of $33 + 1$ rows but none with an OR* matrix of $34 + 1$ rows, leading to the statement of Theorem 1 in Section 3.

5.4 For building blocks: we need an efficient search strategy

To search for (P)SOR building blocks with 8 columns, the strategy described in Section 5.1 still applies but the size of the search space does not allow to perform an exhaustive search. However, in this case, we only need to find a large matrix but not to prove that it is the largest (we found an SOR block with 33 rows and a PSOR block with 35 rows in our case, see Figures 4 and 6). To this end, based on the special structure of (P)SOR matrices, we designed an efficient search strategy to quickly find these large instances. We now develop this strategy for SOR matrices. An equivalent strategy exists for PSOR matrices as well but it requires to introduce some nonessential details. As in Section 5.1, we here use the variation of the (P)SOR condition denoted by (P)SOR* and defined in Remark 1.

The search strategy is based on the reversing operation that reverses the order of the rows and negates the even rows.

Definition 9 (Reversing). Let $A \in \{0,1\}^{m \times n}$. We define A^{rev}, the *reverse* of A, where for all $1 \leq i \leq m$, we have:

$$A_{i,k}^{\text{rev}} = \begin{cases} A_{m+1-i,k} & \text{if } i \text{ is odd,} \\ 1 - A_{m+1-i,k} & \text{if } i \text{ is even,} \end{cases}$$

for all columns k.

The key observation is that reversing an SOR* matrix preserves its Strong Order-Regularity*.

Lemma 7. *If $A \in \{0,1\}^{m \times n}$ is SOR*, then its reverse is also SOR*.*

Proof. For all $i, j, 1 \leq i < j < m$, let $i' = m - j$ and $j' = m - i$ so that $1 \leq i' < j' < m$. Let also $I(i,j) \triangleq \begin{bmatrix} i & i+1 & j & j+1 \end{bmatrix}$ and $I'(i,j) \triangleq m + 1 - I = \begin{bmatrix} j'+1 & j' & i'+1 & i' \end{bmatrix}$ using Matlab notations. From the Strong Order-Regularity* of A, there exist two columns k_1 and k_2 such that:

$$A_{I'(i,j),k_1} = \begin{bmatrix} \alpha & \alpha & \alpha & \overline{\alpha} \end{bmatrix} \quad \text{and} \quad A_{I'(i,j),k_2} = \begin{bmatrix} \beta & \beta & \overline{\beta} & \beta \end{bmatrix}$$

for some $\alpha, \beta \in \{0,1\}$. Then for A^{rev}, the reverse of A, we have:

$$\begin{cases} A_{I(i,j),k_1}^{\text{rev}} = \begin{bmatrix} \alpha & \overline{\alpha} & \alpha & \alpha \end{bmatrix} & \text{and} \quad A_{I(i,j),k_2}^{\text{rev}} = \begin{bmatrix} \beta & \overline{\beta} & \overline{\beta} & \overline{\beta} \end{bmatrix} & \text{if } i \text{ is } odd \text{ and } j \text{ is } odd, \\ A_{I(i,j),k_1}^{\text{rev}} = \begin{bmatrix} \alpha & \overline{\alpha} & \overline{\alpha} & \overline{\alpha} \end{bmatrix} & \text{and} \quad A_{I(i,j),k_2}^{\text{rev}} = \begin{bmatrix} \beta & \overline{\beta} & \beta & \beta \end{bmatrix} & \text{if } i \text{ is } odd \text{ and } j \text{ is } even, \\ A_{I(i,j),k_1}^{\text{rev}} = \begin{bmatrix} \overline{\alpha} & \alpha & \alpha & \alpha \end{bmatrix} & \text{and} \quad A_{I(i,j),k_2}^{\text{rev}} = \begin{bmatrix} \overline{\beta} & \beta & \overline{\beta} & \overline{\beta} \end{bmatrix} & \text{if } i \text{ is } even \text{ and } j \text{ is } odd, \\ A_{I(i,j),k_1}^{\text{rev}} = \begin{bmatrix} \overline{\alpha} & \alpha & \overline{\alpha} & \overline{\alpha} \end{bmatrix} & \text{and} \quad A_{I(i,j),k_2}^{\text{rev}} = \begin{bmatrix} \overline{\beta} & \beta & \beta & \beta \end{bmatrix} & \text{if } i \text{ is } even \text{ and } j \text{ is } even. \end{cases}$$

In every case, the Strong Order-Regularity* of A^{rev} is ensured. $\qquad \square$

Algorithm 2 Back-and-forth search

Input: d, T.

Output: An SOR* matrix.

Initialization: $A^{(0)}$, a random SOR* matrix with d rows obtained from Algorithm 1 using input
$m^{\text{target}} = d$.
$t = 0$.

1. **while** *stopping criterion* **do**
2.   Compute $B^{(t+1)}$ as the result of Algorithm 1 using input $A = A^{(t)}$.
3.   Compute B^{rev}, the reverse of $B^{(t+1)}$.
4. $A^{(t+1)} \triangleq B^{\text{rev}}_{1:d,:}$, the first d rows of B^{rev}.
5. $t \leftarrow t + 1$.
6. **end while**
7. **return** $B^{(t)}$.

Stopping criterion: after at least T steps, stop whenever $B^{(t-T+1)}$ and $B^{(t)}$ have the same number
of rows (stagnation in the last T steps).

Based on Lemma 7, we can now formulate our *back-and-forth search* strategy to find large SOR* matrices as described by Algorithm 2.

In Algorithm 2, the parameter d is typically chosen so that applying Algorithm 1 at step 2 finishes in a reasonable time (so d should be large enough to ensure a manageable size of the search trees) while leaving as much room as possible for the optimization process (so d should not be too large either). When looking for SOR* matrices with 8 columns, we typically used $d = 14$. Also note that in Algorithm 2, it is important to avoid getting the same A over and over again. We rely on the randomness introduced at step 13 of Algorithm 1 to always get a random instance of the possible B matrices. Interestingly, Algorithm 2 is guaranteed to improve the solution at each iteration, as stated by the following proposition. However, we cannot guarantee that it will find a globally optimal solution. Therefore it may be useful to restart it until finding a matrix with a suitable number of rows.

Proposition 3. *In Algorithm 2, the number of rows of $B^{(t+1)}$ is always at least as large as that of $B^{(t)}$ for all $t \geq 1$.*

Proof. At step 2 of Algorithm 2, applying Algorithm 1 with $A^{(t)}$ as the root means performing a search in a subtree of the whole tree where the d first rows are fixed. In this subtree, the matrix B^{rev} computed at the step t is a feasible solution since from Lemma 7, it is SOR* and since from step 4 its d first rows match those of $A^{(t)}$. Therefore, the best SOR* matrix $B^{(t+1)}$ that can be found in the subtree must be at least as good (in terms of its number of rows) as $B^{(t)}$. $\qquad\square$

6 Conclusions and perspectives

Prior to this work, the three main candidates to be the asymptotic maximum size of OR matrices were the lower bound $\Theta(\sqrt{2}^n)$, Hansen and Zwick's conjecture $\Theta(\phi^n)$ with ϕ the golden ratio and the upper bound $\Theta(2^n/n)$. Our results invalidate the first option and leave hope that the second may be overestimated. There are chances that the same bound as for Fibonacci Seesaw also applies here ($O(1.61^n)$ for recall).

Note that in several cases, it is possible to cast an OR matrix back into an AUSO, including when matrices are obtained from Construction 1. On the other hand, there exist OR matrices that do not correspond to any AUSO. Yet, if we could prove that a way back exists for the instances generated from Construction 2*, then our lower bound from Theorem 2 would also apply to AUSOs.

Finally, the analysis of PI through combinatorial matrices could apply to other methods based on a similar iterative principle (that is, methods that, at each iteration, choose a subset of outgoing edges and jump to the antipodal vertex in the sub-cube spanned by these edges). It would indeed be interesting to see if the OR condition can be adapted to variants of PI and if our tools can then be successfully applied.

References

[AD74]   I. Adler and G. B. Dantzig. *Maximum diameter of abstract polytopes.* Springer, 1974.

[Con92]   A. Condon. The complexity of stochastic games. *Information and Computation,* 96(2):203–224, 1992.

[CPS09]   R. W. Cottle, J.-S. Pang, and R. E. Stone. *The linear complementarity problem,* volume 60. Siam, 2009.

[Fea10]   J. Fearnley. Exponential Lower Bounds for Policy Iteration. *In Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP),* pages 551–562, 2010.

[Fri09]   O. Friedmann. An Exponential Lower Bound for the Parity Game Strategy Improvement Algorithm as we know it. *In Proceedings of the 24th Annual IEEE Symposium on Logic In Computer Science (LICS),* pages 145–156, 2009.

[Gär02]   B. Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures & Algorithms,* 20(3):353–381, 2002.

[GHZ98]   B. Gärtner, M. Henk, and G. M. Ziegler. Randomized simplex algorithms on klee-minty cubes. *Combinatorica,* 18(3):349–372, 1998.

[GMR05]   B. Gärtner, W. D. Morris, and L. Rüst. *Unique sink orientations of grids.* Springer, 2005.

[GS06]   B. Gärtner and I. Schurr. Linear programming and unique sink orientations. *In Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithm (DA),* pages 749–757, 2006.

[GW01]   B. Gärtner and E. Welzl. Explicit and implicit enforcing-randomized optimization. *Computational Discrete Mathematics,* pages 25–46, 2001.

[Han12]   T.D. Hansen. *Worst-case Analysis of Strategy Iteration and the Simplex Method.* PhD thesis, Aarhus University, Science and Technology, Department of Computer Science, 2012.

[HDJ12]   R. Hollanders, J.-C. Delvenne, and R. M. Jungers. The Complexity of Policy Iteration is Exponential for Discounted Markov Decision Processes. *In Proceedings of the 51st IEEE Conference on Decision and Control (CDC),* 2012.

[HGDJ14]   R. Hollanders, B. Gerenscér, J.-C. Delvenne, and R. M. Jungers. Improved bound on the worst case complexity of policy iteration. *arXiv preprint arXiv:1410.7583,* 2014.

[HMZ13]   T. D. Hansen, P. B. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM (JACM),* 60(1):1, 2013.

[How60]   R.A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA, 1960.

[HPZ14]   T. D. Hansen, M. Paterson, and U. Zwick. Improved upper bounds for random-edge and random-jump on abstract cubes. *In Proceedings of the 25th Symposium On Discrete Algorithms (SODA),* pages 874–881, 2014.

[Kar84]   N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica,* 4(4):373-395, 1984.

[Kha80]   L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics,* 20(1):53–72, 1980.

[Lud95]   W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Information and Computation,* 117(1):151–155, 1995.

[Mat94]   J. Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures & Algorithms*, 5(4):591–607, 1994.

[MJ02]   W. D. Morris Jr. Randomized pivot algorithms for p-matrix linear complementarity problems. *Mathematical Programming*, 92(2):285–296, 2002.

[MS99]   Y. Mansour and S. Singh. On the Complexity of Policy Iteration. *In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999.

[PY13]   I. Post and Y. Ye. The simplex method is strongly polynomial for deterministic Markov Decision Processes. *In Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1465–1473, 2013.

[RCN73]   S. S. Rao, R. Chandrasekaran, and K. P. K. Nair. Algorithms for discounted stochastic games. *Journal of Optimization Theory and Applications*, 11(6):627–637, 1973.

[Sch13]   B. Scherrer. Improved and generalized upper bounds on the complexity of Policy Iteration. *In Proceedings of the 27th Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 386–394, 2013.

[SS05]   I. Schurr and T. Szabó. Jumping doesn't help in abstract cubes. pages 225–235, 2005.

[SW78]   A. Stickney and L. Watson. Digraph models of bard-type algorithms for the linear complementarity problem. *Mathematics of Operations Research*, 3(4):322–333, 1978.

[SW01]   T. Szabó and E. Welzl. Unique Sink Orientations of Cubes. *In Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 547–555, 2001.

[Tse90]   P. Tseng. Solving H-horizon, Stationary Markov Decision Problems in Time Proportional to log(h). *Operations Research Letters*, 9(4):287-297, 1990.

[Ye11]   Y. Ye. The Simplex and Policy-Iteration Methods are Strongly Polynomial for the Markov Decision Problem with a Fixed Discount Rate. *Mathematics of Operations Research*, 2011.