An Algorithmic Approach for the Zarankiewicz Problem

Matthias Werner
Freiberg University of Mining and Technology
Institute of Computer Science
D-09596 Freiberg, Germany
Email: wmatthias@t-online.de

Corrected Version: October 29, 2012

Abstract

Let $k(m,n) = (k_{ij})$ be a Boolean matrix with $1 \leq i \leq m$ rows and $1 \leq j \leq n$ columns. A rectangle is a $(2,2)$-submatrix consisting of ones. What is the maximum number of values 1 in $k(m,n)$ such that there are no rectangles? This problem can be restated as the Zarankiewicz Problem, whose complexity grows very fast. With systematic enumeration an instance with $m = n = 10$ already needs about 2315 years to be solved on a single CPU.

This paper provides an exact algorithm that solves the Zarankiewicz Problem. The developed Overflow Algorithm utilizes simple combinatorial properties to reduce the solution space. The instance with $m = n = 10$ mentioned above will be computed in 18 min. Based on the structure of the computed solutions further improvements are given, computing that instance in just 2 s. However, this extended reduction in run time is paid by the loss of exact solutions for matrices of certain dimensions.

1 Introduction

In a given Boolean matrix $k(m,n)$ with m rows and n columns a pair of row indices (a,b) and a pair of column indices (c,d) are chosen to form four points in the matrix. When values 1 are assigned to all these four corners then a rectangle is formed as shown in Figure 1. $k(m,n)$ is said to be rectangle free, if for all pairs (a,b) and (c,d) in $k(m,n)$ no rectangles occur as illustrated in Figure 2. $k(m,n)$ represents an adjacency matrix for a bipartite graph with m vertices on one side of its bipartition and n vertices on the other side. If $k_{ij} \equiv 1$, then node i is connected with node j. A complete bipartite graph $K(m,n)$ denotes a bipartite graph where each node from one class is connected with every node from the other class. The adjacency matrix of a complete bipartite graph would be filled with ones completely.

Definition 1 (see [5, p. 187]). *Given $m \geq 3, n \geq 3$ and $r \geq 2, s \geq 2$. The Zarankiewicz Function $Z_{r,s}(m,n)$ determines the least positive integer such that if a Boolean matrix $k(m,n)$ contains $Z_{r,s}(m,n)$ ones then it must have a submatrix $h(r,s)$ with r rows and s columns consisting entirely of ones. For $r = s = 2$ we define:* $\mathbf{maxrf}(m,n) := Z_{2,2}(m,n) - 1$.

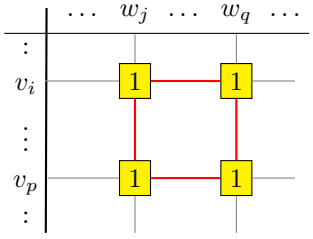

Figure 1: Forbidden rectangle

$v_i \setminus w_j$	w_1	w_2	w_3	w_4	w_5
v_1	1	1	1	1	0
v_2	1	0	0	0	1
v_3	0	1	0	0	1
v_4	0	0	1	0	1
v_5	0	0	0	1	1

Figure 2: Rectangle free $k(5,5)$

It follows from Definition 1 that $\mathbf{maxrf}(m,n)$ is the maximum number of values 1 in the Boolean matrix $k(m,n)$ such that there are no rectangles formed. Figure 2 shows an optimal solution for $m = n = 5$ with the maximum of 12 ones. Swapping columns or rows yields an isomorphic solution, because it corresponds to relabeled vertices within a partition of the bipartite graph. For higher dimensions there can be a lot of other non-isomorphic solutions.

[3] gathers many results and bounds as well as revealing very interesting applications for the Zarankiewicz Problem. For instance, it is closely related with the Rectangle Free Grid Coloring Problem [2, 6]. A Grid $G(m,n)$ is an adjacency matrix for a complete bipartite graph, where every edge has a color and c colors are available. The Grid is rectangle free, if no monochromatic rectangles, i.e. rectangles with all four corners the same color, are formed in $G(m,n)$. In other words a Grid $G(m,n)$ has a c-coloring, if all the edges of $G(m,n)$ can be colored with c colors and $G(m,n)$ is rectangle free.

Example 1. *For a $G(3,3)$ there exists a rectangle free 2-coloring. By removing nodes and associated edges a valid 2-coloring for smaller graphs can be obtained as well.*

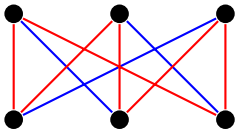

Figure 3: Rectangle free

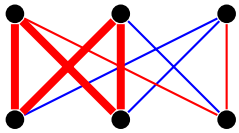

Figure 4: Forbidden rectangle

Figure 3 shows a valid 2-coloring, whereas Figure 4 contains a monochromatic rectangle.

By increasing the size of $G(m,n)$ more and more colors are needed to keep the graph free of monochromatic rectangles. For a given amount of colors c, there are thresholds, which are aggregated to the so called Obstructions Set $OBS(c)$. $OBS(c)$ contains graphs $G(p,q)$, where $G(p+1,q)$ or $G(p,q+1)$ no longer can be colored with c colors without inducing a monochromatic rectangle. In the last years $OBS(c)$ has been completed for $c \in \{2,3\}$ [2], and due to the latest research results [6] also for $c = 4$ and the not yet published solution of B. Steinbach for $K(21,12)$. The following lemma creates the link between the Rectangle Free Grid Coloring and the Zarankiewicz Problem.

Lemma 1 (see [2, p. 4])**.** *If $G(m,n)$ has a rectangle free c-coloring then $G(m,n)$ contains a monochromatic rectangle free subgraph $H \subseteq G(m,n)$ such that for the number of edges $|E(H)|$: $|E(H)| \geq \lceil \frac{m \cdot n}{c} \rceil$. If $\mathbf{maxrf}(m,n) < \lceil \frac{m \cdot n}{c} \rceil$, then $G(m,n)$ has no c-colorings.*

Proof. A valid c-coloring of $G(m,n)$ consists of c monochromatic subgraphs. Due to the pigeonhole principle the number of the edges of one subgraph is equal or greater than $\lceil \frac{m \cdot n}{c} \rceil$. □

An easy, recursive algorithm for computing small dimensions can be found in [8] (solves 7×7 in 28 min). To get a better comprehension of the time complexity of the Zarankiewicz Problem, the number of possible assignments have to be examined. Given the grid of $k(m,n)$ there are $m \cdot n$ entries which indicate by the values 1 or 0 whether $k(m,n)$ contains the edge $e_{ij} := (v_i, v_j)$ or not. Hence $2^{m \cdot n}$ assignments are possible. Rows and columns can be permuted yielding an isomorphic graph. The number of assignments modulo row and column permutations is taken from `http://oeis.org/A089006`. Each assignment has to be checked further for rectangles what involves $\binom{n}{2}\binom{m}{2}$ operations for one assignment.

Table 1: Time estimation to compute $\mathbf{maxrf}(m,n)$

$k = m = n$	evaluate all grids	utilize permutations
8	1.71×10^5 a	66 h 44 min
11	9.55×10^{22} a	2.83×10^{10} a
12	1.15×10^{30} a	4.29×10^{15} a

Table 1 shows theoretical times to calculate $\mathbf{maxrf}(m,n)$ for quadratic grids without (with) utilization of row and column permutations. These estimated times assume a single CPU with 2.67 GHz and 1 instruction per cycle. Even 8×8 would take almost 67 h to compute $\mathbf{maxrf}(8,8)$ with taking advantage of permutation classes. The next section will realize an exact algorithm solving $\mathbf{maxrf}(8,8)$ in just 1 s and $\mathbf{maxrf}(11,11)$ in 7 days.

2 Overflow Algorithm

2.1 Principle of the Overflow Algorithm

The Boolean matrix $k(m,n)$ can be represented as a list of positive numbers a_i in binary notation. Because the permutation of rows yields an isomorphic solution, $a_{i+1} \geq a_i$ is assumed. So no solutions will be created, which would be actually the same by just permuting the rows of them. Running $a_i \in \{1,\ldots,2^n-1\}$ generates all bit patterns for the i-th row with n bits. After a_i has reached the limit 2^n-1, a_{i-1} is incremented by one, if possible (otherwise trying to increment a_{i-2} and so on). The algorithm will finish when a_1 overflows.

Algorithm 2.1 Overflow Principle as pseudo code

```
1:  a_1 ← 1
2:  i ← 1
3:  while a_1 < 2^n do                ▷ algorithm will finish, when a_1 reaches the limit
4:      for i = i + 1, ..., m do      ▷ reset all successors of a_i to a_i
5:          a_i ← a_{i-1}
6:      end for
7:      while a_m < 2^n − 1 do        ▷ a_m runs through {a_{m-1}, ..., 2^n − 1}
8:          a_m ← a_m + 1             ▷ yield an instance for k(m, n)
9:      end while
10:     repeat
11:         i ← i − 1                 ▷ find i where a_i is not at the limit yet
12:     until a_i < 2^n − 1  or  i = 1
13:     a_i ← a_i + 1
14: end while
```

The values $\{1 \leq a_i < 2^n\}$ can be represented as nodes sorted in a row where the most right node begins with $a_i = 1$. A matrix of bit patterns is obtained, where the current state of the algorithm represents a path as shown in Figure 5. Due to rows order only steps to the south and west are allowed. For the complexity the total number of paths has to be computed.

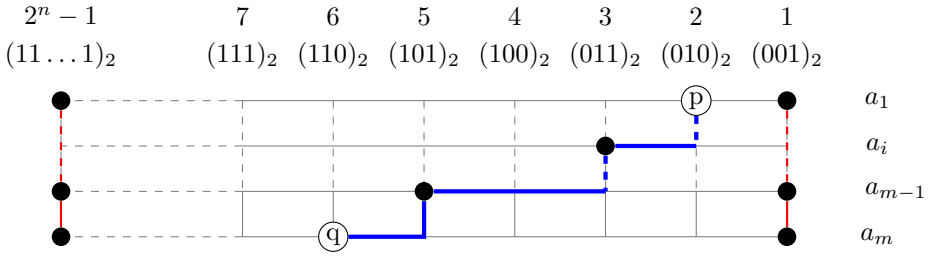

Figure 5: Paths of patterns taken by the Overflow Algorithm

Theorem 1. *Let $\{a_1,\ldots,a_m\}$ be a set of positive numbers with $a_i \in \{1,\ldots,T\}$ as well as $T := T(n) \geq 1$ and $a_i \leq a_{i+1}$, $i \in \{1,\ldots,m\}$. The number $N(m,n,a_1,T)$ of paths from a_1 to T is:*

$$N(m,n,a_1,T) = \binom{T-a_1+m}{m}$$

Proof. Given a bit patterns submatrix with m rows, $a_1 = p$, $a_m = q$ and the alphabet $\Gamma = \{W,S\}$. W and S describe a step to west and south respectively. A word must contain $(q-p =: j) \times W$ and $(m-1) \times S$. Hence the number of words with the length $q-p+m-1$ is $\binom{m+j-1}{j}$. Since each path can be identified by its words with S-W combinations we obtain the result by applying Pascal's rule:

$$N(m,n,a_1,T) = \sum_{j=0}^{T-a_1} \binom{m+j-1}{j} = \binom{T-a_1+m}{m}$$

□

In our case $T = 2^n-1$ is given, thus n should be minimal ($m \geq n$ was already assumed).

If the adjacency matrix is longish enough, then there is a direct result. Given $m \geq \binom{n}{2}$ then the exact value of the Zarankiewicz Function easily can be computed (for proof see [9, p. 23] while [3, p. 130] gives a generalisation for $Z_{r,s}(m,n)$):

$$\mathbf{maxrf}(m,n) = Z_2(m,n) - 1 = m + \binom{n}{2}, \quad m \geq \binom{n}{2} \tag{1}$$

Corollary 1 (see [9, p. 8]). *If the most significant bit in the first row a_1 is shifted from the 0-th to the p-th position, the runtime improves by:*

$$\frac{c_p(m,n)}{c_0(m,n)} := \frac{N(m,n,2^p,2^n-1)}{N(m,n,1,2^n-1)} = \prod_{i=1}^{m} \frac{2^n - 2^p + m - i}{2^n - 1 + m - i}$$

Let be $\beta(p,m,n) = \left(1 - \frac{2^p-1}{2^n-2}\right)^m$. One shows that $\beta(p,m,n) \leq \frac{c_p(m,n)}{c_0(m,n)}$ and if m grows slower than $\sqrt{2^n} = 2^{\frac{n}{2}}$, then the absolute error $\Delta(p,m,n) := \left|\frac{c_p(m,n)}{c_0(m,n)} - \beta(p,m,n)\right|$ converges to zero and hence $\beta(p,m,n)$ converges to $\frac{c_p(m,n)}{c_0(m,n)}$.

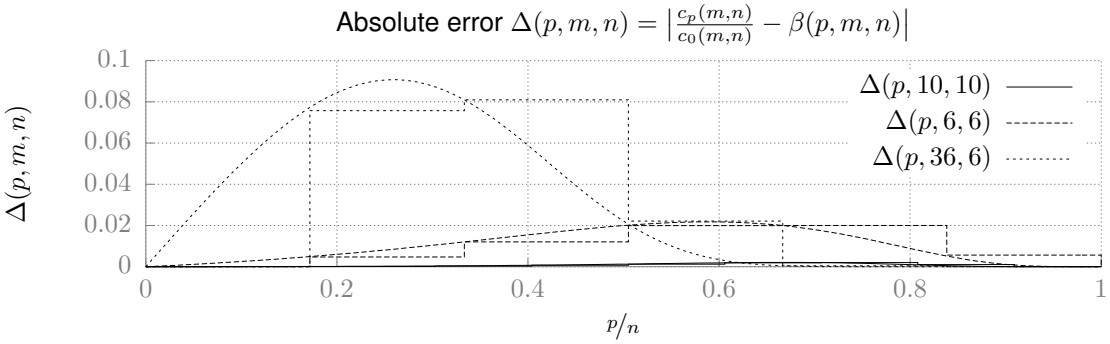

Figure 6: Absolute error $\Delta(p,m,n)$ by estimating $\frac{c_p(m,n)}{c_0(m,n)}$ with $\beta(p,m,n)$

Figure 6 shows the absolute error, which occurs by estimating $\frac{c_p(m,n)}{c_0(m,n)} = \frac{N(m,n,2^p,2^n-1)}{N(m,n,1,2^n-1))}$ with $\beta(p,m,n)$. The error becomes visible when m is much higher than n, as $\Delta(p,36,6)$ illustrates well. In the opposite $\Delta(p,6,6)$ gives a deviation of not more than 2 percentage points. The estimation with $\beta(p,m,n)$ already becomes almost accurate on 10×10.

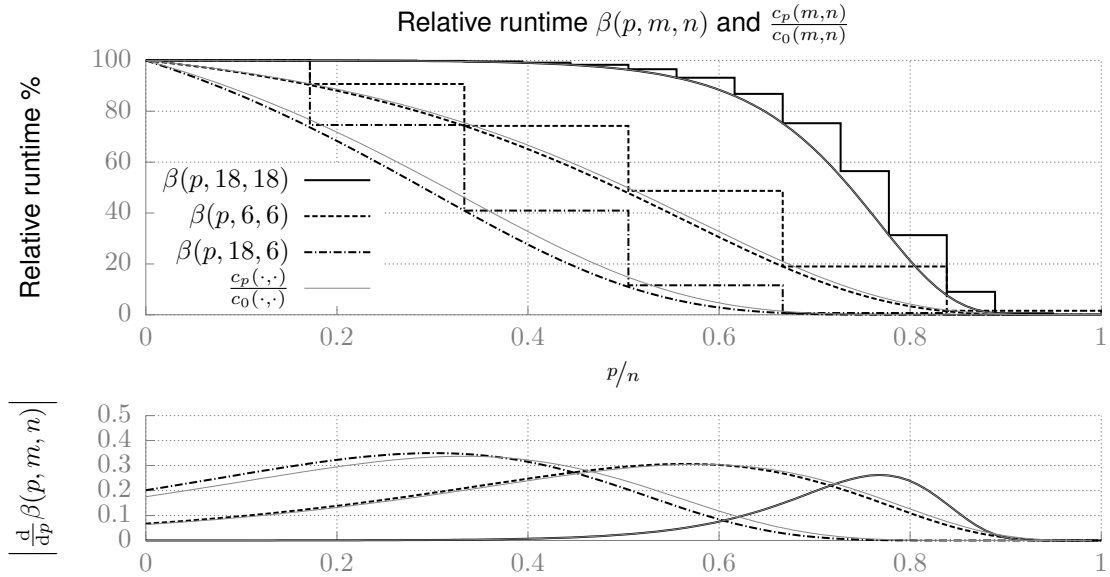

Figure 7: Relative runtime improvement with $a_1 = 2^p$ and $p \in [0,n)$

Figure 7 illustrates the relative runtime behaviour of the Overflow-Algorithm, where a_1 starts at 2^p instead of $2^0 = 1$. The approximation quality of $\beta(p,m,n)$ is shown as well. The estimation $\beta(p,m,n)$ converges to $\frac{c_p(m,n)}{c_0(m,n)}$ under the conditions of Corollary 1. The bigger n the later the runtime acceleration takes impact due to the influence of the ratio $\frac{2^{p+1}-2^p}{2^n} = \frac{2^p}{2^n}$. Small bit positions would do almost nothing to reduce the amount of paths. For $m = n = 18$ the theoretical runtime improves noticeable when the most significant bit is moved at least to the 10th position. With the help of Corollary 1 the considerably simpler derivation of the relative runtime estimation can be utilized for the maximum of the relative runtime acceleration:

$$\begin{aligned} p_{\max}(m,n) := \arg\max\left\{\left|\tfrac{\mathrm{d}}{\mathrm{d}p}\beta(p,m,n)\right|\right\} &= \log_2\left(\tfrac{2^n-1}{m}\right) \\ &= n - \log_2(m) - \log_2(1-2^{-n}) \\ &\approx n - \log_2(m) \end{aligned} \tag{2}$$

For instance, the maximum acceleration of $\beta(p,18,18)$ is $p_{\max}(18,18) = 13.830069\ldots \approx 13.83$. The exact maximum acceleration given by $\arg\max\left\{\left|\frac{\mathrm{d}}{\mathrm{d}p}\frac{c_p(m,n)}{c_0(m,n)}\right|\right\}$ for $m = n = 18$ is $13.830121\ldots$ In Subsection 2.3 we will come back to Equation 2.

In Algorithm 2.1 we have omitted the validation for rectangles. This validation involves $\binom{m}{2}$ tests on the set of the m numbers a_i. Two numbers a and b form a rectangle (Figure 1), if they have at least two one's bits in common, i.e. `c=a&b` is neither zero nor a power of two, so one would write this test: `bool has_rectangle := (c and (c & (~c + 1)) != c);`, where '&' is the bitwise AND operator and '~' the unary bitwise complement.

2.2 Strategies for Improvements

The implemented Overflow Algorithm works without recursion and uses several improvements to speed up the runtime. If a row a_i is incremented then a_{i+1} has to be reset to a_i (since a_{i+1} caught an overflow before). If a_i has two or more bits then $a_{i+1} \leftarrow a_i$ would yield a rectangle. So the second bit of a_{i+1} is shifted up by one position, the following bits are reset to zero:

$$\begin{array}{c|c} a_i & 0\ 1\ 0\ 0\ 1\ 0 \\ a_{i+1} & 0\ 1\ 0\ 1\ 0\ 0 \end{array} \qquad \begin{array}{c|c} a_i & 0\ 1\ 0\ 1\ 0\ 1 \\ a_{i+1} & 0\ 1\ 1\ 0\ 0\ 0 \end{array}$$

Figure 8: Rectangle free incrementing a_{i+1} with respect to a_i

The assignment also can be skipped if $a_i \geq (1100\ldots0)_2$, $i < m$, because a_{i+1} then would always have the first two bits in common with a_i.

Another improvement is to estimate for a given $i \in \{1,\ldots,m-1\}$ the maximum value $\mathbf{maxrf}(m-i,n)$ in the remaining subgrid $(m-i,n)$ by an upper bound $z(m-i,n)$: Let be $e_{\max}$ the current maximum, b_{i-1} the accumulated bit count from previous rows and i the current row. If $b_{i-1} + z(m-i,n) \leq e_{\max}$ then this assignment can be skipped. a_i or the previous rows have not enough bits to improve the local maximum. Now $z(\cdot,\cdot)$ can return the optimal value, if it is already known or it computes an upper bound instead. Following theorem uses [9, p. 25].

Theorem 2. *Let be $m \geq n > 2$. Then* $\mathbf{maxrf}(m,n) \leq \frac{1}{2}\left(m + \sqrt{m^2 + 4 \cdot mn(n-1)}\right)$.

Proof. The number of bits of row a_i, namely the pattern size, is denoted by $d_i \in \{1,\ldots,n\}$. So $\mathbf{maxrf}(m,n) = \sum_{i=1}^{m} d_i$ gives the total number of bits in $k(m,n)$. To proof the upper bound an ideal pattern size $\sigma \in [1,n]$ has to be computed. If $m = m_0 = \binom{n}{2}$ then $\mathbf{maxrf}(m_0,n) = 2 \cdot m_0$ due to Equation 1. Every row of the Boolean matrix $k(m_0,n)$ has two bits. For $m < m_0$, rows from $k(m_0,n)$ can be combined to create higher patterns. Since $k(m_0,n)$ is rectangle free, no rectangles can occur in $k(m,n)$.

A pattern with size σ acquires $\binom{\sigma}{2}$ rows from $k(m_0,n)$. Since every row will have the same pattern size σ, the following condition must be met: $m \cdot \binom{\sigma}{2} \stackrel{!}{=} m_0 = \binom{n}{2}$. Solving this for positive σ yields: $\sigma = \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{n(n-1)}{m}}$. Hence we obtain the upper bound by assigning σ to all rows:

$$\mathbf{maxrf}(m,n) \leq m \cdot \sigma = \frac{m}{2} + \sqrt{mn(n-1) + \frac{m^2}{4}} \tag{3}$$

□

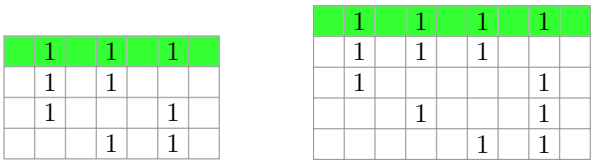

Figure 9: Recipes for creating higher patterns from 2-bit patterns

Figure 9 illustrates the creation of higher pattern sizes using 2-bit patterns as explained in the proof of Theorem 2. For instance, a 3-bit pattern acquires three 2-bit patterns. The upper bound in Equation 3 was found first by I. Reiman [4]. For $n \leq m \leq 14$, the bound is optimal or overestimates the number of edges by only one. Yet it gives a noticeable impact on the runtime of the algorithm, since more branches have to be traversed by the algorithm. Thus we will take the exact maximum value of a subgrid, unless otherwise specified.

Table 2: Runtime with $z(\cdot,\cdot)$ for estimation of the maximum of subgrids

	Upper Bound	Exact
maxrf$(9,9)$	5 min 49 s	23 s
maxrf$(10,10)$	37 h 19 min	18 min

2.3 Applying Heuristics

Figure 10 shows the solution structure of the last found optimum by the Overflow Algorithm. It constructs a nice bow pattern of the most significant bits. The Slot Algorithm in [9, p. 15] uses the so called Slot Architecture that is inspired by B. Steinbach's work in [7]. The Overflow Algorithm seems to produce that Slot Architecture by itself, i.e. subdividing the matrix in groups (slots). Figure 10 shows instances with a slot configuration "$(4,3,2)$", which is defined by the size of grouped most significant bits in rows 6-9, 3-5 and $1,2$.

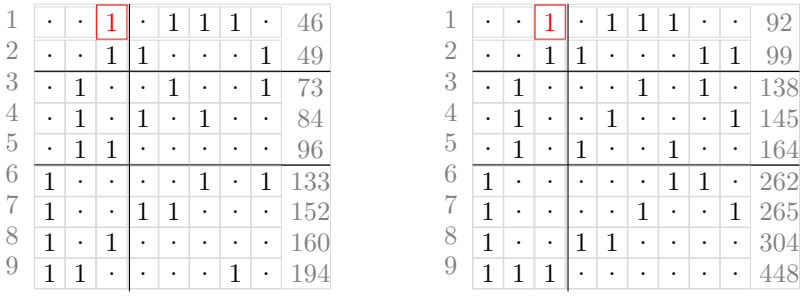

Figure 10: Last obtained optimum for 9×8, 9×9

Another interesting result can be observed on the first row a_1 and its first significant bit. Its position p seems to fit $p_{\max}(m,n) \approx n - \log_2(m)$ found in Equation 2. Now the key idea is to fix a_1 to a pattern where the leading bit is placed at $\lfloor n - \log_2(m) \rfloor$. However the number of bits of a_1 must be known, most likely their positions are important too. The optimal pattern of a_1 as well as the whole optimal Slot Architecture is still unknown. So the pattern of a_1 is estimated with the help of the ideal pattern size σ from Theorem 2. It is assumed that the pattern size of a_1 is $\sigma(s) := \lfloor \sigma + s \rfloor$ with $s \in [0,1)$. s is a weight for choosing the next higher pattern size that must also exist in the optimal solution (assuming that its set of pattern sizes is connected). Figure 11 illustrates some possible configurations of the first three rows depending on σ.

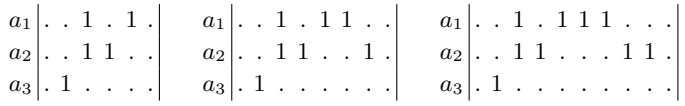

Figure 11: Configurations of first three rows where a_1 is a fixed pattern

3 Experimental Results

Applying the restrictions from the heuristics mentioned in Subsection 2.3 a significant speed up is achieved as illustrated in Table 3. However accuracy no longer can be hold on other certain dimensions. For instance, the leading bit p of a_1 for 12×8 and 12×9 is $p_{\max}(\cdot,\cdot) - 1$. So only a suboptimal solution is found with the suggested heuristic. Conversely 5×5 has an optimal solution with p at $p_{\max}(5,5) + 1$ (yet an optimal solution with $p = p_{\max}(5,5)$ exists). Following questions arise from the applied heuristic ($\hat{a}_1$ refers to the first row of an optimal solution):

- What is the highest leading bit position for $\hat{a}_1$?
- Does $\hat{a}_1$ really have a pattern with size $\sigma(s)$ for given $s \in [0,1)$?
- Which of the other bit positions in $\hat{a}_1$ are possible at all?
- How many rows are in the first group (slot)? (We assumed two rows for the first group, where a_2 can elevate and a_3 already is in the next slot.)

Table 3: Number of solutions and runtime (Intel X5650 2.67 GHz), $\sigma = \sigma(s),\ s = 0.3$

maxrf$(\cdot,\cdot)$		Full Run		#Sol.	a_1 fixed		#Sol.	σ
maxrf$(7,7)$	$= 21$	3 ms	(3 ms)	1	0.5 ms	(0.1 ms)	1	3
maxrf$(14,7)$	$= 31$	11 s	(2 s)	27 641	400 ms	(1.7 ms)	282	2
maxrf$(8,8)$	$= 24$	1 s	(72 ms)	16 596	14 ms	(1 ms)	128	3
maxrf$(9,9)$	$= 29$	23 s	(9 s)	4464	354 ms	(171 ms)	144	3
maxrf$(10,10)$	$= 34$	18 min	(34 s)	32 838	1.6 s	(1.6 s)	1	3
maxrf$(11,11)$	$= 39$	7 d 4 h	(9 h)	1 168 996	34 min	(6 min)	432	4
maxrf$(12,12)$	$= 45$	–	–	–	12 h 28 min	(5 h 7 min)	72	4

Table 3 gives a comparison of the exact and the heuristic variant of the Overflow Algorithm. Additional to the number of optimal solutions, the complete runtime and the time to find the optimal solution (in brackets) is enumerated. The pattern size for a_1 is given in case of the heuristic variant $\sigma(s)$. For 12×12 an optimal solution was found after just 5 h (see Table 1), where $\hat{a}_1$ consists of 4 bits with the most significant bit at $p = 8$.

Table 4: Comparison of relative runtime - theoretical and *experimental* results for 10×10

p	1	2	3	4	5	6	7	8
$\frac{N(10,10,2^p,2^n-1)}{N(10,10,1,2^n-1)}$	99.0 %	97.1 %	93.4 %	86.3 %	73.6 %	53.1 %	26.7 %	5.8 %
$\beta(p,10,10)$	99.0 %	97.1 %	93.4 %	86.3 %	73.5 %	52.9 %	26.5 %	5.7 %
Full Run	100.4 %	100.4 %	99.9 %	93.7 %	64.4 %	53.9 %	88.3 %	5.2 %
First Solution	99.0 %	99.7 %	101.7 %	88.8 %	71.0 %	65.9 %	–	–

Table 4 compares the experimental runtime improvement, where the most significant bit of a_1 starts[1] at $p \in 0,\ldots,n-1$. So the base value 100% refers to the full runtime of the Overflow Algorithm starting with $p = 0$, i.e. $a_1 = 2^p = 1$. With $p = 6$ the algorithm needs 53.9 % of the time compared to $p = 0$. For $p > 6$ there is no optimal solution. Since the maximum could not be found at all, the branches cannot be discarded as early as for $p \leq 6$, where the maximum was found quite fast after 34 s. Thus the relative runtime increases back to 88.3 % for $p = 7$. For $p = 8$ the branches become very lightweight and only a few number of paths are visited at all.

[1] a_1 is not fixed, but only starts at $a_1 = 2^p$ and is incremented until $(1011\ldots1)_2$ is exceeded.

4 Conclusion

The developed exact algorithm solves the fast growing Zarankiewicz Problem in a non-recursive manner and uses simple structure informations to speed up. In comparison to [8] the Overflow Algorithm solves 7×7 in 3 ms instead of 28 min. Table 2 in Strategies for Improvements shows that just the estimation of the maximum of the remaining subgrids by an upper bound can have a noticeable time impact. Even the smallest overestimation of the maximum of the subgrids yields significant time differences. Of course, the exact maximum of a subgrid help to discard a time consuming branch more early.

Based on the complexity observations combined with the solution structure of the Overflow Algorithm we introduced heuristics at the expense of accuracy on certain dimensions. Yet an enormous speedup could be achieved and the 12×12 could be solved correctly in just 12.5 hours. This illustrates the performance gain when the pattern of the first row in the adjacency matrix is known. Moreover the heuristics give a clue how to create such a pattern or at least a range of patterns for the first row. It will be a future task to discover these important patterns.

The source code of the Overflow Algorithm and the Slot Algorithm, which we did not discuss here, is available on my website, see [9]. There also some results and connections to the 4-coloring of 21×12 problem.

References

[1] Darryn E. Bryant and Hung-Lin Fu. C_4-saturated bipartite graphs. *Discrete Mathematics*, 259(1-3):263–268, December 2002. ISSN 0012-365X. doi: 10.1016/S0012-365X(02)00371-0. URL http://dx.doi.org/10.1016/S0012-365X(02)00371-0.

[2] Stephen Fenner, William Gasarch, Charles Glover, and Semmy Purewal. Rectangle Free Coloring of Grids. *ArXiv e-prints*, May 2010. URL http://arxiv.org/abs/1005.3750.

[3] Richard Guy. A many-facetted problem of Zarankiewicz. In G. Chartrand and S. Kapoor, editors, *The Many Facets of Graph Theory*, volume 110 of *Lecture Notes in Mathematics*, pages 129–148. Springer Berlin / Heidelberg, University of Calgary Canada, 1969. ISBN 978-3-540-04629-5. URL http://dx.doi.org/10.1007/BFb0060112. 10.1007/BFb0060112.

[4] I. Reiman. Ueber ein Problem von K. Zarankiewicz. *Acta Mathematica Academiae Scientiarum Hungaricae*, 9:269–279, 1958. doi: 10.1007/BF02020254.

[5] Steven Roman. A problem of Zarankiewicz. *Journal of Combinatorial Theory, Series A*, 18(2):187–198, 1975. ISSN 0097-3165. doi: 10.1016/0097-3165(75)90007-2. URL http://www.sciencedirect.com/science/article/pii/0097316575900072.

[6] B. Steinbach and C. Posthoff. Extremely Complex 4-Colored Rectangle-Free Grids: Solution of Open Multiple-Valued Problems. *Proceedings of the IEEE 42nd International Symposium on Multiple-Valued Logic (ISMVL 2012)*, pages 37–44, 2012. ISSN 0195-623X/12. doi: 10.1109/ISMVL.2012.12.

[7] B. Steinbach and C. Posthoff. The Solution of Ultra Large Grid Problems. *21st International Workshop on Post-Binary ULSI Systems (ULSI 2012)*, pages 1–10, 2012.

[8] B. Steinbach and C. Posthoff. Search Space Restriction for Maximal Rectangle-Free Grids. *10th International Workshop on Boolean Problems*, 2012.

[9] M. Werner. Algorithmische Betrachtungen zum Zarankiewicz Problem. Seminararbeit, May 2012. URL http://11235813tdd.blogspot.de/2012/02/zarankiewicz-problem.html.