Recognizing Integrality of Weighted Rectangles Partitions

Paul Deuker[1] and Ulf Friedrich[*2]

[1]Technical University of Munich, Department of Mathematics, Munich, Germany. ORCID: 0000-0001-8061-4632.
[2]Otto von Guericke University, Faculty of Mathematics, Magdeburg, Germany. ORCID: 0000-0001-6566-245X.

January 23, 2023

The weighted rectangles partitioning (WRP) problem is defined on a set of active and inactive pixels. The problem is to find a partition of the active pixels into weighted rectangles, such that the sum of their weights is maximal. The problem is formulated as an integer programming problem and instances with an integral relaxation polyhedron are characterized by the balancedness of the problem matrix. Numerical experiments illustrate the benefits of detecting balancedness for solving WRP.

Keywords: partition problem, integer programming, balanced matrix.

1 Problem Description

We begin by defining the two-dimensional weighted rectangles partitioning (WRP) problem formally. The setting can be generalized to three or more dimensions in a straightforward way by using higher-dimensional orthotopes instead of two-dimensional rectangles.

For $p, q \in \mathbb{N}$ we consider a rectangular *problem field* $R_{p,q}$ which is divided in $p \times q$ pixels. Each pixel of the problem field is either *active* or *inactive* as illustrated in Figure 1 with $p = 5$ and $q = 7$. We say that a set of pixels is active if all contained pixels are active, e.g., in the upper row of the problem field in Figure 1 there is a 1×2 and a 1×3 active rectangle. Note that WRP can easily be generalized to non-rectangular problem

[*]Corresponding author.

fields by extending the field with empty pixels to a larger rectangle. A problem field is interpreted as a binary $p \times q$-matrix where each active pixel is represented by a one and each inactive pixel is represented by a zero.

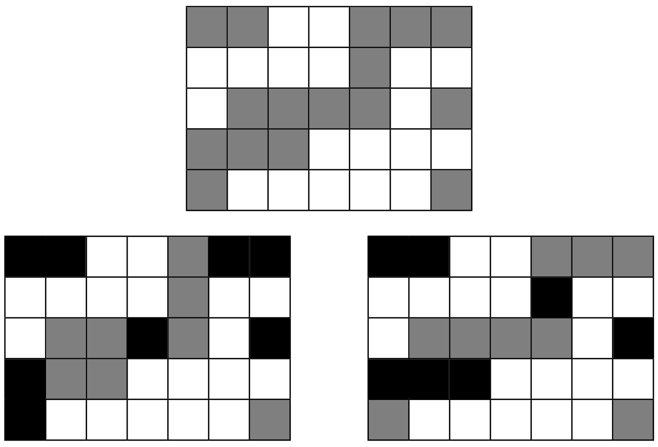


Figure 1: Two different rectangle partitions of a 5×7 problem field. Active pixels are displayed in gray in the upper picture, two sample partitions are displayed in black and gray below.

Let $S_{p,q}$ be the set of all $k \times l$-rectangles which are a product of $k \leq p$ and $l \leq q$ consecutive pixels, i.e., all sub-rectangles of $R_{p,q}$. Each $s \in S_{p,q}$ is assigned a weight $w(s) \in \mathbb{R}$. We consider a generic weight function. Depending on the weights, it can be beneficial to split or merge rectangles to obtain a higher total weight. For example, splitting a 2×2 rectangle with a weight of 10 into two 2×1 rectangles with a weight of 6 will result in a higher total weight.

Definition 1. Given a problem field $R_{p,q}$ and a weight function $w : S_{p,q} \to \mathbb{R}$, the WRP problem is the task to partition all active pixels into disjoint rectangles in $S_{p,q}$ such that the sum of the weights of the used rectangles is maximal.

In Definition 1, *partition* means that all active pixels have to be covered by rectangles without covering any inactive pixels and *disjoint* means that no two rectangles share the same pixel. The problem is always feasible as a trivial solution uses one 1×1 rectangle for each active pixel. Two non-trivial solutions for the above example setting are given in Figure 1. Both solutions consist of eight disjoint rectangles, but can have a different overall value depending on the weight function.

1.1 Related Problems and Applications

The WRP problem is similar to several two-dimensional problems of the packing or partitioning type. In what follows we relate WRP to problems already discussed in the optimization literature. We refer the reader to the survey article [15] for an in-depth comparison of the several variants and to the library of instances and computational benchmarks in [11].

In general, WRP can be identified as a set partitioning problem [7]. Given the set of all *possible* rectangles, the task is to find a partition that meets the problem's preconditions – in our case defined by the active pixels. If a brief, informal description of WRP is sought, we can relate WRP to a two-dimensional cutting stock problem [9, 12] with holes. In both problems, a collection of rectangles is sought that yields the highest possible profit or the lowest possible cost. The main difference is that in the cutting stock typically not all rectangle types may be used, whereas in WRP the field has spaces not to be occupied.

Another optimization problem with many similarities to WRP is the two-dimensional knapsack problem, see [16]. In both problems, rectangles have to be placed on a problem field while maximizing profit. The problem field of WRP can be interpreted as a two-dimensional knapsack with special pockets, where not every item fits into every pocket. A difference to WRP is that each rectangle size can be used arbitrarily often. The two-dimensional strip packing problem [5, 17] can be interpreted as a generalization of WRP to an unbounded problem field.

The literature on rectangular decomposition, e.g., [14, 18, 21], considers the problem of partitioning a given rectangular shape with or without holes into a minimum number of smaller rectangles. This has been extended to more general shapes, see the survey in [22]. In principle, we can reduce this problem to WRP by setting the weights of all rectangles in R to -1. However, the purely discrete setting of WRP is different from the geometric perspective of general rectangle decomposition algorithms which is why we do not detail this line of argumentation here. In [1], a partitioning problem similar to WRP is studied from an approximation perspective. The authors give a concise summary of approximation algorithms and present a quasi-polynomial approximation method based on dynamic programming.

In theoretical computer science, rectangle partitions are an important tool to study the communication complexity of a protocol, see [13]. In particular, any protocol can be interpreted as a partition into active and inactive rectangles. The partition number of a protocol, defined as the smallest number of rectangles to partition both the active and inactive pixels, is a lower bound on the deterministic communication complexity [13, Proposition 2.2]. We can compute the partition number within our model by setting all weights to -1 and solve the problem both for the active and inactive pixels.

Our study of WRP is inspired by the rather unusual application of a slot machine game by an Australian gambling machine manufacturer [2]. The game has a playing field of size 5×7 pixels on which randomly some pixels are active. The goal of the game is to connect adjacent active pixels to rectangles that have different weights such that the sum of all weights is maximal.

1.2 Structure of this Work

In the next section, we formulate WRP as an integer programming problem (IP). Based on this formulation, we search for integral instances of WRP by checking the balancedness of the system matrix. Our main result is the complete characterization of balanced instances given in Theorem 8. In Section 3, we compare an algorithm for checking the balancedness and the total unimodularity of the system matrix to the direct solution of the problem as an integer programming problem. Finally, we assess our findings in Section 4 and point to possible future research directions.

2 Solving the WRP

2.1 IP Formulation

We begin the analysis of WRP by introducing an IP formulation of WRP. Let B be the set of active pixels and let $R = R(B) \subseteq S_{p,q}$ be the set of all *possible* rectangles, where possible means they consist of pixels $b \in B$ only. Ignoring the dependency on B in the notation, we define the matrix $\boldsymbol{A} = (a_{r,b})$ as the incidence matrix of all possible rectangles. In particular, the columns of $\boldsymbol{A}$ are indexed by the rectangles $r \in R$ and the rows correspond to the active pixels $b \in B$, where $a_{r,b} = 1$ if and only if $b \in r$. We define a variable x_r for all $r \in R$ and write $w_r = w(r)$ for the corresponding weight.

Using the above notation, we can formulate WRP as an IP problem.

$$\begin{aligned} \max \quad & \sum_{r \in R} w_r x_r \\ \text{s.t.} \quad & \sum_{r \in R} a_{r,b} x_r = 1 \qquad \forall\, b \in B \\ & x_r \in \{0,1\} \qquad \forall\, r \in R \end{aligned} \tag{1}$$

Apparently, this formulation uses the reasoning of a typical set partitioning formulation, cf. [3], and a key task in its solution is to (implicitly) generate the set R to be partitioned.

2.2 Integral WRP Instances

The solution of the IP (1) is easy when the polyhedron corresponding to the linear relaxation of the feasible set has only integer vertices, i.e., when the instance is integral. A standard result for verifying this property is the Hoffman-Kruskal Theorem [10] that links integral polyhedra to the total unimodularity of the system matrix $\boldsymbol{A}$.

Instead, we apply results on balanced matrices developed in [4, 6, 8] to check for the integrality of the polyhedron. In particular, the polyhedron defined by the linear relaxation of (1) is integral if the system matrix is balanced, see [20, Theorem 21.7]. As every totally unimodular matrix is balanced [20, Section 21.5], we can find more integral instances checking balancedness than by checking total unimodularity. In addition, checking for balancedness can be performed more efficiently for WRP than testing for total unimodularity, see Section 3.

We build upon the original definition of a balanced matrix in [4] using hypergraphs.

Definition 2. A hypergraph $H = (V, E)$ is *balanced* if every odd cycle

$$(a_1, E_1, a_2, E_2, \cdots, E_{2v+1}, a_1)$$

in H has an edge $E_i \in E$ which contains at least three of its vertices $a_j \in V$. A $\{0,1\}$-matrix is called *balanced* if it is the incidence matrix of a balanced hypergraph.

It is easy to see that every *balanced matrix* is a {0,1} matrix that does not contain any square submatrix of odd order with all row sums and all column sums equal to 2. An *unbalanced* matrix is a matrix that does not satisfy the property in Definition 2. We extend this definition for WRP by considering so-called journeys instead of cycles.

Definition 3. A *step* in a WRP problem field is a pair of two pairwise disjunct pixels, which are both contained in the same active rectangle. A *journey* in a WRP problem field is a sequence of different steps, in which every pixel is appearing exactly twice. A journey is *odd* if it contains an odd number of steps.

We can characterize balanced WRP problem fields with the help of journeys. First note that each submatrix in Definition 2 corresponds to a journey.

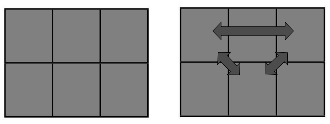
Lemma 4. *Every square submatrix $\boldsymbol{M}$ in Formulation* (1) *of odd order with row sums and column sums equal to 2 corresponds to an odd journey in the WRP problem field.*

Proof. Every column of $\boldsymbol{M}$ represents one rectangle of the problem field and each row of $\boldsymbol{M}$ represents one pixel of the problem field. Since $\boldsymbol{M}$ has row sums and all column sums equal to 2, it defines a journey. Moreover, this journey is odd because $\boldsymbol{M}$ is of odd order. $\square$

Next, we construct three types of problem fields that are unbalanced. These examples are the key observations for the complete characterization of balanced fields of WRP in Theorem 8.

Lemma 5. *A problem field containing a 3×2 rectangle of active pixels is unbalanced.*

Proof. Given the 3×2 rectangle of active pixels in Figure 2, consider the journey indicated by the arrows. We show that this journey leads to an unbalanced submatrix. First,



$$\left(\begin{array}{ccccccccccc|c}
1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & \\
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & I_6 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 &
\end{array}\right)$$

Figure 2: An unbalanced combination of 6 active pixels, a journey using the pixels, and the corresponding matrix below.

observe the corresponding matrix representation of the problem field in Figure 2, where

$\boldsymbol{I}_d$ is the identity matrix in d dimensions. The gray fields correspond to a 3×3 submatrix which is the representation of the journey described in the figure. Clearly, this matrix is unbalanced. □

In the same manner, we can prove that all problem fields that contain the left or middle combination in Figure 3 are unbalanced. A possible journey is defined by the arrows in the right image in Figure 3.

Lemma 6. *The problem fields in Figure 3 and all fields containing them as a subset are unbalanced.*

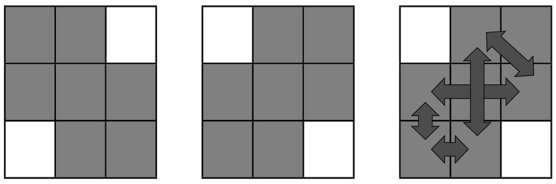


Figure 3: Left and middle: Two unbalanced combinations of active pixels. Right: Illustration of a journey showing that the combinations are unbalanced.

Finally, we examine a third constellation inducing the unbalancedness of the problem field.

Lemma 7. *A problem field containing a connected set of active pixels with inactive pixels inside is unbalanced.*

Here, two pixels are *connected* if they define 2×1 or 1×2 active rectangle.

Proof. In a problem field with a set of connected active pixels with inactive pixels inside it is straightforward to find a journey: The rectangles and active pixels next to the inactive pixels can be used to define the steps of a journey. This idea is exemplified in the rightmost problem field of Figure 4.

Since there is at least one inactive pixel inside of the set, one rectangle in the journey has a side length of three or more active pixels. As one "big" step in this rectangle can be divided into two "smaller" steps, every such journey can be transformed into one with an odd number of steps. By Lemma 4 this journey corresponds to a matrix of odd order with row sums and all column sums equal to 2. Hence, the problem matrix is unbalanced. □

Theorem 8. *A WRP problem field is unbalanced if and only if it contains one of the following pixel constellations*

(a) a 3×2 or a 2×3 active rectangle,

(b) two 2×2 active rectangles having exactly one pixel in common,

(c) a set of active pixels with inactive pixels inside.

Proof. If the problem field contains one of the listed constellations of active pixels, it follows directly from the Lemmas 5, 6, and 7 that this field is not balanced.

Next, let an arbitrary unbalanced problem field be given. We show that all unbalanced problem fields that do not contain the constellations (a) or (b) must contain a set of active pixels with inactive pixels inside. Choose any active pixel of the unbalanced field and assume that options (a) and (b) do not apply. Then, by assumption, the pixel is not contained in a rectangle of size 2×3. This leaves only rectangles of sizes 2×2, $1 \times n$, or $n \times 1$, where $n \in \mathbb{N}$.

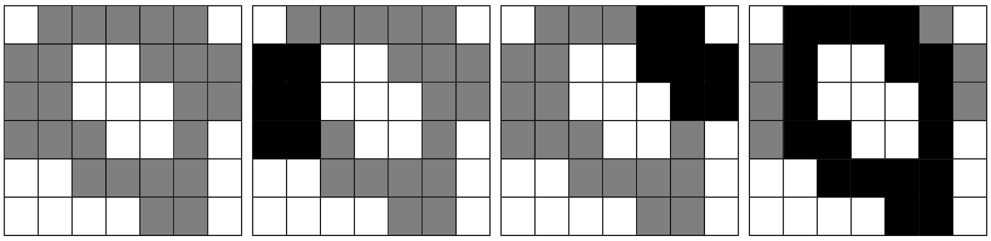


Figure 4: Sample field with the three different cases in Theorem 8.

If the pixel is contained in a 2×2 rectangle and this rectangle is connected to another rectangle of the same size, the two rectangles cannot have pixels in common as we have ruled out this constellation. Therefore, the only possible constellation is the shape given in Figure 5, i.e., two pixels of the rectangles are connected. By Lemma 4, it suffices to find a journey using this constellation of active pixels. In every such journey, a step from one of the two 2×2 rectangles to the other (such as the one highlighted in Figure 5) can only be used in one direction, so every journey has to return using another set of steps, outside of the shown constellation. Moreover, the two pixels prohibited in the figure have to be inactive by assumption. Therefore, one of the two remaining pixels must be inside the active set used by the journey.

The same argumentation holds for rectangles of sizes $1 \times n$ or $n \times 1$. □

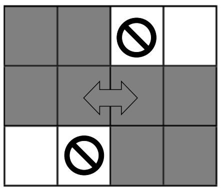


Figure 5: The constellation of pixels in the proof of Theorem 8. Two pixels are prohibited.

From a computational perspective, it is important to discuss the complexity of the criteria introduced above. Checking whether the lemmas used in Theorem 8 apply is relatively easy: The Lemmas 5 and 6 can be checked in $\mathcal{O}(pq)$ for a field of size $p \times q$. For testing Lemma 7, a depth-first search algorithm is needed, which has a runtime of $\mathcal{O}(pq)$.

Thus, the total runtime for the detection of the balancedness is in $\mathcal{O}(pq)$. Moreover, it is sufficient to check for circles with a length greater than 4.

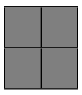
We continue this section with a refinement of the balancedness property for WRP.

Definition 9. A {0,1}-matrix is *totally balanced* if it does not contain a square submatrix of size at least three which has no identical columns and its row and column sums equal to two.

Lemma 10. *A problem field containing a 2×2 rectangle of active pixels is not totally balanced.*

Proof. Figure 6 shows a matrix representation of an active 2×2 rectangle. Clearly, the matrix is not totally balanced. □

In Lemma 4 we have observed that a square matrix of odd order with row and columns sums equal to two corresponds to an odd journey. In the same way, a square matrix of even order with row and column sums equal to two corresponds to an even journey. Therefore, we need to consider journeys of odd or even length when determining total balancedness. As before, we can fully characterize totally balanced instances using journeys.



$$\left(\begin{array}{ccccc|c} 1 & 1 & 1 & 0 & 0 & \\ 1 & 1 & 0 & 1 & 0 & \\ 1 & 0 & 1 & 0 & 1 & I_4 \\ 1 & 0 & 0 & 1 & 1 & \end{array}\right)$$

Figure 6: An active 2×2 rectangle and the corresponding matrix

Proposition 11. *A balanced WRP problem field is totally balanced if and only if it does not contain a 2×2 active rectangle.*

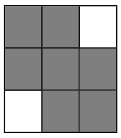
Proof. If a problem field is totally balanced, it does not contain a 2×2 active rectangle, by Lemma 10.

Since the problem field is balanced, it does not contain any of the constellations described in the Lemmas 5, 6, and 7. These constellations correspond to odd journeys. It remains to show that there is no even journey. However, with the three larger structures already ruled out, the only possible rectangular pixel structure which might contain an even journey is a 2×2 rectangle. Hence, the problem field is totally balanced. □

2.3 WRP-induced Hypergraphs

Following Berge [4], we introduced the notion of balanced matrices with the help of hypergraphs. The question arises of how the characterization of balanced WRP matrices in Theorem 8 fits into the original hypergraph perspective. In Figure 7 we see the WRP problem field that was already considered in Lemma 6, the problem field matrix, and one possible hypergraph of this matrix.

Using the definition of Berge directly, we would have to check every possible odd cycle and determine whether it contains an edge with at least three of its vertices. A circle that does not fulfill this property is drawn in bold lines in the hypergraph of Figure 7, so the hypergraph is not balanced. With the new characterization, it suffices to find the journey displayed in Figure 3.



$$\left(\begin{array}{cccccccccccc|c}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \\
1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & I_7 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 &
\end{array}\right)$$

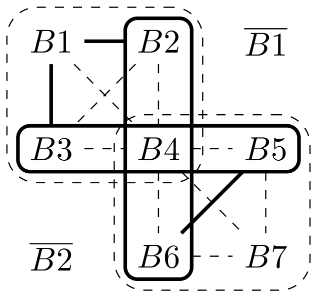


Figure 7: WRP problem field, corresponding matrix representation, and a corresponding hypergraph. Active pixels are labeled $B1, \ldots, B7$ and inactive pixels are labeled $\overline{B1}$, $\overline{B2}$.

3 Numerical Experiment

As observed in Section 2.1, the conditions of Theorem 8 are computationally easy to check. This is an interesting feature when numerically comparing an algorithm for balancedness detection with other integrality conditions. For this comparison, the implementation of a total unimodularity test by Walter and Truemper [24] is used. We call this benchmark *TU-test* in the following. The TU-test algorithm uses an implementation of [23]. Balancedness is checked with a simple Python implementation of Theorem 8. Comparing our specialized approach to the more general TU-test algorithm is, of course, unfair considering that TU-test is not focusing on WRP instances. However, to the best of our knowledge, TU-test is the only publicly available library for integrality testing via unimodularity or balancedness. According to the TU-test project webpage [24] an extension for balancedness detection is currently under development.

For the experiment, 100 random problem fields with the sizes 5×5, 10×10, 15×15, 20×20, and 25×25 were generated. We ran each instance five times and report averaged runtimes. Since the two algorithms test for different conditions, different results are to be expected. Because totally unimodular matrices are also balanced, our algorithm recognizes more integral problem instances than TU-test, which can be seen in Figure 8. For the random data, the figure shows up to four times more balanced instances than totally unimodular instances. These results emphasize that testing integrality via balancedness is preferable to total unimodularity because more instances can be detected this way. In addition, the balancedness detection is also faster in the WRP problem class, which is, however, not guaranteed for general integer programming problems.

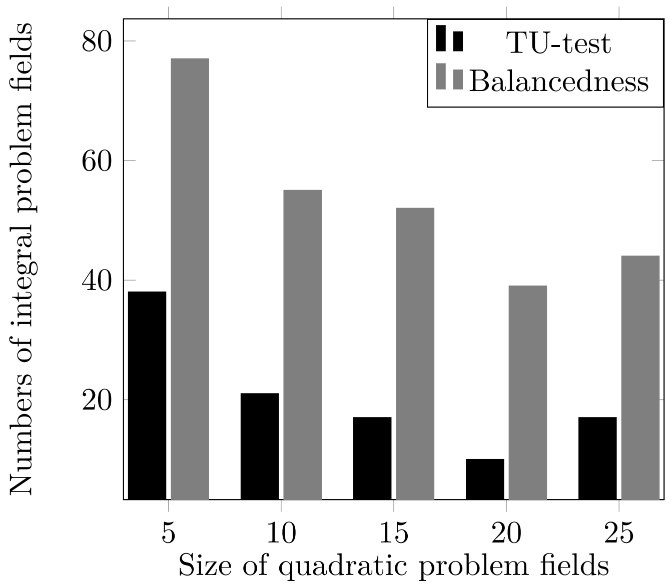


Figure 8: Integral instances detected by the two algorithms.

As expected, there are pronounced differences in the runtimes reported in Figure 9. The balancedness test clearly outperforms the TU benchmark in terms of runtime. Moreover, Figure 9 illustrates that the increase in runtime for the balancedness test is much smaller than for TU-test when the problem field size is raised. The difference in experimental runtimes can, of course, be traced back to different worst-complexities of the two algorithms. The algorithm for balancedness detection is tailored specifically to the WRP problem fields while the TU-test benchmark checks general matrices for total unimodularity. The balancedness recognition algorithm has a worst-case runtime of $\mathcal{O}(pq)$, while the TU-test runs in $\mathcal{O}((p^2q^2)^5)$ worst-case time, as reported in [24].

Table 1 summarizes the number of detected integral problem fields by both tests, the cumulative runtimes of the algorithms, and the cumulative runtime of a basic `gurobipy` implementation of Formulation (1) for reference.

4 Conclusion and Possible Extensions

We have introduced WRP as a geometric optimization problem with close connections to other combinatorial problems of the partitioning type. Starting from a straightforward

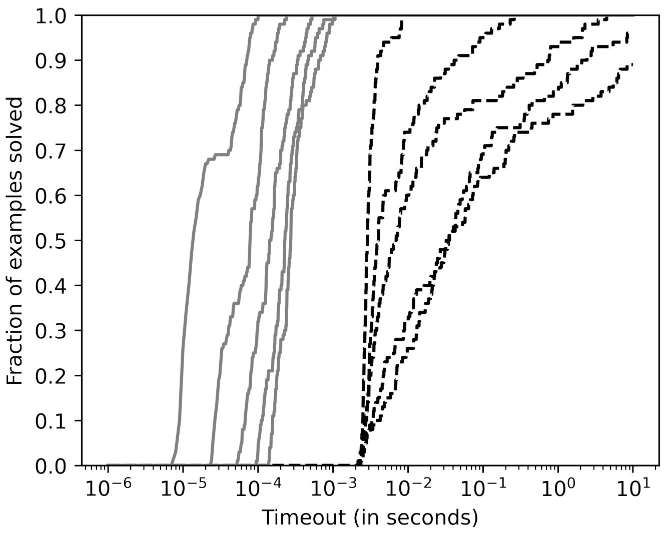


Figure 9: Number of solved instances using the TU-test benchmark algorithm (dashed lines) and balancedness detection via Theorem 8 (solid gray lines). For each algorithm, the five lines show instances of sizes 5×5, 10×10, 15×15, 20×20, and 25×25 from left to right.

IP formulation, we adapted the concept of balancedness to detect the integrality of WRP instances and argued why this approach is more suitable than checking for total unimodularity here. The results proved in Section 2.1 characterize all balanced WRP instances. Based on our numerical study, integrality detection is possible in practical computations.

Several aspects of the problem setting allow for future research. Above, we discussed the problem with a general weight function. Assuming that the weight function is, e.g., monotone or submodular seems reasonable and may lead to specialized solution algorithms or approximation schemes as in [1]. Furthermore, we have limited our discussion to rectangles. Allowing more general shapes such as polyominoes might be an interesting extension of the WRP. However, the concept of journeys used in our proofs does not carry over to more general shapes in an obvious way.

Finally, we conjecture that more general pixel constellations than those used in Theorem 8 can be used to characterize WRP instances with a perfect system matrix, see [19]. Since every balanced matrix is also perfect, these constellations have to be supersets of those described in Theorem 8.

Acknowledgements The second author acknowledges the support of the Research Training Group 2297 *Mathematical Complexity Reduction* (MathCoRe), funded by Deutsche Forschungsgemeinschaft (DFG), grant number 314838170.

Data availability The datasets generated and analyzed in Section 3 are available in the

Table 1: Summary of the problem field sizes of the tested instances, number of recognized instances by the TU-test benchmark (TU) and balancedness detection algorithm (bal), total runtimes (time TU) and (time bal) of the two algorithms, and the runtime (time IP) when solving formulation (1) directly using `gurobipy`. All runtimes in seconds.

size	TU	bal	time TU	time bal	time IP
5×5	38	77	0.33	0.003	0.25
10×10	21	55	1.90	0.008	1.56
15×15	17	52	23.45	0.018	5.92
20×20	10	39	175.78	0.027	15.70
25×25	17	44	1431.31	0.034	49.10

repository `https://github.com/ulf-friedrich/WeightedRectanglesPartition`.

References

[1] Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles, 2013.

[2] Aristocrat Technologies Australia. Lucky fortune link – da sheng yeah, 2021. `www.aristocrat.com/apac/games/lucky-fortune-link-da-sheng-yeah`.

[3] Egon Balas and Manfred W. Padberg. Set partitioning: A survey. *SIAM Review*, 18:710–760, 1976.

[4] Claude Berge. Balanced matrices. *Mathematical Programming*, 1:19–31, 1972.

[5] Andreas Bortfeldt. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, 172(3):814–837, 2006.

[6] Michele Conforti, Gérard Cornuéjols, and Mendu Rammohan Rao. Decomposition of balanced matrices. *Journal of Combinatorial Theory, Series B*, 77(2):292–406, 1999.

[7] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Springer, 2014.

[8] D. R. Fulkerson, A. J. Hoffman, and Rosa Oppenheim. On balanced matrices. In M. L. Balinski, editor, *Pivoting and Extension: In honor of A.W. Tucker*, pages 120–132. Springer Berlin Heidelberg, 1974.

[9] Paul C Gilmore and Ralph E Gomory. Multistage cutting stock problems of two and more dimensions. *Operations Research*, 13(1):94–120, 1965.

[10] Alan J Hoffman and Joseph Bernard Kruskal. Integral boundary points of convex polyhedra. *Linear Inequalities and Related Systems*, pages 223–246, 1956.

[11] Manuel Iori, Vinícius Loti de Lima, Silvano Martello, and Michele Monaci. 2DPackLib: a two-dimensional cutting and packing library. *Optimization Letters*, 16(2):471–480, 2022.

[12] Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000.

[13] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, 1997.

[14] WT Liou, JJ-M Tan, and Richard CT Lee. Minimum rectangular partition problem for simple rectilinear polygons. *IEEE transactions on computer-aided design of integrated circuits and systems*, 9(7):720–733, 1990.

[15] Andrea Lodi, Silvano Martello, and Michele Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

[16] Andrea Lodi and Michele Monaci. Integer linear programming models for 2-staged two-dimensional knapsack problems. *Mathematical Programming*, 94(2):257–278, 2003.

[17] Silvano Martello, Michele Monaci, and Daniele Vigo. An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15(3):310–319, 2003.

[18] Surendra Nahar and Sartaj Sahni. Fast algorithm for polygon decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 7(4):473–483, 1988.

[19] Manfred W Padberg. Perfect zero-one matrices. *Mathematical Programming*, 6(1):180–196, 1974.

[20] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1998.

[21] Valeriu Soltan and Alexei Gorpinevich. Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles. *Discrete & Computational Geometry*, 9(1):57–79, 1993.

[22] Tomáš Suk, Cyril Höschl IV, and Jan Flusser. Decomposition of binary images—a survey and comparison. *Pattern Recognition*, 45(12):4279–4291, 2012.

[23] Klaus Truemper. A decomposition theory for matroids. v. testing of matrix total unimodularity. *Journal of Combinatorial Theory, Series B*, 49(2):241–281, 1990.

[24] Matthias Walter and Klaus Truemper. Implementation of a unimodularity test. *Mathematical Programming Computation*, 5(1):57–73, 2013.