Recognizing Integrality of Weighted Rectangles Partitions

Paul Deuker* Ulf Friedrich†

November 25, 2024

Given a grid of active and inactive pixels, the weighted rectangles partitioning (WRP) problem is to find a maximum-weight partition of the active pixels into rectangles. WRP is formulated as an integer programming problem and instances with an integral relaxation polyhedron are characterized by a balanced problem matrix. A complete characterization of these balanced instances is proved. In addition, computational results on balancedness recognition and on solving WRP are presented.

Keywords Partition Problem, Rectangles, Integer Programming, Balanced Matrix

Mathematics Subject Classification (2020) 90C10, 90C05, 05C50

1 Problem Description

We begin by formally defining the two-dimensional Weighted Rectangles Partitioning problem. For $p, q \in \mathbb{N}$ we consider a rectangular *problem field* $R_{p,q}$ which is sub-divided into $p \times q$ 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 are active sets consisting of 1×2 active pixels and 1×3 active pixels. An active set that consist of $k \leq p$ and $l \leq q$ consecutive pixels is called an *active rectangle*. A problem field $R_{p,q}$ is fully defined by a matrix $\boldsymbol{Q} \in \{0,1\}^{p \times q}$, where each active pixel is represented by a one and each inactive pixel is represented by a zero.

*Tilburg University, Department of Econometrics and Operations Research, P.Deuker@tilburguniversity.edu

†Otto von Guericke University Magdeburg, Faculty of Mathematics, ulf.friedrich@ovgu.de

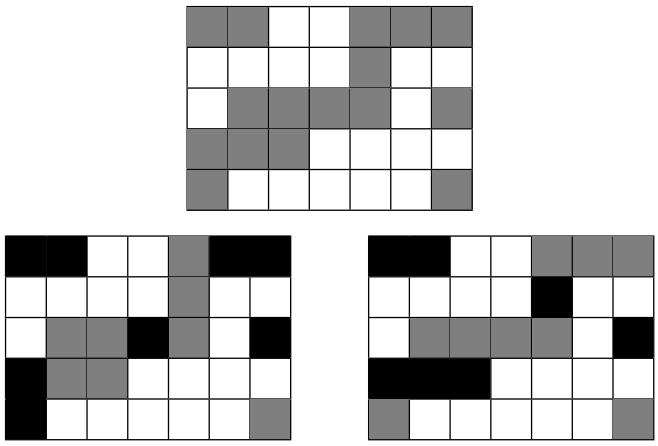


Figure 1: Two different rectangle partitions of a 5×7 problem field. The active pixels are displayed in gray in the upper picture, two sample partitions are displayed in black and gray below.

Let $S_{p,q}$ be the set of all $k \times l$ rectangles that consist of $k \leq p$ and $l \leq q$ consecutive pixels, i.e., all sub-rectangles built on the pixels of $R_{p,q}$. Each $s \in S_{p,q}$ is assigned a weight $w(s) \in \mathbb{R}$.

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

In Definition 1, *partition* means that all active pixels must be covered by rectangles without covering any inactive pixels, and *disjoint* means that no two rectangles contain the same pixel. The problem is always feasible, as a trivial solution uses one 1×1 rectangle for every active pixel. Two non-trivial solutions are shown in Figure 1. Both solutions consist of eight disjoint rectangles, but they can have different total weights. Depending on the weights, it may be beneficial to split or merge rectangles to obtain a higher total weight. For example, splitting a 2×2 rectangle with a weight of ten into two 2×1 rectangles with weights of six each will result in a higher total weight.

1.1 Motivation

Based on Definition 1, WRP is one of the countless variants of partition problems studied in the optimization literature. As discussed in Section 1.2, partition problems have various applications and their study led to numerous theoretical results over the years. Indeed, in their survey on the topic, Balas and Padberg motivate studying partition problems by stating that *"Among all special structures in (pure) integer programming, there are three which have the most wide-spread applications: set partitioning, set covering and the traveling salesman (or minimum length Hamiltonian cycle) problem; and if we*

were to rank the three, set partitioning would be a likely candidate for number one," see [2, page 712]. We agree wholeheartedly on the importance of partition problems for the theory of integer programming, which we aim to extend by introducing a new variant in the form of WRP.

In addition, the decision versions of many partition problems are known to be NP-complete [10] and WRP is NP-complete as well [10, Problem SR25]. The original NP-completeness proof [20] remains unpublished, as explained in [14]. Specifically, if we consider a decision variant of WRP that asks whether there exists a solution with fewer than K rectangles for a given $K \in \mathbb{N}$, this problem is NP-complete. Our main contribution is the characterization of balanced, and thus polynomially solvable, instances of WRP in Section 2.2. Describing these easy instances within the set of all WRP instances is a theoretical contribution that can hopefully be helpful for the solution theory of more general integer programs as well.

For the characterization of WRP instances we apply results on balanced matrices. The relevance of balancedness in the theory of integer programming has been emphasized by awarding the 2000 Delbert Ray Fulkerson Prize [13] to the authors of [7]. Therefore, our motivation for studying balancedness of WRP is, on the one hand, to establish additional theoretic insights for balanced integer programs and, on the other hand, provide algorithmic tools for their analysis.

Moreover, applications of partition problems in areas such as scheduling, logistics, communication theory or political districting are abundant, see the list in [2], and applications of our variant of the partition problem are discussed in Section 1.2 below. Hence, studying WRP is relevant not only from a theoretical, but also from an applied perspective.

1.2 Related Problems and Applications

We observe two major directions for comparing WRP with other problems discussed in the optimization literature: First, focusing the geometric aspect of WRP, we discuss research on rectangular decomposition and related problems. Second, emphasizing the partitioning feature, we connect WRP to the literature on (two-dimensional) partitioning and packing.

In the literature on rectangular decomposition, e.g., [17], [21], [24], the problem of partitioning a given rectangular shape with or without holes into a minimum number of smaller rectangles is studied. This has been extended to more general shapes, as surveyed in [25]. We can reduce the decomposition problem to WRP by setting the weights of all rectangles to -1. Definition 1 introduces WRP as a weight-maximization problem similar to the maximum weight independent set of rectangles (MWISR) problem discussed in [5] and [6]. Here, the task is to select a weight-maximal subset of disjoint rectangles from a given set of feasible rectangles. We can interpret WRP as a special case of MWISR with non-negative weights, where the set of feasible rectangles is defined by the active pixels in the field. In addition, the authors of [6] discuss how MWISR, and therefore also WRP, is related to graph-coloring and other combinatorial optimization problems. However, the pixel-based setting of WRP is somewhat different from the general geometric perspective

of most rectangle decomposition algorithms.

A partitioning problem similar to WRP is studied from an approximation perspective in [1]. The authors provide a summary of approximation algorithms and present a quasi-polynomial approximation method based on dynamic programming.

Turning to the partitioning aspect of WRP, we observe that WRP can be classified as a set partitioning problem [8] where the active pixels are partitioned into rectangles. We can also relate WRP to a two-dimensional cutting stock problem with holes [11], [15]. While in the cutting stock problem not all rectangle types may be used, the WRP problem field has inactive pixels that must not be occupied. Furthermore, we can interpret the set of active pixels as empty space in which (weighted) rectangles must be packed. Thus, WRP is also related to the many two-dimensional packing [18] or knapsack problems [19] discussed in the optimization literature.

In theoretical computer science, rectangle partitions are important for studying the communication complexity of a protocol since they define lower bounds on the deterministic communication complexity [16]. However, the notion of rectangles in the context of communication complexity is more general than our definition of active rectangles above, see [16, Definition 1.12]. Therefore, WRP cannot be applied to compute lower bounds on the communication complexity in an obvious way.

Our study of WRP is originally inspired by an unusual application in a slot machine game: The game has a playing field of size 5×7 pixels, in which random pixels are active. The goal of the game is to connect adjacent active pixels to rectangles with different weights in order to maximize the sum of all weights.

We base our further analysis of WRP on the integer programming formulation of the problem given in Section 2.1. To identify integral instances, we use the concept of balancedness that has been introduced as a generalization of bipartite hypergraphs in [3]. Balanced matrices, i.e., the incidence matrices of balanced hypergraphs, have many interesting properties, most notably the fact that the polyhedron defined by a balanced matrix is integral for all integer-valued right-hand sides. An introduction to balanced matrices and their properties is given in [4].

1.3 Structure of this Work

In Section 2.1, we formulate WRP as an integer programming problem. Based on this formulation, we investigate integral instances of WRP by checking the balancedness of the system matrix. Our main result is the complete characterization of balanced instances stated in Theorem 1. In the computational experiment of Section 3, we discuss advantages of our specialized method for checking balancedness over two benchmark algorithms from the software package [27]. Finally, we assess our findings in Section 4 and suggest possible directions for future research.

The data generated for the tests in Section 3 and implementations of the presented algorithms are available in the following repository:
`https://github.com/ulf-friedrich/WeightedRectanglesPartition`.

2 Solving WRP

2.1 IP Formulation

To analyze the WRP problem, we first introduce an integer programming (IP) formulation. Let B denote the set of active pixels, and let $R = R(B) \subseteq S_{p,q}$ be the set of rectangles that only contain active pixels $b \in B$. Only the rectangles in R can be used in a partition of active pixels.

We write $\boldsymbol{A} = (a_{b,r})$ to represent the incidence matrix of all rectangles in R, where the dependency on B is ignored in the notation. The columns of $\boldsymbol{A}$ are indexed by the rectangles $r \in R$, and the rows of $\boldsymbol{A}$ are indexed by the active pixels $b \in B$, where $a_{b,r} = 1$ if and only if $b \in r$. We introduce a variable x_r for all $r \in R$, and $w_r = w(r) \in \mathbb{R}$ represents the weight of the rectangle r.

Using the above notation, we formulate WRP as an IP:

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

This formulation resembles a typical set partitioning problem [2]. In this case, the size of the formulation depends on $|R|$. By identifying each rectangle $r \in R$ by its upper left pixel and its size, all possible rectangles r can be determined: There are at most pq active pixels that can be the upper left pixel of a rectangle. In addition, each of these rectangles can have a size of at most p consecutive pixels in height (rows) and at most q consecutive pixels in length (columns), giving pq possibilities for the size of the rectangle. Therefore, $|R| \leq p^2 q^2$ applies.

2.2 Integral WRP Instances

Solving 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. One way to verify this property is through the Hoffman-Kruskal Theorem [12], which states that the total unimodularity of the system matrix $\boldsymbol{A}$ implies that Problem (1) is integral. Alternatively, we can apply results on balanced matrices developed in [4, 7, 9] to obtain a sufficient condition for the integrality of the polyhedron.

Definition 2 (21.5 in [23])**.** *A balanced matrix is a {0,1} matrix that does not contain a square submatrix of odd order with all row sums and all column sums equal to two. An unbalanced matrix is a matrix that is not balanced. A WRP problem field is balanced if its problem matrix* $\boldsymbol{A}$ *in Formulation* (1) *is balanced.*

The polyhedron defined by the linear relaxation of Problem (1) is integral if the system matrix is balanced, as stated in [23, Theorem 21.7]. We argue that checking for balancedness is more efficient for WRP than testing for total unimodularity both

theoretically, see Section 2.3, and computationally, based on our experiment discussed in Section 3. Furthermore, finding balanced instances provides a larger set of integral instances compared to total unimodularity because every totally unimodular matrix is balanced [23, Section 21.5].

Our next aim is to find a characterization of the balanced WRP instances among all WRP instances. We use *journeys* as defined in Definition 3 to link pixels in the WRP problem field to submatrices of $\boldsymbol{A}$ in Formulation (1).

Definition 3. *A step in a WRP problem field is an ordered pair of two different pixels $(b_1, b_2) \in B \times B$ which are both contained in the same active rectangle $r \in R$. Every step corresponds to two entries $a_{r,b_1} = 1$ and $a_{r,b_2} = 1$ in the problem matrix $\boldsymbol{A}$. A journey in a WRP problem field is a finite set of steps, such that the corresponding matrix entries form a square submatrix with all row sums and all column sums equal to two. A journey is odd if it has an odd number of steps.*

The submatrix property in Definition 3 means that each pixel of a journey is contained in exactly two steps. The definition states that every odd journey corresponds to a square submatrix of odd order with all row sums and all column sums equal to two. Lemma 1 shows that the reverse is also true.

Lemma 1. *For every square submatrix $\boldsymbol{M}$ of $\boldsymbol{A}$ that has odd order with row sums and column sums equal to two, there is an odd journey in the WRP problem field such that the pixels of the journey correspond to the submatrix $\boldsymbol{M}$.*

Proof. The columns of $\boldsymbol{M}$ define rectangles in the problem field and the rows of $\boldsymbol{M}$ define pixels of the problem field. Each column defines a step because the column sum is two. Since $\boldsymbol{M}$ has row sums and column sums equal to two, it defines a journey. Moreover, this journey is odd because $\boldsymbol{M}$ is of odd order. □

Definition 3 and Lemma 1 directly imply the following Proposition 1.

Proposition 1. *A WRP problem field is balanced if and only if it does not contain an odd journey.*

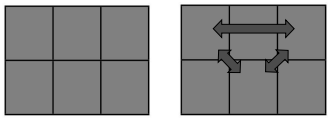
Next, we construct three types of problem fields that are unbalanced. These examples are the key observations for the complete characterization of balanced WRP fields in Theorem 1.

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

Proof. We only give the proof for 2×3 rectangles because the transpose of a balanced matrix is also balanced. Given a 2×3 rectangle of active pixels as shown on the left in Figure 2, consider the journey indicated by the arrows in the middle of Figure 2. We prove that this journey leads to an unbalanced submatrix.

A matrix representation of the problem field is given on the right of Figure 2. The matrix $\boldsymbol{I}_6$ is the identity matrix in 6 dimensions, containing the 6 one-by-one rectangles.

The first highlighted column of the matrix corresponds to the horizontal step of length three, the second highlighted column corresponds to the step in the left 2×2 rectangle, and the third highlighted column corresponds to the step in the right 2×2 rectangle. Thus, the gray matrix entries form a 3×3 submatrix which corresponds to the odd journey shown in the figure. The WRP problem field is unbalanced by Proposition 1. □



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

Figure 2: A set of 2×3 active pixels, a journey using the pixels, and the corresponding matrix on the right. The matrix entries highlighted in gray are in three different columns which define the three steps.

In the same manner, we can prove that all problem fields that contain the left or middle set of active pixels in Figure 3 are unbalanced.

Lemma 3. *A problem field containing two* 2×2 *active rectangles, whose intersection is exactly one pixel, is unbalanced.*

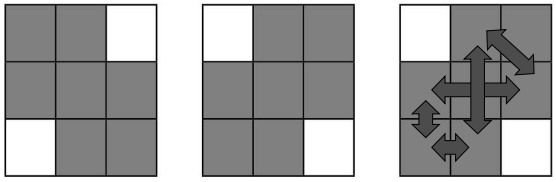


Figure 3: Left and middle: Two unbalanced combinations of active pixels. Right: Illustration of an odd journey with the following five steps: 2×1 rectangle in first column, 1×2 rectangle in third row, 3×1 rectangle in second column, 2×2 rectangle in first and second row, 1×3 rectangle in second row.

Proof. The journey with the five steps depicted in the right picture of Figure 3 implies that the active set in the middle picture is unbalanced. The active set on the left is obtained by transposing the WRP problem field. □

Next, we examine locked pixels as the third pattern implying unbalancedness. In Definition 4, we consider the eight *directions* relative to a pixel's coefficients (i, j) in the binary matrix representation $\boldsymbol{Q}$ of the problem field $R_{p,q}$, namely the two horizontal directions (pixels in the same row i, but in a different column $l \neq j$, smaller or larger), two vertical directions (pixels in the same column j, but in a different row $k \neq i$, smaller

or larger), and all four diagonal directions (pixels in a different row $k \neq i$ and a different column $l \neq j$, both smaller or larger).

Definition 4. *An inactive pixel in a WRP problem field is called locked if there exist active pixels in all eight directions of the problem field that are in a single journey. A set of pixels is called locked if it contains only locked pixels.*

The active pixels in the eight direction do not have to be adjacent to the locked pixel. A set of seven locked pixels is shown in the left picture of Figure 4. By definition, a locked pixel cannot be in the first row, in the first column, in the last row or in the last column of a WRP problem field.

Lemma 4. *A problem field containing a non-empty set of locked pixels is unbalanced.*

Proof. Consider a non-empty set of locked pixels and the journey in Definition 4. If this journey is odd, the problem field is unbalanced by Proposition 1.

Assume that the journey is even. Because the set of locked pixels is non-empty, one rectangle in the journey has a side length of three or more active pixels. We can now either substitute one step in this rectangle (of size at least three) by two smaller steps or merge two smaller steps to one larger step in this rectangle. This way, we either decrease or increase the number of steps by one. The resulting journey is always odd and the result follows from Proposition 1. □

So far, we have discussed three examples of unbalanced problem fields in the three lemmas above. Peculiarly, the pixel sets discussed in the lemmas are also necessary for the characterization of unbalanced WRP problem fields. Hence, Theorem 1 characterizes all (un)balanced WRP problem fields.

Theorem 1. *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 whose intersection is exactly one pixel,*

(c) a non-empty set of locked pixels.

Proof. If the problem field contains any of the listed constellations of active pixels, it follows directly from the Lemmas 2, 3, and 4 that this field is unbalanced.

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 non-empty set of locked pixels. Choose any active pixel of the unbalanced field and assume that this pixel is not in an active set of type (a) or (b). 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 $n \in \mathbb{N}$.

If the pixel is contained in an active 2×2 rectangle, it cannot be contained in another 2×2 rectangle because (b) does not hold. Therefore, the only possible (up to transposition)

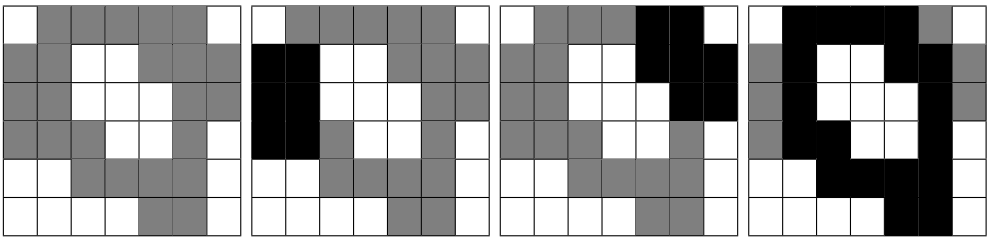


Figure 4: Sample field with the three different cases in Theorem 1.

constellation of two active 2×2 rectangles in the WRP problem field is the shape given in Figure 5, i.e., two active pixels of the rectangles are adjacent in the WRP problem field.

By Lemma 1, there exists an odd journey on the set 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 occur once. Thus, every journey has to use another set of steps, outside of the shown set of pixels. Moreover, the two pixels marked in the figure have to be inactive as (a) does not hold. Therefore, one of the two remaining inactive pixels must by locked. The argumentation also holds for pixels contained in rectangles of sizes $1 \times n$ or $n \times 1$ and the proof is complete. □

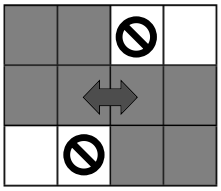


Figure 5: The constellation of pixels in the proof of Theorem 1. The two marked pixels must be inactive.

2.3 Complexity of the Balancedness Test

From a computational perspective, it is important to discuss the complexity of the criteria introduced above. Checking whether the three lemmas for Theorem 1 apply is straightforward. The Lemmas 2 and 3 can be checked in $\mathcal{O}(pq)$ for a field of size $p \times q$. For testing Lemma 4 we can use an algorithm which is similar to a deep-first search algorithm and has a running time in $\mathcal{O}(pq)$: We start from any active pixel that has a at least one non-active adjacent pixel. We then check the adjacent pixels in the two horizontal directions and the two vertical directions. If they are active, they are also taken into account. We repeat this process until we reach the same active pixel twice, or until there are no more pixels to check. Thus, the total, theoretic worst-case runtime for recognizing balancedness via Theorem 1 is in $\mathcal{O}(pq)$.

2.4 Valid Inequalities for Non-Balanced Instances

So far, we have been concerned with recognizing balanced instances of WRP. Next, we focus on *unbalanced*, thus potentially non-integral, instances. The computational performance of solution algorithms for non-integral instances of IP generally depends on the problem formulation given by valid inequalities for the linear relaxation. Adding additional valid inequalities (or cutting planes) to a formulation excludes non-integral points from the linear relaxation and makes the IP formulation tighter [23]. We demonstrate how valid inequalities can be deduced with the help of Lemma 2. In the same way, the pixel structures of Lemma 3 and Lemma 4 can be used for defining valid inequalities.

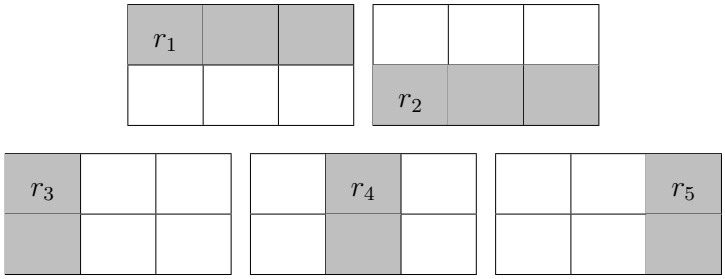


Figure 6: The rectangles that define the valid inequalities (2) and (3).

In Figure 6 five rectangles are highlighted, all of which use subsets of pixels of a 2x3 rectangle. Since WRP is a partition problem by Definition 1, none of the active pixels can be in more than one rectangle used in the solution. Therefore, for each active 2x3 rectangle in the problem field, we can add the following valid inequalities to the IP formulation (1) of WRP

$$3x_{r_1} + x_{r_3} + x_{r_4} + x_{r_5} \leq 3, \tag{2}$$

and

$$3x_{r_2} + x_{r_3} + x_{r_4} + x_{r_5} \leq 3. \tag{3}$$

The valid inequalities (2) and (3) express that either one of two 1x3 rectangles r_1, r_2 can be used in the solution or a combination of the three 2x1 rectangles r_3, r_4, r_5 can be used. On the one hand, if r_1 or r_2 are used in the solution, then $x_{r_1} = 1$ or $x_{r_2} = 1$, respectively, which implies that $x_{r_3} = x_{r_4} = x_{r_5} = 0$. On the other hand, if r_1 and r_2 are not used in the solution, then $x_{r_1} = 0$ and $x_{r_2} = 0$, and therefore the right-hand sides of value three in (2) and (3) do not impose any restriction on the remaining variables x_{r_3}, x_{r_4}, and x_{r_5}.

Using the same reasoning, the two squares r_6 and r_7 in Figure 7 define the valid inequality

$$x_{r_6} + x_{r_7} \leq 1, \tag{4}$$

for each active 2x3 rectangle in the problem field. Clearly, at most one of two 2x2 squares can be used in a solution of WRP.

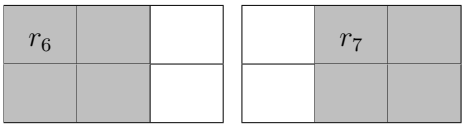


Figure 7: The squares r_6 and r_7 define the valid inequalities (4).

We have tested the computational performance when solving improved formulations of Problem (1), in which the valid inequalities (2), (3), and (4) have been added to the formulation for all 2x3 rectangles of the problem field. However, we cannot report a significant change in the solution times for the improved formulations in this proof-of-concept implementation. In view of Theorem 1, it is also not to be expected that our naive test leads to a large performance speed-up because only a subset of the possible valid inequalities is added to the formulation. A structured computational analysis of valid inequalities for WRP can potentially yield more insights, but diverts too far from the focus of our presentation. The code and computational results of our tests are available via the supplementary software repository.

3 Numerical Experiment

As observed in Section 2.2, the conditions of Theorem 1 can be checked efficiently. This is an interesting feature when comparing an algorithm that recognizes balancedness in WRP with other integrality conditions. We compare a balancedness test B-WRP based on the results of Section 2.2 to the implementation of a total unimodularity test and a recent implementation of a general balancedness test, both from [27]. We call these benchmarks TU and B-gen in the following. The TU benchmark uses an implementation of [26] for testing total unimodularity. The implementation of the general balancedness test B-gen is based on a combination of enumerating subsets of rows and columns and a polynomial-time algorithm [28].

The B-WRP algorithm uses a simple python implementation to check the conditions of Theorem 1. Comparing this specialized approach to the general benchmark algorithms is, of course, unfair considering that both are not focusing WRP instances. However, to the best of our knowledge, the package [27] provides the only publicly available implementations for integrality testing via unimodularity or balancedness.

Testing for integrality alone does not solve the IP formulation for WRP. Therefore, we also compare solution times for balanced and unbalanced random instances. While we do not claim that our results improve how integer programs are solved in practice, we emphasize that algorithms for balancedness testing are crucial for the analysis of IP. Based on balancedness testing, we explain why certain instances can be solved faster than others. Moreover, the random balanced instances used in the runtime comparisons have been found with the B-WRP algorithm, which can likewise be applied to create test settings for other studies.

For the experiment, 100 random problem fields with the sizes 5×5, 10×10, 15×15,

20×20, and 25×25 were generated. Each instance is solved five times on a desktop computer and average runtimes are reported. There is a timeout of 10 seconds for each problem run.

Since the three algorithms test for different conditions, different results are to be expected. Because totally unimodular matrices are also balanced, the two balancedness algorithms B-WRP and B-gen recognize more integral problem instances than TU, as summarized in Figure 8. For the random data, the figure shows up to 2.5 times more balanced instances than totally unimodular instances. This observation emphasizes that testing integrality via balancedness is preferable to total unimodularity because more instances can be recognized this way.

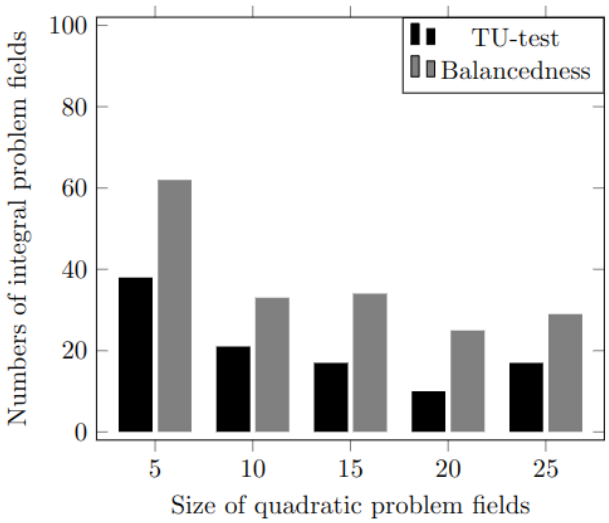


Figure 8: Number of integral instances found by the three algorithms.

As expected, there are pronounced differences in the runtimes reported in Figure 9 and Table 1. The specialized balancedness test B-WRP clearly outperforms both the TU and B-gen algorithm in terms of runtime. Moreover, Figure 9 illustrates that for larger problem fields the runtime of the balancedness test B-WRP increases slower than the runtime for the two benchmark algorithms. The observed differences in experimental runtimes can be traced back to the different worst-case complexities of the three algorithms. The balancedness recognition algorithm B-WRP has a worst-case runtime in $\mathcal{O}(pq)$, see Section 2.3, while the TU algorithm runs in $\mathcal{O}((p^2q^2)^5)$ worst-case time and the general balancedness recognition algorithm B-gen in $\mathcal{O}((p^2q^2)^9)$, as reported in [27]. However, the algorithm B-WRP for balancedness testing is tailored specifically to WRP problem fields while the two benchmarks check general matrices for total unimodularity or balancedness, respectively, and hence solve a more general problem.

Turning to the practical numerical solution of WRP via Formulation (1), we can report in Table 1 that for all sets of instances in our study, the average solution times for the balanced instances (t-IPb) are shorter than the averages solution times of the unbalanced

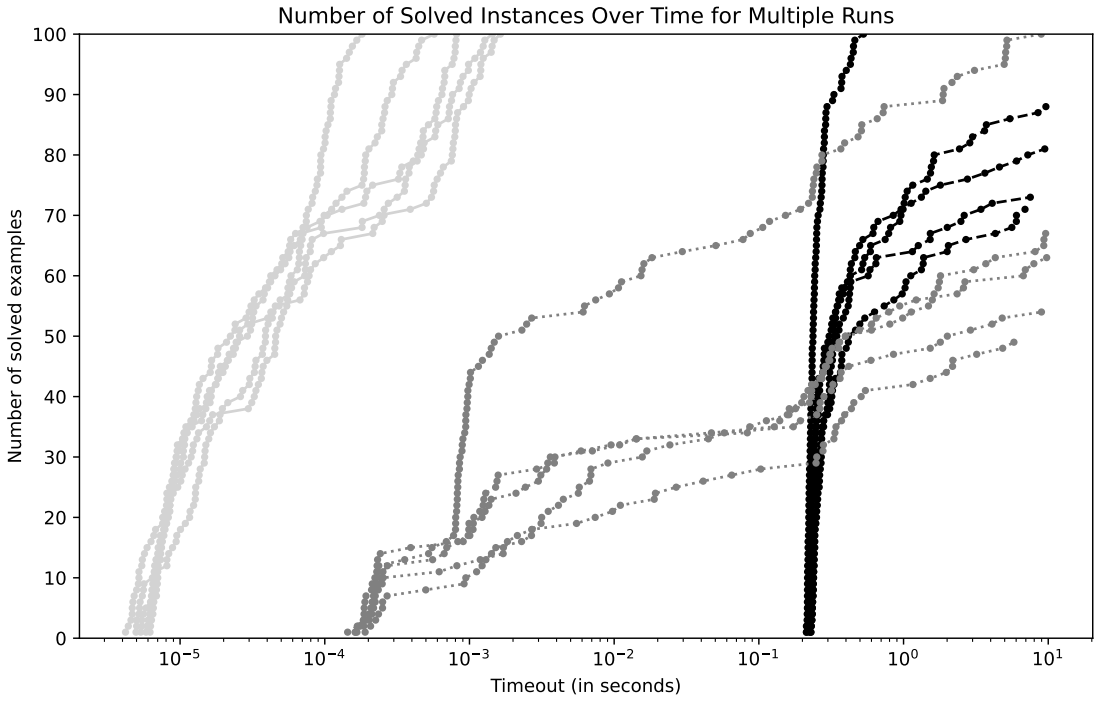


Figure 9: Number of solved instances using the TU algorithm (black lines), the general balancedness algorithm B-gen (dark gray lines), and the B-WRP balancedness test via Theorem 1 (light gray lines). For each algorithm, the five lines show instances of sizes 5×5, 10×10, 15×15, 20×20, and 25×25 from left to right.

instances (t-IPub). In addition, the runtime difference is more pronounced for the larger problem fields with ratios increasing from 1.3 for the instances of size 5×5 to 22.9 for the largest problems of size 25×25. This means that solving the largest balanced instances via Formulation (1) is on average more than 22 times faster than solving the unbalanced instances of the same size. We can conclude that the strong theoretical property of balanced WRP instances implies significant runtime improvements for the solution of WRP as well.

From a practical perspective, it is not necessary to run an integrality test first before applying an IP solver to a given WRP instance. In fact, this observation is true for any problem with an integral IP formulation that can be solved via its LP relaxation. In this regard, our computational study is in-line with the typical experiments used to analyze IP algorithms.

Table 1: Summary of the computational results, columns from left to right: field sizes of the tested instances (size), number of totally unimodular instances (TU), number of balanced instances (bal), total runtime for the TU benchmark (t-TU), total runtime for the specialized balancedness test (t-B-WRP), total runtime for the general balancedness test (t-B-gen), average runtime when solving formulation (1) for the balanced instances (t-IPb), average runtime when solving formulation (1) for the unbalanced instances (t-IPub), and the ratio of the solution times t-IP-unb/t-IP-bal (ratio). All times in seconds, all IP formulations solved with `gurobipy` and a timeout of 10 seconds. Times marked with an asterisk contain instances that have been stopped by the timeout.

size	TU	bal	t-TU	t-B-WRP	t-B-gen	t-IPb	t-IPub	ratio
5×5	38	62	26	0.005	55	0.11	0.14	1.3
10×10	21	33	195*	0.010	395*	0.09	0.21	2.3
15×15	17	34	252*	0.019	418*	0.15	0.74	4.9
20×20	10	25	352*	0.022	538*	0.13	2.33	17.9
25×25	17	29	317*	0.030	491*	0.27	6.18	22.9

4 Conclusion and Possible Extensions

Starting from the IP formulation (1) of WRP, we defined balanced WRP fields to characterize integral instances. In addition, we argued why this approach is more suitable than checking for total unimodularity. The results proved in Section 2.2 characterize all balanced WRP problem fields. The computational experiment in Section 3 shows that balancedness testing is possible and fast in practical computations. Solutions times for the IP formulation decrease significantly for balanced instances.

Several aspects of the problem setting motivate future research. We discussed WRP with a general weight function, but 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 WRP. However, the concept of journeys used in our proofs does not carry over to more general shapes in an obvious way. Further research on valid inequalities for WRP, such as those described in Section 2.4, can potentially reveal more insights on fast solution strategies for non-integral instances of WRP.

Finally, we conjecture that more general pixel sets than those used in Theorem 1 can be used to characterize WRP instances with a perfect system matrix, see [22]. This characterization would yield a more general criterion for integrality testing. Since every balanced matrix is also perfect, these pixel sets have to be supersets of those described in Theorem 1.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments that greatly helped to improve the manuscript. Matthias Walter kindly supported the use of his software [27] in the numerical experiments. The second author acknowledges the support of the Research Training Group 2297 *Mathematical Complexity Reduction* (MathCoRe), funded by Deutsche Forschungsgemeinschaft (DFG), grant number 314838170.

References

[1] Adamaszek, A., Wiese, A.: Approximation schemes for maximum weight independent set of rectangles. In: 2013 IEEE 54th annual symposium on foundations of computer science, pp. 400–409. IEEE (2013)

[2] Balas, E., Padberg, M.W.: Set partitioning: A survey. SIAM Review **18**, 710–760 (1976)

[3] Berge, C.: Sur certains hypergraphes généralisant les graphes bipartites. In: Combinatorial Theory and its Applications I (P. Erdös, A. Rényi and V. Sós, eds.), Colloquia Mathematica Societatis János Bolyai, vol. 4, pp. 119–133 (1970)

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

[5] Chalermsook, P., Chuzhoy, J.: Maximum independent set of rectangles. In: Proceedings of the twentieth annual ACM-SIAM symposium on discrete algorithms, pp. 892–901. SIAM (2009)

[6] Chalermsook, P., Walczak, B.: Coloring and maximum weight independent set of rectangles. In: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 860–868. SIAM (2021)

[7] Conforti, M., Cornuéjols, G., Rao, M.R.: Decomposition of balanced matrices. Journal of Combinatorial Theory, Series B **77**(2), 292–406 (1999)

[8] Conforti, M., Cornuéjols, G., Zambelli, G.: Integer Programming. Springer, Berlin (2014)

[9] Fulkerson, D.R., Hoffman, A.J., Oppenheim, R.: On balanced matrices. In: M.L. Balinski (ed.) Pivoting and Extension: In honor of A.W. Tucker, pp. 120–132. Springer, Berlin (1974)

[10] Garey, M.R., Johnson, D.S.: Computers and Intractability, vol. 174. Freeman San Francisco (1979)

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

[12] Hoffman, A.J., Kruskal, J.B.: Integral boundary points of convex polyhedra. Linear Inequalities and Related Systems pp. 223–246 (1956)

[13] Jackson, A.: The 2000 Fulkerson Prize. Notices of the AMS **47**(9), 1086 (2000)

[14] Johnson, D.S.: The NP-completeness column: an ongoing guide. Journal of Algorithms **6**(3), 434–451 (1985)

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

[16] Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)

[17] Liou, W., Tan, J.M., Lee, R.C.: Minimum rectangular partition problem for simple rectilinear polygons. IEEE transactions on computer-aided design of integrated circuits and systems **9**(7), 720–733 (1990)

[18] Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: A survey. European Journal of Operational Research **141**(2), 241–252 (2002)

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

[20] Masek, W.J.: Some NP-complete set covering problems. Unpublished manuscript (1979)

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

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

[23] Schrijver, A.: Theory of Linear and Integer Programming. John Wiley & Sons, Chichester (1998)

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

[25] Suk, T., Höschl IV, C., Flusser, J.: Decomposition of binary images—a survey and comparison. Pattern Recognition **45**(12), 4279–4291 (2012)

[26] Truemper, K.: A decomposition theory for matroids. V. Testing of matrix total unimodularity. Journal of Combinatorial Theory, Series B **49**(2), 241–281 (1990)

[27] Walter, M., Truemper, K.: Implementation of a unimodularity test. Mathematical Programming Computation **5**(1), 57–73 (2013). URL `http://matthiaswalter.org/TUtest/`

[28] Zambelli, G.: A polynomial recognition algorithm for balanced matrices. Journal of Combinatorial Theory, Series B **95**(1), 49–67 (2005)