Balancedness in the Weighted Rectangles Partitioning Problem

Paul Deuker[a], Ulf Friedrich[b,*]

[a] *Technical University of Munich, Department of Mathematics, Munich, Germany*
[b] *Otto von Guericke University, Faculty of Mathematics, Magdeburg, Germany*

Abstract

A rectangle is subdivided into pixels which are either active or inactive. The weighted rectangles partitioning problem (WRP) is finding a partition of the active pixels into weighted, smaller rectangles and maximizing the total weight. Using an IP formulation, instances with an integral relaxation polyhedron are characterized by the balancedness of the problem matrix. Is is discussed how this approach connects WRP to balanced hypergraphs. Numerical experiments illustrate the benefits of detecting balancedness for solving WRP.

Keywords: partition problem, integer programming, balanced matrix, hypergraph
2010 MSC: 90C10, 90C05, 05C50, 05C65

1. Introduction

The weighted rectangles partitioning problem (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 sum of all weights is maximal.

This game is interesting not only from a gambler's point of view, but also from a mathematical perspective. Since no non-active pixels may be used, we are looking for a *partition into rectangles* of the active pixels such that the total weight is maximized. Our aim is to characterize optimal partitions into rectangles for a general objective function and problem fields of arbitrary size. In addition, based on structural results for the problem, we want to discuss algorithms to find these partitions. Clearly, the WRP is related to other partitioning problems, see the discussion in Section 1.2 or the survey article [15].

1.1. Problem Definition

In the following, we define the two-dimensional WRP formally. The setting can be generalized to three or more dimensions in a straightforward way.

For $p, q \in \mathbb{N}$ we consider a rectangle $R_{p,q}$ which is divided in $p \times q$ pixels. We call this rectangle the *problem field* of the WRP. Each pixel of a 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 extended to non-rectangular problem fields by extending the field with empty pixels. A problem field can be 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.

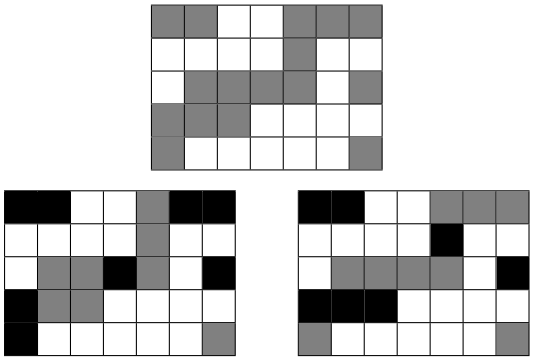

Figure 1: Two different rectangle partitions of a 5×7 problem field. Active pixels are displayed in gray in the upper picture, sample partitions in black and gray below.

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 generic weights without assuming additional properties such as monotonicity or submodularity in the size of the rectangles. Depending on the weight function, 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 maximum-weight rectangles partitioning problem (WRP) is the task to cover all active pixels with disjoint elements from $S_{p,q}$, without covering

*Corresponding author.

any inactive pixels, and maximize the sum of weights of the used rectangles.

By disjoint we mean that two rectangles do not share the same pixel, i.e., they are non-overlapping. The problem is always feasible as a trivial solution uses one 1×1 rectangles 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 generally have a different overall value depending on the weight function.

1.2. Related Research

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 additional details on the problems mentioned below.

In general the maximum-weight partition into rectangles problem can be identified as set partitioning problem [9]. 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 [11, 13] 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 much similarity to WRP is the two-dimensional knapsack problem, see [16]. In both problems rectangles have to be be placed on a problem field while maximizing the 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 [6, 17] can be related to the generalization of WRP to an unbounded problem field.

The literature on rectangular decomposition, e.g., [14, 18, 20], 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 [21]. 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 Section 3 we will see that a WRP problem field can also be represented by a hypergraph. Thus, to solve WRP, we can analogously search for a maximum weight edge partition of hypergraph edges. Here each vertex must be contained in the matching exactly once. We refer to [5, 7] for a detailed description of the setting.

1.3. Structure of this Work

In the next section, we formulate WRP as an integer program. Based on this formulation, we search for easy instances of WRP by considering balancedness of the system matrix. The main results of this section is the complete characterization of balanced instances given in Theorem 8. In Section 3 we link our findings to the classical balancedness definition of Berge for Hypergraphs and reformulate our results accordingly. Section 4 contains a numerical study to show the advantages of balancedness detection when solving WRP instances. We compare our algorithm to naive approaches and existing methods based on totally unimodular matrices. Finally, we give an outlook on possible extensions of WRP, connect these to existing problems, and assess our findings in Section 5.

2. Solving WRP

2.1. IP Formulation

We begin the analysis of WRP by introducing an integer programming (IP) formulation of WRP. Let B be the set of active pixels and the let $R = R(B) \subset S_{p,q}$ 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 $A = (a_{r,b})$ as the incidence matrix of all possible rectangles. In particular, the columns of 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 $b \in r$. Hence, we have $a_r \in \{0,1\}^{|B|}$ for all $r \in R$. We define a variable x_r for all $r \in R$ and write w_r for the corresponding weight.

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

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

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. Balanced 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 [12] that links integer polyhedra to the total unimodularity of the system matrix A. Instead, we apply results on balanced matrices developed in [4, 8, 10] to check 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 [19, Theorem 21.7]. As the results in Section 4 show, checking for balancedness can be performed more efficiently for WRP than testing for totally unimodularity. We build on 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.

Thus, a balanced matrix is a {0,1} matrix that does not contain any square submatrix of odd order having 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.

The definition of a journey is inspired by the definition of a tour in the travelling salesperson problem. The key differences are that not every pixel must be visited and the way of moving from one pixel to another. We can characterize balanced WRP problem fields with the help of journeys.

Theorem 4. *The incidence matrix A of a WRP problem field is unbalanced if and only if the field contains a journey with an odd number of steps, using only rectangles which are no sub-rectangles of each other.*

Proof. First, assume that the problem field contains a journey with the given properties. We prove that A is unbalanced.

For this proof we observe three properties: The journey uses an odd number of steps that correspond to an odd number of rectangles. Each visited pixel is a start and an endpoint of one step, therefore each pixel is used exactly twice. By the definition of a journey, the number of nodes and steps are the same.

We conclude that the rectangles picked in the journey determine the columns of the unbalanced submatrix of the problem matrix in Formulation (1). The pixels picked in the journey determine the rows of this submatrix. Because the number of steps and nodes are the same, this submatrix is square. This shows that A is unbalanced.

For the proof of the equivalence, assume that A is unbalanced now. We have to construct a journey with the stated properties: Since the matrix is unbalanced, there is a square, odd sized submatrix of A containing two nonzero elements in each row and column. Let M be the smallest matrix of this type in what follows.

Each column of M represents one rectangle of the problem field and each row of M represents one pixel of the problem field. As M is the smallest unbalanced submatrix, only rectangles which are no sub-rectangles of each other are taken into account.

For each two active pixels in a rectangle we connect both with one step. As the size of the matrix is odd, we obtain with those steps a journey as defined in Section 2.1 with an odd number of steps and rectangles which are not containing each other. □

Next, we use Theorem 4 to show that three types of problem fields are unbalanced. These examples are the key observations for the proof of Theorem 8 below, which gives a complete characterization of balanced fields of WRP.

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

Proof. Consider the problem field in Figure 2. We construct a journey with the properties of Theorem 4. The

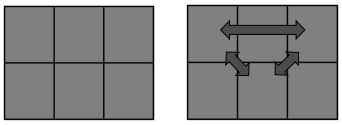

Figure 2: An unbalanced combination of 6 active pixels

corresponding matrix representation of the problem field is given by

$$\left(\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & \end{array} \middle| \; I_6 \right),$$

where I_d is the identity matrix in d dimensions. In view of Theorem 4, a submatrix containing two ones in each row and column must be found. One possible choice is the following gray 3×3 submatrix

$$\left(\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & \end{array} \middle| \; I_6 \right).$$

If a problem field contains a 3×2 rectangle of active pixels, the field's matrix contains the gray submatrix above. Therefore, the matrix is not balanced. □

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

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

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

Corollary 7. *Any problem field containing a circle of active pixels with an inactive pixel inside the circle is unbalanced.*

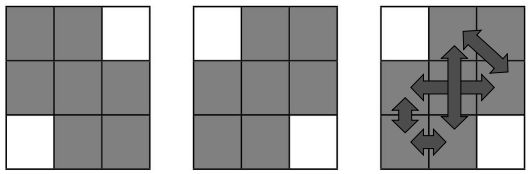

Figure 3: Left and middle: Two unbalanced combinations of active pixels. Right: Illustration of a journey showing that the combinations are unbalanced.

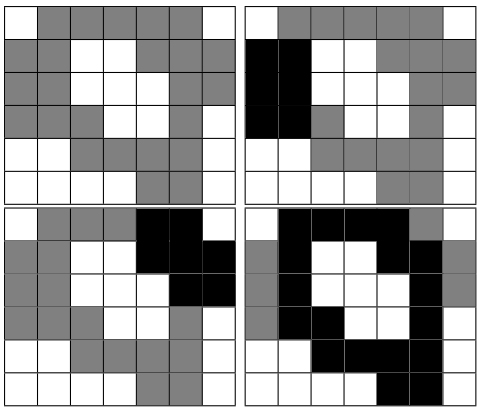

Figure 4: Sample field with the three different cases in Theorem 8.

Proof. In a problem field with a circle of active pixels and an inactive pixel in the middle it is straightforward to find a journey: The rectangles and active pixels next to the white pixels can be used as steps. This idea is exemplified in the lower right problem field of Figure 4.

At least one rectangle of these contains three or more active pixels. As one "big" step in this rectangle can be divided in two "smaller" steps or vice versa, every such journey can be transformed into one with an odd number of steps. Theorem 4 can be applied and the problem field is unbalanced. □

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

- *a* 3×2 *or* 2×3 *active rectangle,*
- *two* 2×2 *active rectangles having exactly one pixel in common,*
- *an active circle with an inactive pixel inside.*

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

Now, consider an arbitrary unbalanced problem field. Active pixels that are not contained in an odd journey can be ignored without loss if generality since such pixels do not affect the balancedness. This leaves the following three cases to be considered, see Figure 4.

Case 1: The field contains a rectangle with one side of the length ≥ 2 and one side of the length ≥ 3. Corollary 5 states that this field is unbalanced.

Case 2: This field contains two active 2×2 rectangles that touch share exactly one pixel. Corollary 6 states that this field is unbalanced as well.

Case 3: Assuming that cases 1 and 2 do not apply, we need to prove that the fields contain an active circle with an inactive pixel inside.

Choosing any active pixel of the unbalanced field, it cannot be part of a rectangle bigger than 2×2, but rectangles of the size $1 \times n$ for $n \in \mathbb{N}$. If it is part of a 2×2 rectangle and this rectangle is connected to another rectangle with the same size, they cannot have pixels in common, as the other cases would apply then. So each of those rectangles have only one pixel which are next to each other. As there can be just one step using this two pixels, no non-circular journey is possible using those two rectangles.

The same reasoning applies if one of this rectangles has one side of size one. If the chosen active pixel is part of a rectangles with size $1 \times n$ for $n \in \mathbb{N}$, the same idea holds and this pixel can be used in a journey exactly once. Since the field only contains active pixels used in an odd journey, this journey must be one using an active circle.

Finally, this circle must have an inactive pixel inside, otherwise the other cases would apply. □

From a computational perspective, it is important to discuss the complexity of the criteria introduced above. Checking whether the corollaries used in Theorem 8 apply is relatively easy: The Corollaries 5 and 6 can be checked in $\mathcal{O}(pq)$ for a field of size $p \times q$.

For testing Corollary 7, a depth-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 length greater than 4.

3. WRP-induced Hypergraphs

Following Berge [4], we introduced the notion of balanced matrices above with the help of hypergraphs: A hypergraph is balanced if every odd cycle has an edge which contains at least three of its vertices. The question arises how the characterisations of balanced WRP matrices in Theorem 8 fits into the original hypergraph perspective.

In Figure 5 we see the WRP problem field that was already considered in Corollary 6 the problem field matrix, and one possible hypergraph of this matrix.

Using the balancedness definition of Berge, we have to check every possible odd cycle and determine whether it contains an edge with at least three of its vertices. Such a circle which does not fulfill the property is the one in bold lines in the hypergraph of Figure 5.

With the new characterization using odd journeys, we just have to find an odd journey which does not use any sub-rectangle of one of its chosen rectangles. As shown in Corollary 6, there are just two possibilities for such journeys, which can be found easily. Because of the oddness, we visit every pixel except the one in the middle.

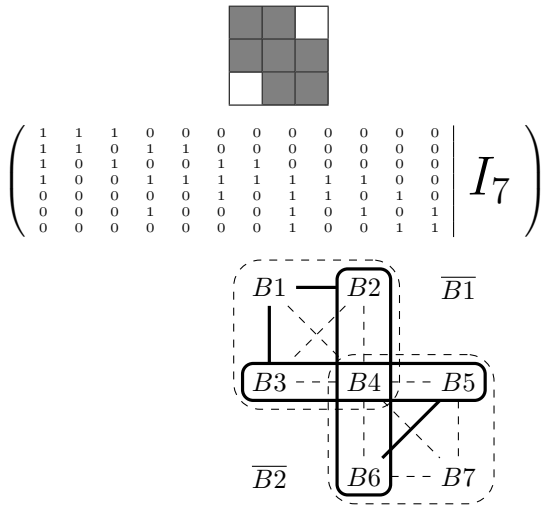

Figure 5: WRP problem field, corresponding matrix representation, and a corresponding hypergraph. Active pixels are labeled $B1, \ldots, B7$ and inactive pixels are labeled $\overline{B1}$, $\overline{B2}$.

Both characterizations give the same pixel set which causes non-balancedness, but the new one in a rather straightforward way.

For an $p \times q$ sized rectangular problem field, the number of possible rectangles is upper bounded by $\frac{(q-1)q(p-1)p}{4}$. Therefore, the hypergraph can have up to the same number of edges and checking Berge's definition becomes intractable for large problem fields. With Theorem 8 one can still determine balancedness in $\mathcal{O}(pq)$ time.

With these advantage in mind, we rephrase the previous balancedness characterizations for WRP hypergraphs. First, note that a problem field that consists of only one row (or one column) induces a hypergraph that is balanced because it is impossible to visit the same pixel twice in any journey. We call an edge of a WRP hypergraph *two-dimensional* if it is not induced by pixels in only one row (or one column) of the WRP problem field. With this notation, several types of unbalanced WRP hypergraphs can be characterized.

Corollary 9 (Balanced WRP hypergraphs)**.** *The following WRP hypergraphs are unbalanced*

1. *a WRP hypergraph containing a two-dimensional edge with more than five vertices,*
2. *a WRP hypergraph containing two two-dimensional edges with four vertices each and one vertex in common,*
3. *a WRP hypergraph that has an inactive node that is completely enclosed by edges from actives nodes.*

Note that these statements are also equivalent to the fact that the hypergraph contains an odd cycle with an edge which contains at least three of its vertices.

If we extend the definition of rectangles to general solids, we automatically end up with hypergraphs as a general characterization of connections between pixels respectively nodes. However, as not every hypergraph edge represents a rectangle and not every sub-rectangle of such a rectangle might be represented by an edge in the hypergraph, we cannot apply Theorems 4 and 8 on general hypergraphs.

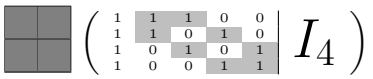

Figure 6: An active 2×2 rectangle and the corresponding matrix

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

Definition 10. 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.

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

Proof. If a problem field is totally balanced, it is balanced as well. If it is balanced, it can not contain any of the properties described in the Corollaries 5, 6, and 7. Therefore, the only remaining possible rectangular pixel structure which might contain an even journey is a 2×2 rectangle. This rectangle contains an even journey, see Figure 6. □

4. Numerical Experiment

The conditions of Theorem 8 are computationally easy to check. This is an interesting feature when numerically comparing the results above with other integrality conditions. For this comparision, the implementation of a totally unimodularity test by Walter and Truemper [23] is used. We call this benchmark *TU-Test* in the following. The TU-Test algorithm uses an implementation of [22]. Balancedness is checked with a simple Python implementation of Theorem 8. Both algorithms are single thread and may are not fully optimized for performance.

For the experiment, 100 random problem fields with the sizes 5×5, 10, 15×15, 20×20, and 25×25 were generated. We run each instance five times each and report averaged runtimes. Since the two algorithms test for different conditions, different results are to be expected. As totally unimodularity matrices are also balanced [19, Section 21.5] our algorithm recognizes more integral problem instances than TU-Test, which can be seen in Figure 7.

There are pronounced differences in the runtimes reported in Figures 8 and 9. The balancedness test clearly outperforms the benchmark in terms of runtime as shown in Figure 8. In addition, 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.

Next, we connect balancedness detection to the solution of the original optimization problem by using the number of detected integral problem fields, the cumulative runtimes

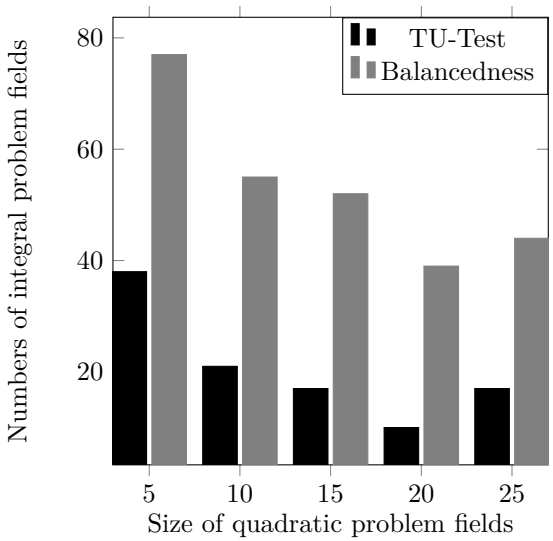

Figure 7: Integral instances detected by the two algorithms.

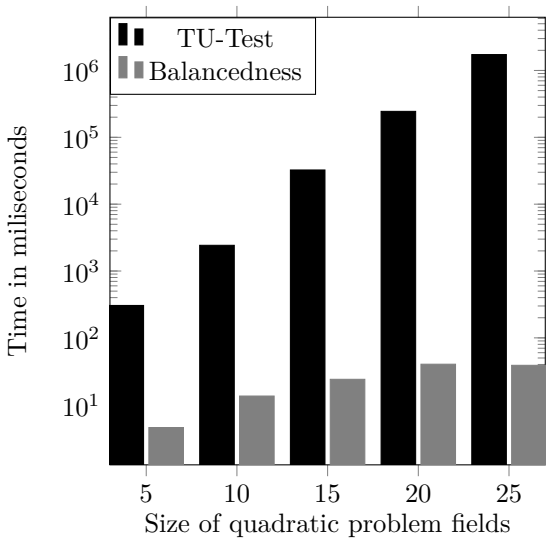

Figure 8: Runtimes of solution algorithms over all instances (integral and non-integral).

size	TU	bal	time TU	time bal	time IP
5×5	38	77	0.35s	0.005s	0.245s
10×10	21	55	2.42s	0.013s	1.56s
15×15	17	52	32.5s	0.024s	5.92s
20×20	10	39	244s	0.040s	15.7s
25×25	17	44	1729s	0.039s	49.1s

Table 1: Overview of the problem field sizes of the tested instances, number of recognized instances by the TU-Test benchmark (TU) and ballancedness detection algorithm (bal), total runtimes time TU and time bal of the algorithms and the runtime time IP when solving formulation (1) using gurobipy.

of the algorithms and the cumulative runtime of a basic `gurobipy` implementation of Formulation (1) are reported in Table 1. We conclude that the balancedness algorithm can be used as pre-solver for the IP to detect easy instances immediately because of its speed. Comparing the last three columns of Table 1 implies that using TU-Test as a pre-solver can not be recommended since the determination of total unimodularity took longer than solving the IP.

We conclude that there is strong numerical evidence that balancedness detection superior to testing for total unimodularity when considering WRP instances. 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 specfically to the WRP problem fields while the TU-Test benchmark checks general matrices for total unimodularity. Thus the balancedness recognition algorithm runs in $\mathcal{O}(pq)$ and the TU-Test runs in $\mathcal{O}((\#\text{ active pixels}+\#\text{ possible rectangles})^5) = \mathcal{O}((|B| + p^2q^2)^5)$, as reported in [23]. We expect that adapting the TU-Test algorithm to the special type of WRP instances might improve its runtime significantly and partly close the gap reported above. According to the TU-Test project webpage [23], an extension for balancedness detection is currently under development and a comparison with this new version might reveal additional insights.

5. Conclusion and Future Work

We have introduced WRP as a geometric optimization problem with close connections to other combinatorial problems of the partitioning type. Starting from a straightforward IP formulation, we adapted the classical concept of balancedness to detect the integrality of WRP instances and argued why this approach is more suitable than checking for total unimodularity. The theorems proved in Section 2.1 characterize all balanced WRP instances. Based on our numerical study, integrality detection and pre-solving based on our method 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 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 and allowing more general shapes such as polyominoes is an interesting extension. However, it is unclear how the concept of journeys used in the proofs above carries over to more general shapes. We have discussed briefly in Section 3 how this questions relates our findings to detecting balancedness for general hypergraphs, i.e., not necessarily WRP-induced hypergraphs. An immediate open question is therefore whether it is possible to identify classes of balanced hypergraphs that represent generalizations of WRP as introduced above. In addition, our discussion of WRP can easily be extended to three or more dimensions with the same hypergraph arguments.

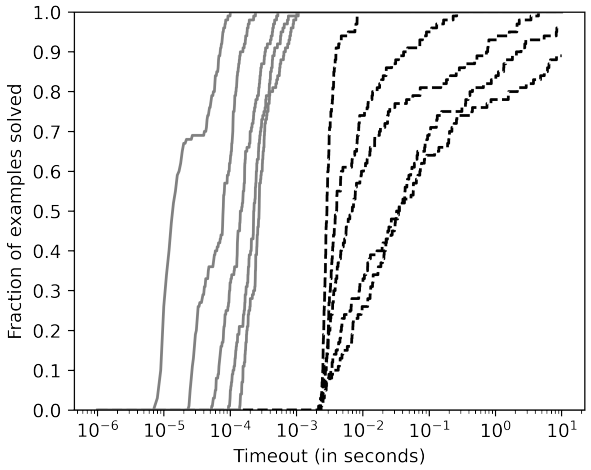

Figure 9: Number of solved instances using the TU-Test benchmark algorithm (dashed lines) and balancedness detection via Theorem 8 (solid gray lines). For each algorithm, the five lines show instances of sizes 5×5, 10, 15×15, 20×20, and 25×25 from left to right.

Finally, a more sophisticated computational study shall reveal additional insights on using balanced hypergraphs for the solution of larger WRP instances and related partition problems.

References

[1] Adamaszek, A., Wiese, A., 2013. Approximation schemes for maximum weight independent set of rectangles.
URL `https://arxiv.org/abs/1307.1774`

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

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

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

[5] Berge, C., 1984. Hypergraphs: Combinatorics of Finite Sets. Elsevier.

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

[7] Bretto, A., 2013. Hypergraph Theory: An Introduction. Springer.

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

[9] Conforti, M., Cornuéjols, G., Zambelli, G., 2014. Integer Programming. Springer.

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

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

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

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

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

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

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

[17] Martello, S., Monaci, M., Vigo, D., 2003. An exact approach to the strip-packing problem. INFORMS Journal on Computing 15 (3), 310–319.

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

[19] Schrijver, A., 1998. Theory of Linear and Integer Programming. John Wiley & Sons.

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

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

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

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