In Search of Balance: The Challenge of Generating Balanced Latin Rectangles

Mateo Díaz[1(✉)], Ronan Le Bras[2], and Carla Gomes[2]

[1] Center for Applied Mathematics, Cornell University, Ithaca, NY 14853, USA
md825@cornell.edu
[2] Computer Science Department, Cornell University, Ithaca, NY 14853, USA
{lebras,gomes}@cs.cornell.edu

Abstract. Spatially Balanced Latin Squares are combinatorial structures of great importance for experimental design. From a computational perspective they present a challenging problem and there is a need for efficient methods to generate them. Motivated by a real-world application, we consider a natural extension to this problem, balanced Latin Rectangles. Balanced Latin Rectangles appear to be even more defiant than balanced Latin Squares, to such an extent that perfect balance may not be feasible for Latin rectangles. Nonetheless, for real applications, it is still valuable to have well balanced Latin rectangles. In this work, we study some of the properties of balanced Latin rectangles, prove the nonexistence of perfect balance for an infinite family of sizes, and present several methods to generate the most balanced solutions.

Keywords: Latin Rectangles · Experimental design · Local search · Constraint satisfaction problem · Mixed-integer programming

1 Introduction

In recent years we have seen tremendous progress in search and optimization procedures, driven in part by hard challenging structured problems, such as those from combinatorial design. As an example, Spatially Balanced Latin squares (SBLSs) have led to the notion of streamlining constraints [7], in which additional, non-redundant constraints are added to the original problem to increase constraint propagation and to focus the search on a small part of the subspace, and other notions such as XOR constraints, a powerful general purpose streamlining technique, with applications in search, probabilistic reasoning, and stochastic optimization [1,5,6,8], and local search procedures [9].

SBLSs have been studied in detail, e.g., [4,10,11,14]. In the recent work, by combining streamlining reasoning and human-computation, Le Bras et al. [10,11] discovered the first polynomial time construction for the generation of SBLSs, for any size n such that $2n + 1$ is prime, and formally proved correctness of the

D. Salvagnin and M. Lombardi (Eds.): CPAIOR 2017, LNCS 10335, pp. 68–76, 2017.
DOI: 10.1007/978-3-319-59776-8_6

procedure. This constructive solution is the first example of the use of automated reasoning and search to find a general constructive algorithm for a combinatorial design problem for an unlimited range of sizes. Previous automated reasoning results in combinatorial mathematics were restricted to finding constructions for a particular (fixed) problem size.

The demand for balanced Latin squares arises from agronomic field experiments, where one wants to compare n treatments consisting of n fertilizers applied in different orderings. Such experiments can be modeled using a Latin Square − a standard procedure in the field of experimental design. However, for agronomic experiments there is a caveat, geometric imbalance can potentially bias the results, see for example [14]. In this paper, we propose as a new benchmark problem Partially Balanced Latin Rectangles (PBLRs), a natural extension SBLSs. PBLRs are also used in experimental design; in fact, in practice rectangular sizes are more common than squares, one of the key reasons why we study them. Despite significant progress concerning SBLSs, little is known about PBLRs. Here, we present some of the properties of balanced Latin rectangles, prove the nonexistence of perfect balance for an infinite family of sizes, and present several methods to generate the most balanced solutions. We also identify an interesting easy-hard-easy pattern in the run times of a MIP algorithm, for proving optimality, and a CSP approach, for finding feasible solutions. We believe that the PBLR problem is ideal as a benchmark for the study of different search and optimization approaches.

2 Preliminaries

Definition 1. *Let $k, n \in \mathbb{N}$ be positive numbers such that $k \leq n$ and A a $k \times n$ matrix. We say that A is a Latin rectangle if every cell of A contains a number in $\{1, \ldots, n\}$ and there are no repetitions in a row or a column.*

Let $u, v \in \{1, \ldots, n\}$ be different symbols and $i \leq k$. Define $d_i(u, v)$ as the distance between the symbols u and v in row i. We define the distance $d(u, v) := \sum_{i \leq k} d_i(u, v)$.

Definition 2. *We say that a $k \times n$ rectangle A is spatially balanced if all the distances $d(u, v)$ are the same. In order to abbreviate we refer to them as SBLR(k, n).*

The following proposition provides novel necessary and sufficient conditions for the existence of SBLRs.

Proposition 1. *If there exists a solution for SBLR(k, n) then the distance between any pair of symbols is equal to $\dfrac{k(n+1)}{3}$.*

Proof. If there is a solution for SBLR(k, n) then the sum of all the possible distances divided by all the possible pairs of symbols must be integer. For a fixed row, we have $n-d$ pairs with distance d. Since we have k rows, the sum of all the distances is equal to $k \sum_{d=1}^{n} d(n-d) = k \left(n \sum_{d=1}^{n} d - \sum_{d=1}^{n} d^2 \right) = \frac{kn(n-1)(n+1)}{6}$. Dividing this by the number of pairs, $\binom{n}{2}$, gives the result.

As a corollary, $k \equiv_3 0$ or $n \equiv_3 2$ are necessary conditions to have balance, since the distances are integers, $k(n+1)$ must be multiple of 3 and one of the two conditions must hold.

We note that two Latin rectangles are in the same *isotopy class* if it is possible to obtain one from the other by permuting its rows, columns or symbol labels. However, spatial balance is invariant only under row permutations or relabels of the symbols. In order to avoid redundancy, we say that a Latin rectangle is in its reduced form if the first row is the sequence $1, 2, 3, \ldots, n$ and the first column is in Lexicographic order, i.e. the digits are in ascending order.

2.1 Non-existence Results

Our experiments suggest that perfectly balanced rectangles do not exist, see Sect. 4. The next results support this claim for small ks.

Proposition 2. *For $k = 2$ the only n for which SBLR$(2, n)$ is nonempty is 2.*

Proof. When $(k, n) = (2, 2)$, the only solution is perfectly balanced. Let $n > 2$ and assume that there is a solution for SBLR(k, n). By Proposition 1 we know that $n = 3l + 2$, for some integer $l > 0$. Without loss of generality we may assume that in the first row the extremal symbols are 1 and n. Then, for the pair $(1, n)$ the smallest distance that a solution could achieve is n, as shown in the rectangle.

1	...	i	$i+1$	...	n
*	...	1	n	...	*

Hence, $d(1, n)$ cannot satisfy the constraint in Proposition 1, since $\frac{2(n+1)}{3} = 2l + 2 < 3l + 2 = n \leq d(1, n)$. This contradiction completes the proof.

Proposition 3. *For $k = 3$ the only n for which SBLR$(3, n)$ is nonempty is 3.*

Proof. For the case $(k, n) = (3, 3)$, any solution is balanced. For the cases $(k, n) = (3, 4)$ and $(k, n) = (3, 5)$, we used a complete approach to check all the possible solutions and proved that none exists.

Suppose now that $n > 5$, by Proposition 1 we know that if $A \in$ SBLR(k, n) then for every pair of symbols we have $d(a, b) = n + 1$. Again, without loss of generality we may assume that the first row of A is the sequence $1, 2, 3, \ldots, n$. Then $d_1(1, n) = n-1$, this forces the pair 1 and n to be next to each other in the next two rows, otherwise $d(1, n) \neq n+1$. Analogously, since $d_1(1, n-1) = n-2$, the pair $(1, n-1)$ must have a distance of 1 in one of the remaining rows and 2 in the other, as it is shown in the rectangle.

1	...	*	*	*	...	*	*	*	*	...	$n-1$	n
*	...	$n-1$	1	n	...	*	*	*	*	...	*	*
*	...	*	*	*	...	$n-1$	*	1	n	...	*	*

Thus, we can bound $d(n-1, n) = d_1(n-1, n) + d_2(n-1, n) + d_3(n-1, n) \leq 1 + 3 + 2 = 6 < n + 1$, which yields a contradiction, this completes the proof for all the remaining cases.

Proposition 4. *If there exists a solution for SBLR*(k,n) *then* $6 - \dfrac{18}{n+1} \leq k$.

Proof. Suppose that $A \in \text{SBLR}(k,n)$, without loss of generality we may assume that A is in its reduced form. With this setting we know that

$$d_1(1,n-1) = n-2, \qquad d_1(1,n) = n-1 \quad \text{and} \quad d_1(n-1,n) = 1. \tag{1}$$

By the triangular inequality we have that for every $i = 1,\ldots,k$, $d_i(n-1,n) \leq d_i(1,n-1) + d_i(1,n)$, summing over all the $i \geq 2$,

$$\sum_{i=2}^{k} d_i(n-1,n) \leq \sum_{i=2}^{k} d_i(1,n-1) + \sum_{i=2}^{k} d_i(1,n).$$

Using the distances that we know (1), we obtain $d(n-1,n) - 1 \leq d(1,n-1) - (n-1) + d_i(1,n) - (n-2)$, since A is perfectly balanced, $\frac{k(n+1)}{3} \leq \frac{2k(n+1)}{3} - 2n + 4$. By simplifying this expression we get the stated result.

With our last result we proved that there are no solutions for $k = 4$ and $n > 8$, and there are also no solutions for $k = 5$ and $n > 17$. Thus, at least for small ks and for all the sizes that do not satisfy $k \equiv_3 0$ or $n \equiv_3 2$, perfect balanced cannot be achieved.

2.2 Partially Balanced Latin Rectangles

These results suggest that $\text{SBLR}(k,n)$ may be an infeasible problem when $k < n$. For real applications if balance cannot be achieved, it is useful to have the most balanced solution. To measure the balance of a rectangle R we define the *imbalance function*, $\mathbb{I}(R) = \sum_{i<j} \left| d(i,j) - \frac{k(n+1)}{3} \right|$. Note that this function is non-negative and if the rectangle is perfectly balanced $\mathbb{I}(R) = 0$.

Algorithm 1. Random-restart hill climbing balanced Latin Rectangles

```
input  : k, n
output: A well Balanced Latin Rectangle R
Generate a cyclic n × n Latin Square L and define R as a k × n rectangle of
  zeros;
for a fixed number of iterations do
    Take k random rows of L and assign them to R;
    for a fixed number of iterations do
        Draw a pair of columns (i, j) at random and assign to R̄ a copy of R
          with the columns (i, j) switched;
        if I(R̄) < I(R) then
            R = R̄;
            if I(R) = 0 then
                Break all the loops, we found perfect balance;
```

Definition 3. *Let R be a $k \times n$ Latin rectangle, we say that it is partially spatially balanced if $\mathbb{I}(R)$ is minimum. For a fixed size, we denote by $PBLR(k, n)$ the set of these rectangles.*

Thus, if perfect balance is achieved, PBLR and SBLR are equivalent. This presents a new challenge, since the minimum value of the imbalance is unknown a priori. Now, we need to determine the minimum imbalance, that we call I_{kn}, and at the same time generate the associated partially balanced rectangle.

3 Proposed Solutions

We developed three different approaches for generating rectangles with minimum imbalance: a local search algorithm, a constraint satisfaction encoding and a mixed-integer programming encoding. In this section we describe these methods and discuss some advantages and disadvantages of each encoding.

3.1 Local Search

The objective of the algorithm is to minimize the imbalance function, by performing random-restart hill climbing local search on the space of Latin rectangles. Our algorithm starts with a cyclic solution, i.e., every row is obtained by shifting by one position the previous row, and it was adapted from one of the best known algorithms for balanced Latin squares proposed in [13]. One issue with this algorithm is that it starts with an initial solution and never leaves its isotopic class. So if there is no solution for that particular class, the algorithm will only explore that class. Nevertheless, this technique was used before with successful results for Latin squares. In that case, the stopping criterion is clear, $\mathbb{I}(R) = 0$; but for partially balance Latin rectangles the value of $\min_R \mathbb{I}(R)$ is unknown. Thus, while this incomplete approach can only be used to find upper bounds on the minimum imbalance, it constitutes a relevant baseline and in practice, it is able to find good solutions fast.

3.2 Constraint Satisfaction Problem and Mixed-Integer Programming Encodings

For any $m \in \mathbb{N}$ define $[m] = \{1, \ldots, m\}$. Let Δ_n be the set of possible pairs (i, j) with $i, j \in [n]$ such that $i < j$. We consider a simple constraint satisfaction problem that encodes the problem, where the variables R_{ij} indicate the position (column number) of the symbol j in row i. The first set of *alldifferent* constraints guarantees that the symbols cover all the columns, while the second one guarantees the non-repetition of symbols in columns. The last constraint assures that the imbalance of solutions is upper bounded by the constant M.

$$\begin{array}{ll} R_{ij} \in [n] & \forall i \in [k], j \in [n] \\ \mathit{alldiff}(R_{i1}, ..., R_{in}) & \forall i \in [k] \\ \mathit{alldiff}(R_{1j}, ..., R_{kj}) & \forall j \in [n] \\ \sum_{(i,j) \in \Delta_n} \left| \frac{k(n+1)}{3} - \sum_{l=1}^{k} |R_{li} - R_{lj}| \right| \leq M. & \end{array} \tag{2}$$

A classical approach to boost the performance of this algorithm is to use symmetry breaking constraints [2,3,12], which is done with the use of the following constraints:

$$\begin{array}{ll} R_{1j} = j & \forall j \in \{1, \ldots, n\} \\ R_{i1} < R_{(i+1)1} & \forall i \in \{1, \ldots, k-1\} \end{array} \tag{3}$$

These constraints enforce that the Latin rectangle is in its reduced form, i.e., the first row is the sequence $1, 2, 3, \ldots, n$, and the first column is in Lexicographic order, i.e. the digits are in ascending order. By imposing these constraints we are shrinking the size of the space of solutions by a factor of $n!k!$ and only eliminating equivalent elements from isotopic classes. For balanced squares, previous formulations use other streamlining constraints such as symmetry, that have proven to be very effective and, in fact, the solutions found with those constraints served as basis to uncover the underlying pattern that gave rise to the polynomial time construction in [10]. Unfortunately, the rectangular shape prevents us from applying such constraints.

The main drawback with the CSP formulation is that its performance depends on the scheme chosen for the values of M and it does not incrementally learn from one iteration to the next. To tackle this issue we transform this formulation into a mixed-integer program, using $\mathbb{I}(\cdot)$ as an objective function. The MIP formulation can be written as:

$$\begin{array}{ll} \min \sum_{(i,j)\in\Delta_n} D_{ij} & \\ \text{s. t. } R_{ij} \in [n] & \forall i \in [k], j \in [n], \\ \quad D_{ij} = \left| \frac{k(n+1)}{3} - \sum_{l=1}^{k} |R_{li} - R_{lj}| \right| & \forall (i,j) \in \Delta_n \\ \quad R_{ih} \neq R_{il} & \forall i \in [k], (h,l) \in \Delta_n, \\ \quad R_{il} \neq R_{jl} & \forall (i,j) \in \Delta_k, l \in [n] \end{array}$$

The formulation in this case is almost the same as in (2), with the only difference that the left-hand side of the last constraint becomes the function that we are minimizing. Also, note that the alldifferent constraints translate to a conjunction of binary disequalities.

Both the CSP and MIP encoding can be used to certify optimality, namely they can prove that they find the minimal imbalance I_{kn}. Nevertheless, the MIP solver does not require an initial guess for the objective function bound. In addition, through branch and cut methods, it generates lower bounds on the optimal value of imbalance using a hierarchy of linear relaxations. In particular, even before proving optimality, the MIP solver can certify that there is no perfect balance if it finds a nontrivial lower bound, which could be potentially practical even for the square case.

4 Experimental Results

We identified an interesting **easy-hard-easy** pattern in the runtimes of the MIP solver, for proving optimality, and the CSP, for finding a PBLR given I_{kn}, when

k/n varies. For a fixed n, the problem becomes harder as k increases; it is easier when $k = n$, see Fig. 1.

All the experiments were run using 12 cores on a single compute server with 2.93 GHz Intel(R) Xeon(R) x5670 processors and 48 GB RAM. For the optimization and satisfiability problems we use IBM ILOG CPLEX-CP Optimizer.

Proving Optimality. To compare the CSP and MIP formulations, we ran experiments for proving optimality, each method with $6 \leq n \leq 8$, $2 \leq k \leq n$ and a time limit of 35 h. The MIP solver was able to solve the problem for all the instances up to $n = 8$ and $k = 4$, see Table 1. To prove optimality using CSP

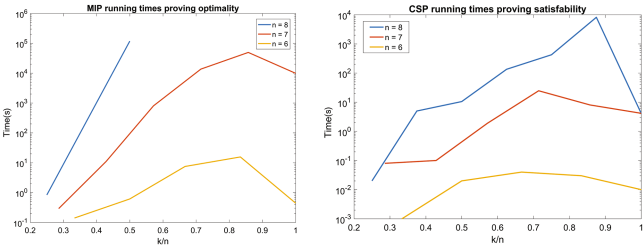

Fig. 1. Easy-hard-easy patterns in running times of the MIP solving for proving optimality and the CSP proving satisfiability of I_{kn} with $n = 6, 7, 8$.

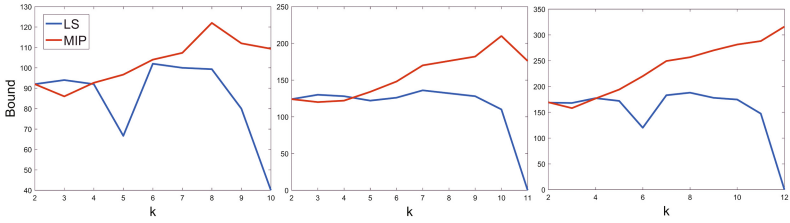

Fig. 2. Bounds found by the local search algorithm and the MIP solver, from left to right $n = 10, 11, 12$.

Table 1. Minimum imbalance found with our algorithms. The column corresponds to k and the row to n. Grey cells indicate a proven optimal value.

	2	3	4	5	6	7	8	9	10	11	12
4	$2.\overline{6}$	4	$5.\overline{3}$								
5	8	6	8	0							
6	16	12	$13.\overline{3}$	16	0						
7	28	22	$22.\overline{6}$	$22.\overline{6}$	20	$18.\overline{6}$					
8	40	36	32	30	24	28	0				
9	$65.\overline{3}$	56	$56.\overline{6}$	56	52	66	60	0			
10	92	86	92	$66.\overline{6}$	102	100	$99.\overline{3}$	80	40		
11	124	120	122	122	126	136	132	128	110	0	
12	168	158	$162.\overline{6}$	$170.\overline{6}$	120	183	$184.\overline{6}$	178	$174.\overline{6}$	$147.\overline{3}$	0

we start with an initial feasible guess for the bound M and decrease it until the problem is unsatisfiable. A natural way to measure the time is to take the sum of the last satisfiable case and the first infeasible one. The CSP solver was good at proving optimality for the square cases where balance is feasible. In almost any other case, if $n \leq 6$ it takes more than the time limit to run the unsatisfiable case. Thus, the MIP seems to be a better method for proving optimality in rectangles. However, the CSP formulation is fast at proving satisfiability once we have the minimum imbalance a priori. **Finding well balanced solutions for larger sizes.** For larger sizes, it is very difficult to prove optimality. As a consequence, we use our local search algorithm to find upper bounds. To compare this approach with the MIP formulation, we ran the MIP solver with a time limit of 1500 s and execute Algorithm 1 with 1000 iterations on both loops (it takes less than 1200 s). For large sizes the pattern is similar, if $k > 3$ the local search approach outperforms the MIP one, see Fig. 2.

5 Conclusions

We study the problem of generating balanced Latin rectangles. We conjecture (and prove for several cases) that unlike squares, it is impossible to have perfect balance for Latin rectangles. We examined the problem of finding the most balanced solutions, namely partially balanced Latin rectangles. We developed different methods to prove optimality and to find well balanced solutions when it is too expensive to guarantee optimality. We identified an interesting easy-hard-easy pattern in the run times of the MIP, for proving optimality, and CSP, for finding a feasible solution. For larger problems, our simple local search approach outperforms the MIP approach.

Finding Latin rectangles with minimum imbalance seems to be a very challenging computational problem. We believe that this problem is ideal as a benchmark for different search and optimization approaches. We leave as an open challenge for future researchers to improve the results in Table 1.

Acknowledgments. This work was supported by the National Science Foundation (NSF Expeditions in Computing awards for Computational Sustainability, grants CCF-1522054 and CNS-0832782, NSF Computing research infrastructure for Computational Sustainability, grant CNS-1059284).

References

1. Ermon, S., Gomes, C.P., Sabharwal, A., Selman, B.: Low-density parity constraints for hashing-based discrete integration. In: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014, pp. 271–279 (2014)
2. Fahle, T., Schamberger, S., Sellmann, M.: Symmetry breaking. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 93–107. Springer, Heidelberg (2001). doi:10.1007/3-540-45578-7_7

3. Gent, I.P., Smith, B.M.: Symmetry breaking in constraint programming. In: Proceedings of ECAI-2000, pp. 599–603. IOS Press (2000)
4. Gomes, C., Sellmann, M., Van ES, C., Van Es, H.: The challenge of generating spatially balanced scientific experiment designs. In: Régin, J.-C., Rueher, M. (eds.) CPAIOR 2004. LNCS, vol. 3011, pp. 387–394. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24664-0_28
5. Gomes, C.P., Hoffmann, J., Sabharwal, A., Selman, B.: Short XORs for model counting: from theory to practice. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 100–106. Springer, Heidelberg (2007). doi:10.1007/978-3-540-72788-0_13
6. Gomes, C.P., Sabharwal, A., Selman, B.: Model counting: a new strategy for obtaining good bounds. In: Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, Boston, Massachusetts, USA, 16–20 July 2006, pp. 54–61 (2006)
7. Gomes, C.P., Sellmann, M.: Streamlined constraint reasoning. In: Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, 27 September–1 October 2004, Proceedings, pp. 274–289 (2004)
8. Gomes, C.P., van Hoeve, W.J., Sabharwal, A., Selman, B.: Counting CSP solutions using generalized XOR constraints. In: AAAI, pp. 204–209 (2007)
9. Van Hentenryck, P., Michel, L.: Differentiable invariants. In: Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, 25–29 September 2006, Proceedings, pp. 604–619 (2006)
10. Le Bras, R., Gomes, C.P., Selman, B.: From streamlined combinatorial search to efficient constructive procedures. In: AAAI (2012)
11. Le Bras, R., Perrault, A., Gomes, C.: Polynomial time construction for spatially balanced Latin squares (2012)
12. Rossi, F., Van Beek, P., Walsh, T.: Handbook of Constraint Programming. Elsevier, Amsterdam (2006)
13. Smith, C., Gomes, C., Fernandez, C.: Streamlining local search for spatially balanced Latin squares. In: IJCAI, vol. 5, pp. 1539–1541. Citeseer (2005)
14. Van Es, H., Van Es, C.: Spatial nature of randomization and its effect on the outcome of field experiments. Agron. J. **85**(2), 420–428 (1993)