The Challenge of Generating Spatially Balanced Scientific Experiment Designs*

Carla Gomes, Meinolf Sellmann, Cindy van Es, and Harold van Es

Cornell University
{gomes,sello}@cs.cornell.edu {clv1,hmv1}@cornell.edu

The development of the theory and construction of combinatorial designs originated with the work of Euler on Latin squares. A Latin square on n symbols is an $n \times n$ matrix (n is the order of the Latin square), in which each symbol occurs precisely once in each row and in each column. Several interesting research questions posed by Euler with respect to Latin squares, namely regarding orthogonality properties, were only solved in 1959 [3]. Many other questions concerning Latin squares constructions still remain open today.

From the perspective of the Constraint Programing (CP), Artificial Intelligence (AI), and Operations Research (OR) communities, combinatorial design problems are interesting since they possess rich structural properties that are also observed in real-world applications such as scheduling, timetabling, and error correcting codes. Thus, the area of combinatorial designs has been a good source of challenge problems for these research communities. In fact, the study of combinatorial design problem instances has pushed the development of new search methods both in terms of systematic and stochastic procedures. For example, the question of the existence and non-existence of certain quasigroups (Latin squares) with intricate mathematical properties gives rise to some of the most challenging search problems in the context of automated theorem proving [16]. So-called general purpose model generation programs, used to prove theorems in finite domains, or to produce counterexamples to false conjectures, have been used to solve numerous previously open problems about the existence of Latin squares with specific mathematical properties. Considerable progress has also been made in the understanding of symmetry breaking procedures using benchmark problems based on combinatorial designs [5, 6, 9, 13]. More recently, the study of search procedures on benchmarks based on Latin squares has led to the discovery of the non-standard probability distributions that characterize complete (randomized) backtrack search methods, so-called heavy-tailed distributions [8].

In this paper we study search procedures for the generation of *spatially balanced Latin squares*. This problem arises in the design of scientific experiments. For example, in agronomic field experiments, one has to test and compare different soil treatments. Two different soil treatments may correspond to two different fertilizers or two different ways of preparing the soil. Most agronomic field experiments are implemented through randomized complete block designs (RCBD) where each block has as many experimental units as treatments [4]. Use of blocks is in most cases justified by spatial variability in fields, and this layout is an attractive way to organize replications. This approach to experimental design uses random allocation of treatments to plots, which is used to ensure that a treatment is not continually favored or handicapped in successive replications by user bias or some extraneous source of variation [2]. Although this

* This research was partially supported by AFOSR grants F49620-01-1-0076 (Intelligent Information Systems Institute) and F49620-01-1-0361 (MURI).

randomization approach is intuitively attractive, it has been shown to cause biases and imprecision under most field conditions [15]. The reason for this is that underlying soil characteristics are typically non-random and show field trends, spatial autocorrelation, or periodicity [14]. For example, fertility patterns in fields often exhibit high and low areas due to, among others, erosion, drainage variability, and management history. The classical randomization process does not explicitly account for such field patterns, and many realizations of such RCBD designs may result in undesirable outcomes.

To address the limitations of the traditional RCBD designs, van Es and van Es [15] proposed *spatially balanced* experimental designs that are inherently robust to non-random field variability. This approach uses dummy indicators and the treatments are randomly assigned to the indicators. In other words, treatments are randomly allocated to optimized designs, rather than to plots. Such designs may be spatially-balanced complete block designs or spatially-balanced Latin squares, the latter being a special case of the former where the number of treatments equal the number of replications.

We report our preliminary results concerning the generation of spatially-balanced Latin squares. In a spatially-balanced Latin square all pairs of symbols (treatments, in agronomic terms) have the same total distance in the Latin square. The distance of two symbols in a given row is the difference between the column indices of the symbols. The existence of spatially-balanced Latin squares is an open question in combinatorics, and no polynomial time constructions for the generation of spatially-balanced Latin squares have been found yet. Therefore, in order to get some insights into the structure of spatially-balanced Latin squares we used general local and complete search methods. We discovered that local search methods do not scale well on this domain, failing to find the global optimum for instances larger than order 6. This result was somehow surprising, especially given that local search methods perform well on generating (regular) Latin squares. Note that generating spatially-balanced Latin squares is considerably more difficult than generating regular Latin squares.[1] On the other hand, our results with a CP based approach were very promising and we could generate totally spatially-balanced Latin squares up to order 18. Furthermore the CP based models provided us with interesting insights about the structure of this problem that allowed us to conjecture the existence of polynomial time constructions for generating spatially-balanced Latin squares. We are currently working on finding such efficient constructions.

The structure of the paper is as follows. In the next section we provide basic definitions. In section 2 we describe our simulated annealing approach and we present our main CP based model. In section 3 we provide empirical results.

1 Preliminaries

Definition 1. [Latin square and conjugates] *Given a natural number $n \in \mathbb{N}$, a Latin square L on n symbols is an $n \times n$ matrix in which each of the n symbols occurs exactly once in each row and in each column. We denote each element of L by l_{ij}, $i, j \in \{1, 2, \cdots, n\}$. n is the order of the Latin square. Given a Latin square L of order n, its* row (column) conjugate *R (C) is also a Latin square of order n, with symbols, $1, 2, \cdots, n$. Each element r_{ij} (c_{ij}) of R (C) corresponds to the row (column) index of L in which the symbol i occurs in column (row) j.*

[1] While the current state of the art of local search and backtrack search methods can easily generate Latin squares of order 100 or larger, the largest spatially-balanced Latin squares that we can generate is 18, using considerably more sophisticated techniques. There are constructions for generating Latin squares of arbitrary order. Our comparison considers only the generation of Latin squares using local search or backtrack search methods.

Definition 2. [Row distance of a pair of symbols] *Given a Latin square L, the distance of a pair of symbols (k, l) in row i, denoted by $\mathrm{d}_i(k, l)$, is the absolute difference of the column indices in which the symbols k and l appear in row i.*

Definition 3. [Average distance of a pair of symbols in a Latin square] *Given a Latin square L, the average distance of a pair of symbols (k, l) in L is $\bar{\mathrm{d}}(k, l) = \sum_{i=1}^{n} \mathrm{d}_i(k, l)/n$.*

We make the following important observation:

Remark 1. Given a Latin square L of order $n \in \mathbb{N}$, the expected distance of any pair in any row is $\frac{n+1}{3}$.

Proof. (See [15]) When we denote the probability that a random pair has distance h with $P(h)$, the expected distance is $\mu = \sum_{h=1}^{n-1} hP(h)$. It holds $P(h) = \frac{2(n-h)}{n(n-1)}$, and therefore, $\mu = \frac{2}{n(n-1)} \left(n \sum_{h=1}^{n-1} h - \sum_{h=1}^{n-1} h^2 \right)$. Simplification yields $\mu = \frac{n+1}{3}$.

Clearly, a Latin square L of order $n \in \mathbb{N}$ is totally spatially balanced if every pair of symbols $1 \leq k < l \leq n$ has an average distance $\bar{d}(k, l) = \frac{n+1}{3}$. Consequently, we define:

Definition 4. *Given a natural number $n \in \mathbb{N}$, a* Totally Spatially Balanced Latin Square *(TBLS) is a Latin square in which $\bar{\mathrm{d}}(k, l) = \frac{n+1}{3} \quad \forall\, 1 \leq k < l \leq n$.*

Since $n\bar{\mathrm{d}}(k, l) = \sum_i \mathrm{d}_i(k, l) \in \mathbb{N}$, it follows that:

Remark 2. If there exists a TBLS of order n, then $n \bmod 3 \neq 1$.

In the following section, we present different computational approaches for the generation of Totally Spatially Balanced Latin Squares. We refer to this problem as the TBLS problem.

2 Totally Spatially Balanced Latin Square Models

2.1 Simulated Annealing

We started by developing a simulated annealing approach for the TBLS problem. Our work borrows ideas from a successful simulated annealing approach for the Traveling Tournament Problem [1].

Objective Function Given that we are interested in the research question of the existence of Totally Spatially Balanced Latin Squares, our problem becomes a decision problem. Therefore, we first relax some of its constraints and try to minimize the constraint violation. We relax both the balancedness and the Latin square constraints. In case we cannot find a totally balanced Latin square, we would like to balance both the worst case pair of symbols as well as the average over all pairs. Therefore, we penalize the unbalancedness of a square with the term, $b := \sum_{1 \leq k < l \leq n} \left(\bar{\mathrm{d}}(k, l) - \frac{n+1}{3} \right)^4$.

The second term of the objective function penalizes a symbol if it occurs more than once in the same column. Denoting with $f_k(i)$ the number of times that symbol k occurs in column i, the Latin square penalty is defined as $ls := \sum_{1 \leq i,k \leq n} \max\{f_k(i) - 1, 0\}$.

The overall objective then is to minimize $b + \beta\ ls$, whereby β is a variable factor that oscillates during the optimization. It allows us to guide the search towards or away from the search region containing Latin squares. For more details on strategic oscillation we refer the reader to [7, 1].

Neighborhood As with every local search technique, the other fundamental design decision regards the neighborhood. We experimented with different neighborhoods, finding that a rather simple type of moves gives smoother walks and results in better performance than more complicated neighborhoods. In our approach, there is just one simple move allowed: Swap a random pair of symbols in a random row, whereby we only consider such pairs that will result in a change in the number of Latin square constraints that are fulfilled.[2]

2.2 Constraint Programming Approach

In our basic CP model every cell of our square is represented by a variable that takes the symbols as values. We use an AllDifferent constraint [11] over all cells in the same column as well as all cells in the same row to ensure the Latin square requirement. We also keep a dual model in form of the row conjugate that is connected to the primal model via channeling constraints for the quasi-group completion problem [10, 12]. This formulation is particularly advantageous given that by having the dual variables at hand it becomes easier to select a "good" branching variable as well as to perform symmetry checks, as we will see shortly. In order to enforce the balancedness of the Latin squares, we introduce variables for the values $\bar{\mathrm{d}}(k, l)$ and enforce that they are equal to $\frac{(n+1)}{3}$.

Variable Selection As with many discrete problems, it turns out that the selection of the branching variable has a severe impact on the performance of our algorithm. For Latin square type problems it has been suggested to use a strategy that minimizes the options both in terms of the position as well as the value that is chosen. In our problem, however, we must also be careful that we can detect unbalancedness very early in the search. Therefore, we traverse the search space symbol by symbol by assigning a whole column in the row conjugate before moving on to the next symbol. For a given symbol, we then choose a row in which the chosen symbol has the fewest possible cells that it can still be assigned to. Finally, we first choose the cell in the chosen row that belongs to the column in which the symbol has the fewest possible cells left.

Symmetry Breaking In order to avoid that symmetric search regions are explored repeatedly, we implemented Symmetry Breaking by Dominance Detection (SBDD) (see [5, 6]). According to our strategy for the variable selection, we try all different mappings of symbols in the current search node to the symbols of the previously explored nodes. Given that mapping, using the dual model we can easily check in linear time whether there exists a permutation of the rows such that the current search node is dominated. However, since there exist $n!$ different permutations of the symbols, this symmetry check is rather costly. In order to reduce the computational effort, it is important to treat unassigned symbols implicitly, which gives great advantages especially when comparing against previously expanded search nodes that are located high up in the search tree. Still, symmetry breaking is expensive. In the following section we will therefore evaluate whether this enhanced symmetry breaking procedure pays off.

Composition of TBLS We also developed a very promising strategy for generating TBLS instances using as building blocks TBLS instances of smaller orders. Given a

[2] We would like to thank an anonymous reviewer for suggesting this neighborhood!

TBLS instance of order n, our model generates TBLS instances of orders $2n$ (and $3n$) by making 1 (or 2) copies of the initial TBLS instance of order n, and appropriately renaming the symbols of the copy (or 2 copies). In this approach we only manipulate *entire* columns of the building blocks and therefore the number of variables is reduced to the number of columns of the composed TBLS. The domain of each variable in this model corresponds to the different columns of the building blocks.

3 Computational Results

We now present preliminary computational results obtained with our implementations of the local search as well as the constraint programming algorithm. The simulated annealing approaches were implemented in C++ compiled with the gnu g++ compiler version 3.2.2 on an Intel XEON 2.0 GHz CPU and 1.0 GB RAM. The CP approaches were implemented using ILOG Solver 5.1 and the gnu g++ compiler version 2.91 on an Intel Pentium III 550 MHz CPU and 4.0 GB RAM.

Table 1 shows our results for the two approaches. Comparing the two CP variants, the first (CPI) using an initialization of the first row and the second (CPS) using SBDD, we find that sophisticated symmetry breaking does not pay off for the problem sizes that we can tackle so far. Note that we cannot initialize the first row when using the traversal strategy without spending a lot of time in the symmetry checks since then all symbol permutations must be checked from the beginning. Instead, we can initialize the first column in the row dual, thus fixing the traversal of symbol 0.

	Order						
	3	5	6	8	9	11	12
CPI - time	0.01	0.02	0.06	16.14	241	-	-
CPS - time	0.01	0.02	0.09	>300	-	-	-
LSO - time	0.01	0.01	0.36	79.56	153	334	883
LSO - max. dev.	opt	opt	opt	0.25	0.22	0.36	0.5
LSO - av. dev.	opt	opt	opt	0.04	0.11	0.16	0.15
LSN - time	0.01	0.01	0.03	32.03	60.95	246	no LS
LSN - max. dev.	opt	opt	opt	opt	opt	0.36	no LS
LSN - av. dev.	opt	opt	opt	opt	opt	0.14	no LS

Table 1. Comparison of constraint programming (CP-X) based and local search (LS-X) based approaches for the TBLS. Time given in CPU seconds. All values are medians over 10 runs.

Next, we see that the local search approach can also find optimal solutions for orders up to 9 if we do not use strategic oscillation (LSN). However, strategic oscillation (LSO) helps us to obtain feasible Latin square solutions (see order 12 for example), which is why we favor this approach for higher orders for which we are unable to provide totally balanced Latin squares so far. Table 1 also gives the maximum and the average deviation from the perfect balance in these cases.

As mentioned in the previous section, we also developed a CP based model for the generation of spatially-balanced Latin squares by means of *composition of columns of spatially-balanced Latin squares*. In this approach we use spatially-balanced Latin squares of order n as building blocks to produce spatially-balanced Latin squares of order $2n$ or order $3n$. Using such a strategy we were able to generate, for example, a spatially-balanced Latin square of order 18, by composing spatially-balanced Latin squares of order 9. We are currently working on streamlining this approach in order to produce efficient constructions for the generation of totally spatially-balanced Latin square instances of order n, $n \bmod 2 = 0$ or $n \bmod 3 = 0$. While we do not see how the local search approach could be further improved so that it can provide optimal solutions for much larger orders, the idea of composing squares can be stated naturally as a constraint program.

4 Conclusions and Future Work

We present several models for the generation of totally spatially-balanced Latin squares. While it is unclear at this stage how the local search approach could be tuned to give optimal solutions for orders greater than 9, our results with CP based models were very encouraging; We could find totally spatially-balanced Latin instances up to order 18. Moreover, our different CP based models provided us with good insights about the structure of the problem. In fact, we conjecture that totally spatially-balanced Latin squares can be generated using a polynomial time construction, based on a representation that exploits the underlying traversal structure of Latin squares corresponding to matchings in bipartite graphs, as well as the duality between rows, columns, and symbols in a balanced Latin square. We also conjecture that, for certain orders, spatially-balanced Latin squares can be generated by means of composition, in polynomial time. If some symbols are pre-assigned to specific cells of the Latin square, our conjecture is that the problem of deciding if a partially filled Latin square can be completed into a balanced Latin square is an NP-complete problem. We hope that our results will further stimulate research on this interesting and challenging problem.

References

1. A. Anagnostopoulos, L. Michel, P. van Hentenryck, and Y. Vergados. A Simulated Annealing Approach to the Traveling Tournament Problem. In *Proc. CPAIOR'03*, 2003.
2. A. Atkinson and R. Bailey. One hundred years of design of expriements on and off the pages of biometrika. *Biometrika*, 88:53–97, 2001.
3. R. Bose and S. Shrikhande. On the falsity of euler's conjecture about the nonexistence of two orthogonal latin squares of order 4t+2. In *Proc. Nat. Ac. of Sc. 45*. 1959.
4. W. Cochran and G. Cox. *Experimental design*. John Wiley and Sons, Inc., 1950.
5. T. Fahle, S. Schamberger, and M. Sellmann. Symmetry Breaking. In *Principles and practice of Constraint Programming (CP01) Lecture Notes in Computer Science*, pages 93–107. Springer-Verlag, 2001.
6. F. Focacci and M. Milano. Global Cut Framework for Removing Symmetries. In *Principles and practice of Constraint Programming (CP01) Lecture Notes in Computer Science*, pages 77–92. Springer-Verlag, 2001.
7. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
8. C. P. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *J. of Automated Reasoning*, 24(1–2):67–100, 2000.
9. W. Harvey. Symmetry Breaking and the Social Golfer Problem. In *Proc. SymCon'01*, 2001.
10. B. Hnich, B. M. Smith, and T. Walsh. Dual modelling of permutation and injection problems. 2003.
11. J. Regin. A Filtering Algorithm for Constraints of Difference in CSPs. In *Proc. of AAAI*, pages 362–367, 1994.
12. I. J. D. Rodriguez, A. del Val, and M. CebriÃ¡n. Redundant Modeling for the QUasigroup Completion Problem. In F. Rossi, editor, *Principles and practice of Constraint Programming (CP03) Lecture Notes in Computer Science*, pages 288–302. Springer-Verlag, 2003.
13. B. Smith. Reducing Symmetry in a Combinatorial Design Problem. In *Proc. CPAIOR'01*, pages 351–360, 2001.
14. H. van Es. Sources of soil variability. In *Methods of Soil Analysis, Part 4: Physical Properties*. Soil Sci. Soc. Am, 2002.
15. H. van Es and C. van Es. The spatial nature of randomization and its effects on outcome of field experiments. *Agron. J.*, (85):420–428, 1993.
16. H. Zhang. Specifying latin square problems in propositional logic. In *Automated Reasoning and Its Applications*. MIT Press, 1997.