Streamlining Local Search for Spatially Balanced Latin Squares

Casey Smith, Carla Gomes, and Cesar Fernandez
Computer Science Department
Cornell University
{casey,gomes}@cs.cornell.edu, cesar@eup.udl.es

Abstract

Streamlined constrained reasoning powerfully boosts the performance of backtrack search methods for finding hard combinatorial objects. We use so-called spatially balanced Latin squares to show how streamlining can also be very effective for local search: Our approach is much faster and generates considerably larger spatially balanced Latin squares than previously reported approaches (up to order 35; the previous best results could only generate solutions up to order 18). We also provide a detailed characterization of our streamliner and solution topology for small orders. We believe that streamlined local search is a general technique suitable for solving a wide range of hard combinatorial design problems.

1 Introduction

The idea underlying streamlining constraints [Gomes and Sellmann, 2004] is simple but powerful: In order to increase the effectiveness of propagation, "streamlining constraints" partition the solution space into different portions with different global properties. Since in general the streamlined constraints are not implied by the original problem constraints, the resulting search and solution spaces are subsets of the original spaces. Gomes and Sellmann showed the effectiveness of streamlined constraints for solving hard combinatorial design problems, in particular spatially balanced Latin squares (SBLS, defined in Section 2). Using streamlined backtrack search methods, they solved considerably larger SBLS problem instances than with local search: up to order 18, while a local search method applied to the original problem could only generate solutions up order 9.

This paper shows how streamlining can also be effective for local search. We note that while the addition of streamlining constraints is straightforward in backtrack search methods, it is not always the case for local search methods, since constraints are in general enforced via the objective function. The key is to consider streamlining constraints that lead to an alternate neighborhood representation. We applied streamlined local search to the problem of generating spatially balanced Latin squares. Our streamliner defines a basic swap neighborhood as a column permutation of the input Latin square. This provides additional interesting structural properties, making our local search procedure scale up considerably better than previously reported approaches. We solve instances up through order 35; the previous best results could only generate solutions up to order 18. Moreover, our streamliner provides useful intuitions on a construction for spatially balanced Latin squares. We believe that streamlined local search is a general technique suitable for finding objects with intricate combinatorial constraints.

2 Spatially Balanced Latin Squares

A Latin Square of order n is an n by n grid where each of the n^2 cells in the grid is assigned one of n symbols such that each symbol appears exactly once in each column and each row. Creating a Latin square is by no means difficult. We can define a *cyclic construction* as an order n Latin square where the element in row i and column j has the value $(i + j)$ mod n (see Figure 1). The first row will have all the symbols in order: [1 2 3 ... n]. Each subsequent row is the same as the previous row with all the values shifted one location to the left, wrapping the first element to the end. Thus, the second row will be [2 3 ... n 1]; the third will be [3 ... n 1 2]; and so on. Note that the columns follow the same pattern: the first column is [1 2 3 ... n]^T; the second is [2 3 ... n 1]^T; and the third is [3 ... n 1 2]^T.

One challenge comes in generating *spatially balanced* Latin squares (SBLS) where the sum of the distances between two symbols across all rows is constant for all pairs of symbols. First, we define $d_i(j, k)$ to be the distance between symbols j and k in row i. This distance is calculated as the absolute difference of the column indices of where symbols j and k appear in row i. For example, if row i were [3 5 1 2 4], then $d_i(5, 2)$ would be $|2 - 4| = 2$. Then we can define the total distance $d(j, k)$ to be

$$d(j,k) = \sum_{i=1}^{n} d_i(j,k). \quad (1)$$

A SBLS requires that all pairs of symbols have the same total distance:

$$\forall i \neq j, k \neq l : d(i,j) = d(k,l). \quad (2)$$

SBLS are useful for agronomic field experiment design [van Es and van Es, 1993] and are compelling constraint satisfaction problems because of their high degree of structure. As

1	2	3	4	5		1	4	5	3	2
2	3	4	5	1		2	5	1	4	3
3	4	5	1	2		3	1	2	5	4
4	5	1	2	3		4	2	3	1	5
5	1	2	3	4		5	3	4	2	1
(A)						(B)				

Figure 1: (A) A cyclic Latin square (order 5). (B) A SBLS where $\forall j, k : d(j, k) = 10$.

shown in [Gomes *et al.*, 2004], $d(j, k) = \frac{n(n+1)}{3}$ for all $j \neq k$ in any SBLS, implying that no SBLS exist for orders n such that $n \bmod 3 = 1$.

3 Streamlined Local Search

Gomes and Sellmann showed that a standard local search approach performs remarkably poorly at finding SBLS. We hypothesize that this is due to the enormity of the search space and the global nature of the constraints. The search space is so big and the solutions are so few that the odds of local search finding a solution are very slim for all but the smallest orders. If we can cleverly restrict the search space to a small subspace which contains a higher density of solutions, we can increase the odds that local search will find a solution.

Consider the subspace resulting from starting with a Latin square and restricting the search space to be a column order permutation of the original square. That is, take a Latin square and swap entire intact columns to permute the order in which the columns appear. Since this reordering of the columns will never make the square non-Latin (all symbols will appear in all rows and in all columns exactly once), we can try permuting the column order to make the Latin square spatially balanced. Note that the space of column permutations of a particular Latin square is vanishingly small compared to the entire space of Latin squares. The goal is to find an initial Latin square such that this small subspace is dense with SBLS. Cyclically constructed Latin squares empirically provide such a subspace.

If the initial Latin square is generated by the cyclic construction, some structural properties can be exploited to speed the search. First, observe that there will be at most $\lfloor \frac{n}{2} \rfloor$ unique values of the total distance among the $\frac{n(n-1)}{2}$ pairs of symbols. To see this, define

$$d_{mod}(j, k) = \min(|j - k|, |j - k + n|, |j - k - n|). \quad (3)$$

Thus, $d_{mod}(j, k)$ is the distance between symbols j and k in symbol space, modulo the number of symbols (n). For example, for an order 6 Latin square, $d_{mod}(2, 4) = 2$ because the shortest path from 2 to 4 is by counting 2 3 4, revealing a distance of 2. Similarly, $d_{mod}(1, 5) = 2$ because the shortest path from 1 to 5 is by counting 5 6 1, revealing a distance of 2. Examining the cyclic construction reveals that pairs of symbols with identical values of $d_{mod}(j, k)$ will have identical values of $d(j, k)$, even if the column order is permuted. Thus since there are $\lfloor \frac{n}{2} \rfloor$ different nonzero values of $d_{mod}(j, k)$, there are at most $\lfloor \frac{n}{2} \rfloor$ different total distances between pairs of symbols for any column order permutation of the cyclic

A	B	C	D	E
1	2	3	4	5
2	3	4	5	1
3	4	5	1	2
4	5	1	2	3
5	1	2	3	4

(A)

A	E	C	D	B
1	5	3	4	2
2	1	4	5	3
3	2	5	1	4
4	3	1	2	5
5	4	2	3	1

(B)

Col	A	B	C	D	E
A	-	X	Y	Y	X
B	-	-	X	Y	Y
C	-	-	-	X	Y
D	-	-	-	-	X
E	-	-	-	-	-

(C)

Figure 2: (A) A cyclic Latin square (order 5). (B) A column permutation of the cyclic Latin square. Note that no matter how the columns are reordered,
$d(1, 2) = d(2, 3) = d(3, 4) = d(4, 5) = d(5, 1) =$
$|A - B| + |B - C| + |C - D| + |D - E| + |A - E|$ and
$d(1, 3) = d(2, 4) = d(3, 5) = d(4, 1) = d(5, 2) =$
$|A - C| + |B - D| + |C - E| + |D - A| + |E - B|$
(C) A matrix showing how the intra-column distances contribute to the pair distances. The 10 pairs of symbols correspond to group X ((1,2), (2,3), (3,4), (4,5), (5,1)) and group Y ((1,3), (2,4), (3,5), (4,1), (5,2)).

construction. As an example, if we check the total distance between symbols 1 and 2, we do not need to check the distance between any other consecutive symbols, i.e., 2 and 3, 3 and 4, and so on because they will all have identical values (see Figure 2). Next, note that since the columns always remain intact no matter how their order is permuted, we need only store the first row as we search for SBLS. From the first row, we can calculate the $d(j, k)$ corresponding to each of the $\lfloor \frac{n}{2} \rfloor$ possibly distinct $d(j, k)$. We also observe that we can fix the first column to be headed by symbol 1 without decreasing our probability of finding a solution: there will be an equal number of solutions with 1 heading the first column as with any other number.

These observations naturally lead to a gradient descent algorithm. First, we define the *imbalance* of a square to be

$$\text{imbalance} = \sum_{i<j} \left| d(i, j) - \frac{n(n+1)}{3} \right|. \quad (4)$$

That is, the imbalance is the sum over all pairs of how far from ideal spacing the total spacing of the pair is. Starting with a random permutation of the first row, we swap the location of two symbols such that it decreases the imbalance of the square. When we reach a local minimum, we randomize and try again.

Algorithm 1 SBLS Streamlined Local Search

1: Generate a Latin Square of order n by the cyclic construction
2: Randomly permute the order of the columns
3: Select a column and try swapping it with every other column
4: Keep the swap which minimizes the imbalance
5: If stuck, then randomize the order of the columns
6: Repeat from step 3

Finally, we note that we can generate solutions which are symmetric across the main diagonal. Since spatial balance is defined as the sum of distances within rows, swapping intact

order	6	8	9	11	12	14	15	17	18	20	21	23	24	26	27	29	30	32-35
CP	0.06	16	241															
SS	0.05	0.88	0.91	9.84	531	5K												
CS	0.02				14.4				107K									
CCCP	4e-5	3e-4	1e-4	1e-3	1e-3	0.02	0.01	0.25	2.3	16	16	104	281	609	4K	43k	≈160K	≈1.2M

Table 1: Solution times are given in seconds. CP, SS, and CS are the times to find the first solution via standard constraint programming, symmetric streamlining, and composition streamlining as reported by Gomes and Sellmann [2004] as run on a 550 MHz Pentium III machine. CCCP is the cyclic construction column permutation method presented here. Times given are the mean time to find a solution averaged across several independent runs on a 1.0 GHz Pentium III machine. Note that although our processor was only twice as fast, our performance is orders of magnitude better. Note that the largest solution from previous methods was only order 18, while our method can find solutions up to order 35. Additionally, no previous method had found solutions for orders 15 and 17.

1	2	3	4	5	6		1	3	4	2	6	5		1	3	4	2	6	5
2	3	4	5	6	1		2	4	5	3	1	6		3	5	6	4	2	1
3	4	5	6	1	2		3	5	6	4	2	1		4	6	1	5	3	2
4	5	6	1	2	3		4	6	1	5	3	2		2	4	5	3	1	6
5	6	1	2	3	4		5	1	2	6	4	3		6	2	3	1	5	4
6	1	2	3	4	5		6	2	3	1	5	4		5	1	2	6	4	3
	(A)							(B)							(C)			

Figure 3: (A) A cyclic Latin square (order 6). (B) The same cyclic Latin square with the columns permuted to form a SBLS. (C) The same SBLS as in (B), but with the rows reordered the same as the columns to make a symmetric SBLS.

rows will not affect the balancing. If we reorder the rows in the same way as the columns, the solution will become symmetric across the main diagonal and spatially balanced in terms of both the intra-row and intra-column distances (see Figure 3).

4 Empirical Results

The best performance on finding SBLS was reported by Gomes and Sellmann [2004], so we compare our results to theirs. Our streamlining is much simpler than theirs and performs dramatically better. They could only generate solutions up to order 18 (and were unable to find solutions for orders 15 and 17). We have generated SBLS for all possible orders up through 35. Times in seconds are given in Table 1.

We also tried using our streamlining with complete backtrack search. While it improves performance over previous methods, the local search method presented here performs considerably better. In addition, we tried composing solutions from smaller cyclic Latin squares using local search, but the performance was slightly worse than the simple streamliner presented here.

Although we have only discussed permuting the columns of the cyclic construction, in principle, we could permute the columns of any Latin square in an attempt to find SBLS. However, most arbitrary random Latin squares (such as those generated by Jacobson and Matthews [1996]) *cannot* be permuted to be spatially balanced. This observation is remarkable, given that cyclic constructions are much more imbalanced than an average random Latin square. For instance, the imbalance of an order 30 cyclic Latin square is 46920 while the imbalance of a typical order 30 random Latin square is only around 12000.

Through exhaustive enumeration, we analyzed the solution space of SBLS for small orders. Interestingly, we found that for orders 6 and 8, all the SBLS can be generated by row permuting and/or one-to-one symbol remapping (that is change all instances of symbol i to symbol j, symbol k to symbol l, and so on) the SBLS obtained as column permutations of the cyclic construction. However, for higher orders, there exist some solutions which cannot be derived in that way. We are currently studying the topology of the solution space for higher orders trying to identify relationships among solutions. We are also studying the matrix of distances (as in Figure 2). We believe that the characterization of solution relationships and the distance matrix provide valuable clues for the design of a construction for generating SBLS of arbitrary size.

5 Conclusions

We describe a competitive streamlined local search approach for generating SBLS. We believe that streamlined local search is a general technique, effective for finding objects with intricate combinatorial properties. We hope that this work will inspire other researchers interested in solving hard combinatorial design problems.

References

[Gomes and Sellmann, 2004] C. Gomes and M. Sellmann. Streamlined constraint reasoning. In *CP 2004*, 2004.

[Gomes *et al.*, 2004] C. Gomes, M. Sellmann, C. van Es, and H. van Es. The challenge of generating spatially balanced scientific experiment designs. In *CPAIOR 2004*, 2004.

[Jacobson and Matthews, 1996] Mark Jacobson and Peter Matthews. Generating uniformly distributed random latin squares. *J. of Combinatorial Designs*, 4(6):405–437, 1996.

[van Es and van Es, 1993] Harold van Es and Cindy van Es. The spatial nature of randomization and its effects on the outcome of field experiments. *Agronomy Journal*, 85:420–428, 1993.