From Streamlined Combinatorial Search to Efficient Constructive Procedures

Ronan Le Bras and **Carla P. Gomes** and **Bart Selman**
Computer Science Department
Cornell University
Ithaca, NY 14853

Abstract

In recent years, significant progress in the area of search, constraint satisfaction, and automated reasoning has been driven in part by the study of challenge problems from combinatorics and finite algebra. This work has led to the discovery of interesting discrete structures with intricate mathematical properties. While some of those results have resolved open questions and conjectures, a shortcoming is that they generally do not provide further mathematical insights, from which one could derive more general observations. We propose an approach that integrates specialized combinatorial search, using so-called streamlining, with a human computation component. We use this approach to discover *efficient constructive procedures* for generating certain classes of combinatorial objects *of any size*. More specifically, using our framework, we discovered two complementary efficient constructions for generating so-called Spatially Balanced Latin squares (SBLS) of any order N, such that 2N+1 is prime. Previously constructions for SBLSs were not known. Our approach also enabled us to derive a new lower bound for so-called weak Schur numbers, improving on a series of earlier results for Schur numbers.

Introduction

In recent years, significant progress in the area of search, constraint satisfaction, and automated reasoning has been driven in part by the study of challenge problems from combinatorics and finite algebra. This work has led to the discovery of interesting discrete structures with intricate mathematical properties (e.g., (Slaney, Fujita, and Stickel 1993; Fujita, Slaney, and Bennett 1993; Zhang and Hsiang 1994; McCune 1997; Kouril and Franco 2005)). While such results often resolve open questions and conjectures (such as "Do conjugate-orthogonal quasigroups of certain orders exist?"), a shortcoming of these results is that they generally do not provide further mathematical insights, from which one could derive more general observations. This is because the search procedures do, in essence, a form of proof by demonstration, i.e., identify a combinatorial object of a given size, or show that no such object exists, by an exhaustive case analysis. In fact, the inherent lack of further mathematical insights has been a common criticism of mathematical results obtained using search and inference procedures. For example, the famous 1974 computer assisted proof of the four color theorem (Appel and Haken 1977) is still viewed with reservations by many mathematicians for not providing deeper mathematical insights.

Similarly, although the 1996 proof of Robbins conjecture using an equational theorem prover can actually be checked step-by-step by hand (McCune 1997; Dahn 1998), researchers in the area claim it cannot be "understood" in the usual sense that a mathematical proof is understood, thus again not providing much further insight.

To address this concern, we will introduce a general framework that will enable us to obtain efficient constructive procedures, i.e. polynomial-time algorithms that take as input a size parameter N, and generate a certain combinatorial object of size N. Our approach combines specialized combinatorial search, using so-called streamlining, with human insights, in a complementary, iterative approach. Our work is in part inspired by the exciting new area of human computation, where it is acknowledged that for certain tasks, particularly those involving visual (pattern recognition) abilities, humans still clearly outperform fully automatic approaches (Law and von Ahn 2011). However, instead of "simply" invoking human computation to obtain results, we propose a setting where combinatorial search complements the human component and there is an iterative process to uncover the constructive procedure. So, in the process, computer and human truly complement each other.

We propose our work as a broader framework for mathematical discovery and reasoning. Although fully automated reasoning procedures have been used to obtain several new results in mathematics, it appears that some further mathematical knowledge or intuition is still required to assist the automated process. For example, the HR system (Colton and Miguel 2001; Charnley, Colton, and Miguel 2006) has found many interesting new concepts (e.g., integer sequences), but was not devised to discover constructive procedures. An integrated human computation component is therefore a promising approach in this context. For alternative approaches to mathematical discovery, see e.g., (Colton, Bundy, and Walsh 1999; Herwig et al. 2007).

To demonstrate the effectiveness of our framework, we

						Pair Symbol Distance (per row)				
						(1,2)	(1,3)	(2,4)	(2,6)	(5,6)
1	**2**	**3**	**4**	**5**	**6**	1	2	2	4	1
2	**4**	**6**	**5**	**3**	**1**	5	1	1	1	1
3	**6**	**4**	**1**	**2**	**5**	1	3	2	3	4
4	**5**	**1**	**3**	**6**	**2**	3	1	5	1	3
5	**3**	**2**	**6**	**1**	**4**	2	3	3	1	3
6	**1**	**5**	**2**	**4**	**3**	2	4	1	3	2
Total Row Distance (pair)						14	14	14	14	14
Average Row Distance (pair)						2.33	2.33	2.33	2.33	2.33

Figure 1: Spatially balanced Latin square of order 6 and corresponding row distances, total distance, and average distance for several example pairs of numbers (or symbols).

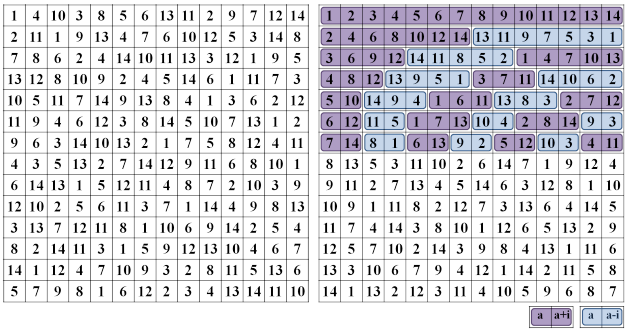

Figure 2: **Left:** Typical SBLS (order 14) obtained with general constraint-based search. **Right:** SBLS (order 14) whose construction was discovered by our procedure. Each row consists of alternations of increasing and decreasing sequences with increment/decrement equal to the row index. (Pattern highlighted for the top seven rows. Pattern continues but is hidden. See procedure `SBLS-Sequence`.) The resulting pattern for the last row (and last column due to symmetry) is quite striking: a decreasing sequence $(14 \ldots 8)$ interleaved with an increasing sequence $(1 \ldots 7)$.

have to show success on problems that lie beyond the reach of any current fully automated inference procedure *and* are open questions of interest to the mathematical community. We present new results in two domains: (1) Spatially Balanced Latin Squares (SBLS), and (2) Schur numbers. Specifically, we will introduce *the first constructive procedure* for SBLS. That is, we will present a polynomial time procedure that generates an SBLS of any size N, such that $2N + 1$ is prime. No such procedure was known to exist. In fact, such Latin squares were not known to exist beyond order 35 (Smith, Gomes, and Fernandez 2005). In addition, we provide a *new lower-bound* on so-called weak Schur numbers by identifying new structural patterns. Schur numbers are closely related to Ramsey theory, and are a notoriously hard area of combinatorics.

Framework and Preview of Results

In this section, we provide an introduction to our approach, and a brief preview of results. To ground our description in an example domain, we first introduce the notion of spatially balanced Latin squares.

Spatially balanced Latin squares — A Latin square of order n, is an n by n grid with each grid cell containing a number from $1, \ldots, n$, without any repetitions in a row or column. In combinatorics, a Latin square represents the multiplication table of a quasigroup. A spatially balanced Latin square (SBLS) poses further constraints on the cells (Es and Es 1993). For each pair of numbers (i, j) in a row, one can assign a distance based on the number of cells that separates them (literally the absolute difference between the column indices of the cells containing i and j). For example, in the first row in Fig. 1, the distance between the pair of numbers $(2, 4)$ is 2. We can now calculate the average distance for this pair taken over all rows, which is $14/6 = 2.33$. If the average distance between all possible pairs of numbers is the same, we say the Latin square is *spatially balanced*. Fig. 1 gives an SBLS of order 6. The figure gives the distance for a few pairs but note that all $\binom{6}{2} = 15$ pairs have average distance 2.33.

SBLSs were first introduced in the context of experiment design, in particular in agronomics and more recently in gene array design (Es and Es 1993; Es et al. 2007). Consider testing the effectiveness of different soil treatments from among n possibilities, with one treatment per cell. Since the effects of different treatments in cells near each other within a row may introduce subtle correlations in the outcomes, one would like to keep all pairs of treatments at the same distance when averaged over all rows in the experimental grid. (We assume rows are well-separated; doubly spatially balanced designs also balance within columns.) Reasonable size agronomic experiments can have dozens of tests in each row, while gene arrays are often even larger. So, researchers would like to design relatively large spatially balanced Latin squares.

The requirement of being *spatially balanced* is clearly a rather intricate global constraint over the Latin square and not many SBLSs were known to exist. Over the years, it has provided a compelling combinatorial search challenge, leading to the development of several new search ideas (Gomes and Sellmann 2004; Gomes et al. 2004; Neveu, Trombettoni, and Glover 2004). The largest square reported in the literature was of order 35 (Smith, Gomes, and Fernandez 2005). Prior to that, no SBLS larger than order 18 was known to exist. Two open questions arose (Gomes and Sellmann 2004): (1) *whether SBLS would exist for arbitrary large orders*, and, if so, (2) *whether there was an effective way of constructing them*. In this work, we answer both questions in the affirmative.

Framework for discovering constructive procedures — We developed a general approach for discovering constructive procedures. Note that search-based solution techniques in combinatorial design generally provide specific solutions (e.g., a SBLS of order 35) but, as discussed earlier, such methods do not provide any clear insights into a possible larger solution. In fact, when we look at a solution obtained by a search procedure, there is often no structure visible

whatsoever. See Fig. 2 (left) for an example SBLS of order 14. This does not mean that there is no hidden structure in the solution. Such structure can easily be hidden because each solution is part of an exponentially large equivalence class that can be obtained by permuting the numbering of the values in the cells or by permuting rows. However, although symmetry breaking is a standard practice in systematic constraint-based searches (Gent and Smith 2000; Flener et al. 2009, e.g.) and allows to reduce a large equivalence class to one representative member, it might prevent the discovery of structure by ruling out the solutions that exhibit a pattern. Alternatively, one can try to generate many different solutions, hoping to see some particular pattern in at least one of them. Unfortunately, even the basic permutation classes are too large. E.g., for $n = 14$, we do not see any kind of structure in thousands of examples. Another strategy is to go to smaller size solutions, and then consider all possible solutions or at least a good fraction of them. This is feasible for small sizes, e.g., $n = 3$, or 5, with respectively, 12, and 5760 solutions. (Note no order 4 SBLS exists.) In one or more of those solutions, one may be able to spot some regularity. It is not clear, however, whether those regularities are accidental or scale up to larger solutions.

In order to study this, we need to generate larger solutions that contain the proposed regularity. A particularly effective combinatorial search strategy, called streamlining (Gomes and Sellmann 2004), allows us to do so. (For a related search technique, called "tunneling," see (Kouril and Franco 2005).) In streamlining, one can add specific desired regularities (e.g., symmetries or partial patterns) to the search engine, which then proceeds to search for solutions with those regularities. If the streamlined search does not give a larger size solution, the proposed regularity is quite likely accidental and we look for a new pattern in the small-scale solutions. On the other hand, if the streamlined search can find larger size solutions, we proceed by generating a number of new solutions for a larger size (e.g., $n = 6$ or 8, 9, 11) that all contain the proposed structural regularity. We now have larger size solutions with some basic regularity. Moreover, new regularities often reveal themselves at the larger sizes. We then add these new regularities to the search and try to find yet larger solutions.

Below, we give a template of our general framework that leverages and further extends the concept of *streamlined* combinatorial search, coupling it with a human computation component, in a complementary, iterative approach. Using this approach, we discovered two complementary efficient constructive procedures that generate spatially balanced Latin squares for arbitrary large sizes. (Not all orders allow for SBLS, as we will discuss below.)

Fig. 2 (right) gives the result of one of our constructive procedures, called `SBLS-sequence` (see Alg. 2), for a SBLS of order 14. In contrast with the left panel, structure is clearly visible. Streamlined-search was used to first identify this pattern; the construction procedure is a more direct representation of the streamliners. The solution consists of alternations of increasing and decreasing arithmetic progressions (i.e., sequences of the form a, $a + \delta$, $a + 2\delta$, etc.) with the increment (decrement) in each row equal to the row index. The starting values of the progressions also follow a well-defined pattern, as discussed below.

$\mathcal{O} \leftarrow \emptyset$; // Conjectured streamliners
$\Gamma \leftarrow \emptyset$; // Search streamliners
$\rho \leftarrow \rho_0$; // Search parameter
$\mathcal{S} \leftarrow \emptyset$; // Solutions found
$\tau \leftarrow$ *false*; // Timeout flag
repeat
  Solve$(P_\rho, \Gamma, t) \rightarrow (\mathcal{S}', \tau)$; // Search for new solutions
  if $\mathcal{S}' \cap \mathcal{S} \neq \emptyset$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}'$; // Case 1: successful search
    Analyze$(\mathcal{S}) \rightarrow \mathcal{O}'$; // Conjecture new streamliners
    $\mathcal{O} \leftarrow \mathcal{O} \cup \mathcal{O}'$;
    $\rho \leftarrow \rho + 1$;
  else if τ *is true* **then**
    Select $\Gamma' \subseteq \mathcal{O}$; // Case 2: timed-out failed search
    $\Gamma \leftarrow \Gamma \cup \Gamma'$; // Strengthen streamliners
  else
    Select $\Gamma' \subseteq \Gamma$; // Case 3: exhaustive failed search
    $\Gamma \leftarrow \Gamma \setminus \Gamma'$; // Weaken streamliners
    $\rho = max\{\rho : \mathcal{S}(\Gamma) \cap \mathcal{S}(P_\rho) \neq \emptyset\}$+1;
    Select $\Gamma'' \subseteq \Gamma'$; // Find next parameter of interest
    $\mathcal{O} \leftarrow \mathcal{O} \setminus \Gamma''$; // Drop unpromising streamliners
until $\mathcal{O} = \emptyset$;

Algorithm 1: `Discover-Construction` procedure for a given problem P, with parameter set ρ and timeout t.

Since the SBLS is obtained with an effective constructive procedure, one can easily generate spatially balanced Latin squares of arbitrary large orders. As noted earlier, the best previously known search-based strategy could find solutions up to order 35. We conjecture that many combinatorial design problems may have such hidden effective construction methods.

Using our framework, we also obtained new results for Schur numbers, another notoriously hard combinatorial problem. Details are given below.

Discovering Constructive Procedures

Consider a combinatorial problem P_ρ, with k integer parameters represented by ρ (i.e., $\rho \in \mathbb{N}^k$). We denote the set of solutions of this problem with $\mathcal{S}(P_\rho)$. For example, in the SBLS problem, ρ is a single parameter, n, the order of the Latin square. $\mathcal{S}(P_6)$ corresponds to all SBLSs of order 6.

Our goal is to design an efficient, constructive procedure C that generates a non-empty solution set, $C(P_\rho) \subseteq \mathcal{S}(P_\rho)$ for some $\rho \in \mathbb{N}^k$. We define the *support* D_C of the construction as $D_C = \{\rho \in \mathbb{N}^k : C(P_\rho) \neq \emptyset\}$. The construction C is *novel* if D_C intersects with the set of parameters for which no solution is known and is *generalizable* if $|D_C| = \infty$. So, a novel construction provides us with solutions for new parameter values, and a generalizable construction ensures a construction for an infinite range of paramater values.

We propose a general framework for discovering constructive procedures. This framework makes extensive use of streamliners. A *streamliner* (Gomes and Sellmann 2004) corresponds to a set of constraints that can be evaluated and

propagated on *partial* solutions of P_ρ. Typically, a streamliner intentionally discards entire subspaces of the search space (which potentially contain solutions) in order to focus on a highly structured subspace and therefore boost constraint reasoning. When visualizing a set $\mathcal{S}$ of solutions, a user might conjecture a set $\mathcal{O}$ of streamliners that could potentially generalize to higher parameter values (procedure $Analyze$). Moreover, we define $\mathcal{S}(\Gamma)$ as the subset of solutions of $\mathcal{S}$ that satisfy the set Γ of streamliners.

The procedure `Discover-Construction` illustrates our framework. The procedure formalizes the approach we discussed in the introduction. Below we provide additional details using the SBLS example. The procedure is an iterative process. For a given set of parameter values ρ, a set of streamliners Γ and a 'feedback' timeout t, the search (procedure $Solve$) returns a set $\mathcal{S}'$ of solutions as well as a Boolean τ indicating whether the search timed out. If a streamliner is successful (i.e., previously unseen solutions were discovered), the search proceeds to a higher parameter value. Otherwise, if the search led to a time-out, the user may strengthen the current streamliner. If not, the user may weaken the current set of streamliners by selecting a subset Γ' of streamliners not to be considered in the next search. In addition, the user may decide to no longer consider some of these streamliners in any subsequent search. Finally, the user resumes the search, starting with the parameter value for which no solution satisfies the current set of streamliners. Note that although Alg. 1 assumes an *a priori* known total order on the parameters ρ, with ρ_0 the minimum parameter value, it can be easily extended to a partial order. $\rho + 1$ the successor of ρ according to this order.

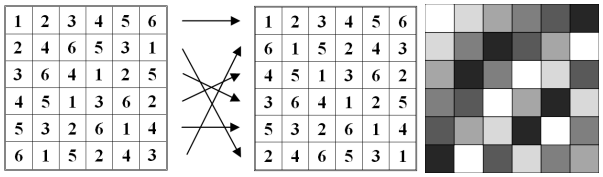

Figure 3: Different representations of SBLSs. **Left:** SBLS of order 6 obtained with the `SBLS-sequence` construction. **Center:** Its column conjugate. The arrows highlight that the SBLS and its column conjugate are identical up to a row permutation in this construction. **Right:** Color visualization of the SBLS on the left.

Discovering Efficient Constructive Procedures for Spatially Balanced Latin Squares

Early on in our research developing a framework for human-guided streamlined combinatorial search to discover constructive procedures, the importance of considering *different representations and visualizations* for a given combinatorial object became clear. For example, in order to gain insights into the underlying combinatorial structure of SBLSs, it is crucial to visualize Latin squares *numerically*, as opposed to symbolically (using, e.g., colors or patterns). The importance of considering different representations for modeling constraint satisfaction problems (CSP) has also been reported in the literature (Rossi, Beek, and Walsh 2006). For example, in order to scale up solutions for the Latin square completion problem[1] it is critical to combine redundant Latin square encodings to enhance constraint propagation. In particular, one can consider the Latin square *conjugates*. Given a Latin square represented as a set of triples (rcs), we obtain its 6 conjugates by reordering the items of the triple (one of the conjugates corresponds to the original Latin square, by doing nothing). For example, given Latin square L, its column conjugate L^c is obtained by reordering the triple (rcs) as (rsc), which corresponds to assigning to cell (r, s) of Latin square L^c the column number c (i.e., the column number in which symbol s appears in row r of the original Latin square L). In Figure 3, we show different representations and visualizations of an SBLS of order 6. The center panel provides the column conjugate of the SBLS depicted in the left panel.

Table 1: Number of SBLSs generated by size and streamliner. A 60-second time-out was used. Bold indicates exhaustive search.

Streamliners	5	6	8	9	11	14
$\Gamma_1 = \emptyset$	**5760**	15878	-	-	-	-
$\Gamma_2 = \Gamma_1 \cup \{$*Symmetric*$\}$	**240**	8447	714	43	-	-
$\Gamma_3 = \Gamma_2 \cup \{$*Reduced*$\}$	**2**	**14**	**14**	51	-	-
$\Gamma_4 = \Gamma_3 \cup \{$*Columns* 2 & n$\}$	**1**	**1**	**2**	**1**	**1**	-
$\Gamma_5 = \Gamma_4 \cup \{$*Multiples of* $i\}$	**1**	**1**	**2**	**1**	**1**	1

We applied our `Discover-Construction` procedure to the SBLS problem. For the *Solve* step, we use IBM Ilog Solver. Our CSP SBLS model encodes both the original square and its column conjugate, using channelling constraints to link the variables in both representations. Our model enforces the global ALLDIFF constraint on the rows and columns of the original Latin square and its conjugate. The balancedness of the square is expressed through the column conjugate. Our CSP encoding is similar to the one proposed in (Gomes and Sellmann 2004).

Fig. 4 depicts the user interface of our system, applied to the SBLS problem. The interface allows the user to streamline the search, specify new streamliners, and visualize the solutions found (given the timeout) for the selected set of streamliners. Sequentially adding streamliners allows us to generate combinatorial objects of increasing sizes. As illustrated in Table 1, the search with no additional streamliners ($\Gamma_1 = \emptyset$) provides SBLSs of order up to 6. Note that this interface is not specific to the SBLS problem, and when addressing a new problem, a user needs only design one custom control that displays a solution of this problem, would it be a grid, a series, or even a graph. For example, we successfully applied our framework to the problem of finding *highly balanced rows* (see below), and the *rectangle-free coloring of grids problem* (Fenner et al. 2010) and the *Van der Waerden numbers* (Herwig et al. 2007).

[1]This problem is also known as the Quasigroup Completion Problem (QCP) since a Latin square corresponds to the multiplication table of a quasigroup.

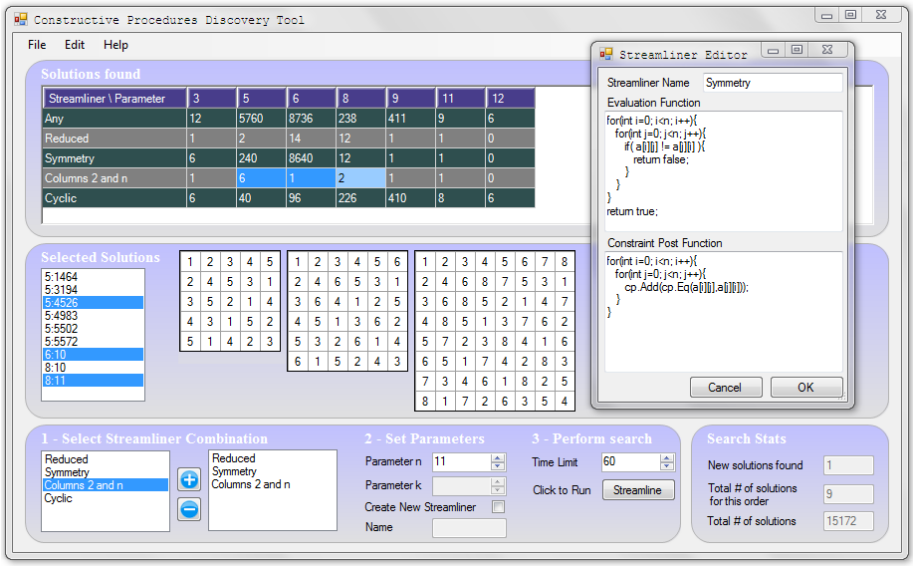

Figure 4: User interface for human-guided streamlined search to discover constructive procedures (SBLS problem).

SBLS-Sequence — Imposing reduced form (first row and column set to the sequence $1 \ldots n$) and diagonal symmetry allows us to generate SBLSs of orders 8 and 9. At this point, a pattern on the second and last columns (*Columns* $2 \,\&\, n$) becomes noticeable (see Figure 2, right panel, and Figure 4). Imposing this additional streamliner allows us to generate an SBLS of order 11 and formulate the conjecture that each row i begins with the multiples of i, in increasing order. This streamliner allows us to find an SBLS of order 14. At this point a pattern of increasing and decreasing sequences for the SBLSs of orders 3, 5, 6, 8, 9, 11, and 14 is clearly noticeable. It only remains to figure out how to switch from one sequence to the next. In fact, the *bounding conditions* become more noticeable as the size of the SBLS increases. As illustrated in Figure 2 (right), the pairs of values that delimit an increasing and a decreasing sequence are $(10, 14)$ and $(11, 13)$ in row 5, and $(12, 11)$, $(13, 10)$ and $(14, 9)$ in row 6. Overall, the sum of the values of each pair appears to be constant within a row, and to decrease by 1 from one row to the next. This final observation leads us to discover the efficient `SBLS-sequence` construction (Alg. 2). By creating Latin squares of different orders with this procedure, we observed that for orders N, where $2N + 1$ is prime, the Latin square is indeed spatially balanced. We also were able to proof this formally, as discussed below.

for *row* $i = 1, \ldots, N$ **do**
  $k = 1$; // Sequence number
  $j = 1$; // Column index
  $a_{i,j} = i$; // First symbol of row i
  while $j < N$ **do**
    if k *is odd* **then** // Odd sequence
      while $a_{i,j} + i \leq N$ *and* $j < N$ **do**
        $a_{i,j+1} = a_{i,j} + i$;
        $j = j + 1$;
    else // Even sequence
      while $a_{i.j} - i \geq 1$ *and* $j < N$ **do**
        $a_{i,j+1} = a_{i,j} - i$;
        $j = j + 1$;
    if $j < N$ **then** // Switch sequence
      if k *is odd* **then**
        $a_{i,j+1} = 2N + 1 - i - a_{i,j}$;
      else
        $a_{i,j+1} = i - a_{i,j}$;
      $k = k + 1$;
      $j = j + 1$;

Algorithm 2: `SBLS-sequence` procedure for SBLS of order N, when $2N + 1$ is prime.

The intuition behind the `SBLS-sequence` construction (Alg. 2) is to fill every row i $(i = 1, ..., n)$ of the square, starting with i and successively adding i to the previous cell, which we call the first sequence in row i. Once we reach the largest multiple v of i that is lower or equal to n, we perform what we call an *upper bounce*: the next cell is assigned symbol $2n + 1 - i - v$. This symbol is the starting point for the second sequence. We progress downwards by successively subtracting i until we reach a value v' that is lower or equal to i. At this point, we perform a *lower bounce*: the next cell is assigned symbol $i - v'$. This symbol represents the starting point of the next sequence, and we repeat these steps until the row is full. The complexity of the `SBLS-Sequence` procedure is $\mathcal{O}(n^2)$.

In the following, we denote by A_n the squares of order n generated by our `SBLS-Sequence` construction, and a_{ij} the symbol in row i and column j of the square A_n. A_5,A_6, and A_8 are depicted in Figure 4 and A_{14} in Figure 2, right panel. We can also represent the squares produced by the `SBLS-Sequence` construction using the following closed form expression.

Definition 1 (`SBLS-SEQUENCE`)**.** *For any order* n *where*

$2n+1$ is prime, construct $A_n = (a_{ij})_{1 \le i,j \le n}$ such that, for all $1 \le i, j \le n$:

$$a_{ij} = \begin{cases} ij - \lfloor \frac{k_{ij}}{2} \rfloor (2n+1) & \text{if } k_{ij} \text{ is odd,} \\ \lfloor \frac{k_{ij}}{2} \rfloor (2n+1) - ij & \text{otherwise} \end{cases}$$

where the group k_{ij} is defined as $k_{ij} = \lfloor \frac{2ij}{2n+1} \rfloor + 1$.

We proved that our construction does produce SBLSs of order n, when $2n+1$ is prime. To do so, we first proved that the construction generates squares with the *Latin square* property (Theorem 1). The proof of the balancedness property (Theorem 2) is more involved. Interestingly, we used our streamlining framework to help prove the correctness of the construction. Due to space limitations we only state the theorems, omitting the proofs. They can be found in (Le Bras, Perrault, and Gomes 2012).

Theorem 1 (Latin square property)**.** *For any order n where $2n+1$ is prime, the square A_n, generated by the* `SBLS-Sequence` *construction, is a Latin square.*

Theorem 2 (Balancedness property)**.** *For any order n where $2n+1$ is prime, the square A_n, generated by the* `SBLS-Sequence` *construction, is a balanced square.*

SBLS-Cyclic — As mentioned above, without streamlining the search, we can only generate SBLSs of small orders. The streamliner *Reduced form* allows us to dramatically reduce the search space by removing symmetries, without eliminating non-isomorphic (under permutations) solutions. For this reason, it is typically used in CSP formulations of the SBLS problem. However, a downside of this streamliner is that it imposes a rather strict order on the first column and first row, potentially obfuscating the visualization of interesting structures. One question that arose was: Are there cyclic constructions for SBLSs? In order to investigate the existence of cyclic patterns, the *Reduced form* streamliner had to be disabled. In fact, a cyclic square such as the one in Figure 5 is entirely defined by its first row. Moreover, we were able to characterize the necessary and sufficient conditions that this first row must satisfy for the cyclic square to be an SBLS. We call such rows *highly balanced rows*. The problem of finding a construction for cyclic SBLSs therefore translates into the problem of finding a construction for highly balanced rows, which we solved using our framework. Without any streamliner, the search allows us to generate highly balanced rows of order up to 14. Among these rows, we isolated the rows starting with the first powers of 2, as this pattern was noticeably generalizable from low orders up to order 14. This observation led to yet another streamliner (according to which each integer i is followed by $2i$ whenever $2i \le N$), and ultimately, to the efficient `SBLS-Cyclic` construction presented in 3. Additional details can be found in (Le Bras, Perrault, and Gomes 2012).

Schur Numbers

Ramsey theory studies combinatorial structures that must reach a certain degree of order as the size of the structure increases. In this field, the *weak Schur number* problem considers intervals of integers $[1, n]$ for n in $\mathbb{N}$. The k^{th} weak

1	2	6	3	9	7	13	4	11	10	12	8	5	14
2	3	7	4	10	8	14	5	12	11	13	9	6	1
3	4	8	5	11	9	1	6	13	12	14	10	7	2
4	5	9	6	12	10	2	7	14	13	1	11	8	3
5	6	10	7	13	11	3	8	1	14	2	12	9	4
6	7	11	8	14	12	4	9	2	1	3	13	10	5
7	8	12	9	1	13	5	10	3	2	4	14	11	6
8	9	13	10	2	14	6	11	4	3	5	1	12	7
9	10	14	11	3	1	7	12	5	4	6	2	13	8
10	11	1	12	4	2	8	13	6	5	7	3	14	9
11	12	2	13	5	3	9	14	7	6	8	4	1	10
12	13	3	14	6	4	10	1	8	7	9	5	2	11
13	14	4	1	7	5	11	2	9	8	10	6	3	12
14	1	5	2	8	6	12	3	10	9	11	7	4	13

1	2	4	8	13	3	6	12	5	10	9	11	7	14
14	1	2	4	8	13	3	6	12	5	10	9	11	7
7	14	1	2	4	8	13	3	6	12	5	10	9	11
11	7	14	1	2	4	8	13	3	6	12	5	10	9
9	11	7	14	1	2	4	8	13	3	6	12	5	10
10	9	11	7	14	1	2	4	8	13	3	6	12	5
5	10	9	11	7	14	1	2	4	8	13	3	6	12
12	5	10	9	11	7	14	1	2	4	8	13	3	6
6	12	5	10	9	11	7	14	1	2	4	8	13	3
3	6	12	5	10	9	11	7	14	1	2	4	8	13
13	3	6	12	5	10	9	11	7	14	1	2	4	8
8	13	3	6	12	5	10	9	11	7	14	1	2	4
4	8	13	3	6	12	5	10	9	11	7	14	1	2
2	4	8	13	3	6	12	5	10	9	11	7	14	1

Figure 5: Cyclic SBLS (left) and its conjugate (right).

$c_{1,1} = 1$; // Generate 1st row of the conjugate
for *column* $j = 2, \ldots, N$ **do** // Observed pattern $1, 2, 4, \ldots$
  if $2c_{1,j-1} \le N$ **then**
    $c_{1,j} = 2c_{1,j-1}$;
  else
    $c_{1,j} = 2N + 1 - 2c_{1,j-1}$;
for *row* $i = 2, \ldots, N$ **do** // Subsequent rows
  $c_{i,1} = c_{i-1,N}$; // Shifted version of previous
  for *column* $j = 2, \ldots, N$ **do**
    $c_{i,j} = c_{i-1,j-1}$;
for *row* $i = 1, \ldots, N$ **do** // Generate SBLS from conjugate
  for *column* $j = 1, \ldots, N$ **do**
    $a_{i,c_{i,j}} = j$;

Algorithm 3: `SBLS-Cyclic` procedure.

Schur number *WS*(k) is the largest integer n for which there is a partition of $[1, n]$ into k sets such that no two distinct integers and their sum belong to the same set. Formally, we define this notion as follows.

Definition 2. *A set S is weakly (or strictly) sum free if for any $x, y \in S$ and $x \ne y$, it holds $x + y \notin S$.*

Definition 3. *For any $k \ge 1$, WS(k) is the largest integer n for which there exists a partition of $[1, n]$ into k weakly sum-free sets.*

The weak (or strict) Schur numbers have been studied for over seventy years. Rado (1941) showed that for any given k, *WS*(k) is finite. Through exhaustive search, Blanchard, Harary, and Reis (2006) established the values of the first four weak Schur numbers, which are 2, 8, 23 and 66. They also provided a lower bound on the fifth weak Schur number, namely *WS*$(5) \ge 189$. Recently, Eliahou et al. (2012) formulated this problem as a Boolean satisfiability problem and improved the bound of *WS*(5) to 196.

Interestingly, back in the 1950s, Walker (1952) not only provided the correct values for *WS*(3) and *WS*(4) but also claimed[2], without providing any detail or proof, that *WS*$(5) = 196$. Given that these results were clearly obtained without computer assistance, it is tempting to speculate that

[2]Walker claimed that $N = 197$ is the smallest number for which no partition of $[1, N]$ into 6 sets is weakly sum free, hence claiming *WS*$(6) = 196$.

Table 2: Partition of $[1, 581]$ into 6 weakly sum-free sets, proving $WS(6) \geq 581$. ('5-7' means '5 6 7' are in the set). Each of the six sets is such that the sum of any two of its members is not in the set.

1 2 4 8 11 22 25 50 63 68 139 149 154 177 182 192 198 393 398 408 413 436 450 455 521 526 540 563 568 578
3 5-7 19 21 23 51-53 64-66 136-138 150-152 179-181 193-195 395-397 409-411 438-440 451-453 523-525 536-538 565-567 579-581
9 10 12-18 20 54-62 140-148 183-191 399-407 441-449 527-535 569-577
24 26-49 153 155-176 178 412 414-435 437 539 541-562 564
67 69-135 454 456-520 522
196 197 199-392 394

Walker may have discovered a (partial) construction rule for the weak Schur numbers.

Recently, using combinatorial search techniques, Eliahou et al. (2012), reporting on results from 2011, provided a partition of $[1, 572]$ into 6 weakly sum-free sets, proving $WS(6) \geq 572$. Fonlupt et al. (2011) further improved this lower bound to 574 using a multilevel *tabu* search. However, Eliahou et al. (2012) added a note to their paper before it went to press in January 2012, claiming $WS(6) \geq$ **575**. This represents, until now, the best lower bound for the sixth weak Schur number.

Using our proposed framework, we constructed partitions for the first five weak Schur numbers that confirmed their current best known values. For the sixth weak Schur number, however, our framework generated a partition of $[1, 581]$ into 6 weakly sum-free sets (see Table 2), thus improving on the current best lower bound, by asserting that $WS(6) \geq$ **581**.

In the following we describe how we applied our proposed framework to the weak Schur number problem.

We first add a streamliner to order the sets of the partition, as a symmetry-breaking constraint would do. Given a k-wise partition, we order the k sets by increasing value of their minimum integer and we denote $m(j)$ the minimum integer of the j^{th} set. A striking feature of the solutions to the $WS(k)$ problem for $k \in \{1, , 4\}$ is that $m(j) = WS(j-1)+1$. For example, the first $WS(3) = 23$ integers of any strictly sum-free partition of $[1, WS(4) = 66]$ belong to the first three sets. In fact, as pointed out in (Fonlupt et al. 2011), the solutions to the $WS(4)$ problem are extensions of solutions to the $WS(3)$ problem. When studying the (non-strict) Schur numbers, a similar yet weaker feature might be observed for symmetric partitions (Fredricksen and Sweet 2000): the maximum of the $m(j)$ of a k-wise partition appears to be less or equal to $S(k-1)+1$, for $k \in \{1, , 6\}$. Inspired by the (non strict) Schur numbers, instead of imposing $m(j) = WS(j-1)+1$, we slightly relax this constraint to the following set of streamliners $\{m(j) \geq WS(j-1)-d : d \in [0, k]\}$. Interestingly, symmetry appears to be a recurring property of the solutions for the (non strict) Schur numbers, yet it leads to poor partitions for the strict version.

Nevertheless, these streamliners, coupled with partial pre-assignments of some integers, did not allow us to go beyond 575, the current best lower bound. The two key patterns that allowed us to improve the lower bound of $WS(6)$ up to 581 refer to sequences of consecutive numbers in the sets. Indeed, a surprising regularity of many solutions for the 6-wise partition problem of $[1, N]$ for $572 \leq N \leq 575$ is that the second set is mainly made of sequences of length 3 (i.e. three consecutive numbers, which is the largest possible as $m(2) = 3$), whereas the third is mainly made of sequences of length 9, etc. This observation led to partition of $[1, 578]$. Moreover, the pattern on how these sequences interleave between the sets appears to be recurring as well. As shown in Table 2, the last 3 numbers appear in the second set, the fourth to last integer belongs to the first set, followed by a sequence of length 9 in the third set, and so on. Streamlining on this feature led us to generate a strictly sum-free partition of $[1, 581]$ into 6 sets.

Although this result is not an example of a fully constructive procedure yet, any progress on Schur numbers is quite significant given their long history, and our framework enabled the discovery of this new lower-bound.

Conclusions

We have introduced a general framework for discovering efficient, constructive procedures for generating classes of complex combinatorial objects. The framework integrates, in an iterative approach, streamlined constraint search with a human computation component to identify possible patterns in solutions.

We demonstrated the effectiveness of our proposal with the discovery of two efficient constructive procedures for generating arbitrarily large spatially balanced Latin squares. No such procedures were known before. We also used the framework to provide a new lower-bound for weak Schur numbers.

Our framework opens up a new avenue for discovering constructive procedures for a range of challenge problems in combinatorics, finite algebra, and related areas. More generally, the integration of combinatorial reasoning and human computation techniques is an exciting research direction. We are also combining this approach with machine learning techniques for mathematical and scientific discovery.

Acknowledgments

The authors would like to thank the reviewers for their constructive comments. This work was supported by the National Science Foundation (NSF Expeditions in Computing award for Computational Sustainability, grant 0832782; NSF IIS award, grant 0514429) and the Intelligent Information Systems Institute, Cornell University (Air Force Office of Scientific Research, AFOSR, grant FA9550-04-1-0151). The first author was partially funded by an NSERC fellowship.

References

Appel, K., and Haken, W. 1977. Every planar map is four colorable. *Illinois Journal of Mathematics* 21(3):429–567.

Blanchard, P.; Harary, F.; and Reis, R. 2006. Partitions into sum-free sets. *Integers: Electronic Journal of Combinatorial Number Theory* 6(A7).

Charnley, J.; Colton, S.; and Miguel, I. 2006. Automatic generation of implied constraints. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva del Garda, Italy*, 73–77.

Colton, S., and Miguel, I. 2001. Constraint generation via automated theory formation. In *Principles and Practice of Constraint Programming, 7th International Conference, CP 2001, Paphos, Cyprus, November 26 – December 1*, 575–579.

Colton, S.; Bundy, A.; and Walsh, T. 1999. Automatic concept formation in pure mathematics. In *IJCAI*, 786–793.

Dahn, B. I. 1998. Robbins algebras are boolean: A revision of mccune's computer-generated solution of robbins problem. *Journal of Algebra* 208(2):526 – 532.

Eliahou, S.; Marín, J.; Revuelta, M.; and Sanz, M. 2012. Weak Schur numbers and the search for G.W. Walkers lost partitions. *Computers & Mathematics with Applications* 63(1):175–182.

Es, H. V., and Es, C. V. 1993. The spatial nature of randomization and its effects on field experiments. *Agron. J.* 85:420–428.

Es, H. V.; Gomes, C. P.; Sellmann, M.; and Es, C. V. 2007. Spatially-balanced complete block designs for field experiments. *Geoderma Journal* 140:346–352.

Fenner, S.; Gasarch, W.; Glover, C.; and Purewal, S. 2010. Rectangle free coloring of grids. Technical Report arXiv:1005.3750.

Flener, P.; Pearson, J.; Sellmann, M.; Van Hentenryck, P.; and gren, M. 2009. Dynamic structural symmetry breaking for constraint satisfaction problems. *Constraints* 14:506–538.

Fonlupt, C.; Robilliard, D.; Marion-Poty, V.; and Boumaza, A. 2011. A multilevel approach with backtracking to tabu search for exploring weak schur numbers. In *Proceeding of the Artificial Evolution Conference (EA'11)*. Springer.

Fredricksen, H., and Sweet, M. M. 2000. Symmetric sum-free partitions and lower bounds for schur numbers. *Electronic Journal of Combinatorics* 7.

Fujita, M.; Slaney, J.; and Bennett, F. 1993. Automatic generation of some results in finite algebra. In *Proceedings of the 13th international joint conference on Artifical intelligence - Volume 1*, 52–57. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Gent, I. P., and Smith, B. M. 2000. Symmetry breaking in constraint programming. In *Proceedings of ECAI-2000*, 599–603. IOS Press.

Gomes, C. P., and Sellmann, M. 2004. Streamlined constraint reasoning. In Wallace, M., ed., *CP*, volume 3258 of *Lecture Notes in Computer Science*, 274–289. Springer.

Gomes, C.; Sellmann, M.; Es, C. V.; and Es, H. V. 2004. The challenge of generating spatially balanced scientific experiment designs. In *In CP-AI-OR'04*, 387–394.

Herwig, P. R.; Heule, M. J. H.; Lambalgen, P. M. V.; and Maaren, H. V. 2007. A new method to construct lower bounds for van der waerden numbers. *Journal of Combinatorics* 14(1):2005.

Kouril, M., and Franco, J. 2005. Resolution tunnels for improved sat solver performance. In *In Proc. of 8th International Conference on Theory and Applications of Satisfiability Testing*, 143–157.

Law, E., and von Ahn, L. 2011. *Human Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Le Bras, R.; Perrault, A.; and Gomes, C. P. 2012. Polynomial Time Construction for Spatially Balanced Latin Squares. Technical report, Cornell University Library, http://hdl.handle.net/1813/28697.

McCune, W. 1997. Solution of the robbins problem. *Journal of Automated Reasoning* 19:263–276.

Neveu, B.; Trombettoni, G.; and Glover, F. 2004. Id walk: A candidate list strategy with a simple diversification device. In Wallace, M., ed., *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, 423–437. Springer.

Rado, R. 1941. Some Solved and Unsolved Problems in the Theory of Numbers. *The Mathematical Gazette* 25(264):72–77.

Rossi, F.; Beek, P. v.; and Walsh, T. 2006. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. New York, NY, USA: Elsevier Science Inc.

Slaney, J.; Fujita, M.; and Stickel, M. 1993. Automated Reasoning and Exhaustive Search: Quasigroup Existence Problems. *Computers and Mathematics with Applications*.

Smith, C.; Gomes, C.; and Fernandez, C. 2005. Streamlining local search for spatially balanced latin squares. In *IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence*, 1539–1541. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Walker, G. W. 1952. A problem in partitioning. *Amer. Math. Monthly* 59(253).

Zhang, H., and Hsiang, J. 1994. Solving open quasigroup problems by propositional reasoning. In *In Proceedings of the International Computer Symp*.