Latin Squares and Redundant Inequalities

Bart Demoen
Department of Computer Science, KU Leuven, Heverlee 3001, Belgium
Bart.Demoen@cs.kuleuven.be

Maria Garcia de la Banda
Faculty of Information Technology, Monash University, Caulfield 3145, Australia
Maria.GarciadelaBanda@monash.edu

Abstract

A complete classification of redundant sets of inequalities in the specification of the Latin Square problem of size N is proven. Related issues on variations of the same problem are discussed.

1 Introduction

Solving the Latin Square problem of size N requires filling out an $N \times N$ square with numbers from 1 to N in such a way that each row and column contains every number exactly once. A common formulation of Latin Square as a constraint satisfaction problem (CSP) [6] uses (a) $N \times N$ variables $x_{ij}, i, j \in [1..N]$, representing the value assigned to the cell in row i and column j of the square, (b) $N * N$ *domain* constraints indicating that the domain of each variable x_{ij} is $[1..N]$, and (c) $2 * N$ *all_different* constraints [7] of N variables each, one for the variables in each column and in each row. We refer to this CSP, that is, to the set of $N * N$ domain constraints and $2 * N$ *all_different* constraints, as *LatinSquare(N)*.

In this paper we prove that every *all_different* constraint is implied by the remaining 2*N-1 *all_different* constraints or, put differently, that any single *all_different* constraint in *LatinSquare(N)* is redundant (See Section 2). As a result, the CSP obtained by eliminating any of the $2*N$ *all_different* constraints is *equivalent* to *LatinSquare(N)*, since they have the same set of solutions. Furthermore, we also prove that no set of two (or more) *all_different* constraints is redundant, that is, that the CSP obtained by removing two or more *all_different* constraints is not equivalent to *LatinSquare(N)*, since it has more solutions.

An *all_different* constraint for N variables can also be formulated as the conjunction of the $N * (N - 1)/2$ pairwise binary inequality constraints on its input variables. For example, *all_different*$(\{x_1, x_2, x_3\})$ is logically equivalent

to the conjunction of the constraints $x_1 \neq x_2$, $x_1 \neq x_3$, and $x_2 \neq x_3$. Given the results mentioned before for the *all_different* constraints, it is clear that when Latin Square is specified using inequality constraints, there is a redundant set of $N*(N-1)/2$ inequalities (those associated to any single *all_different* constraint). Two natural questions arise for this specification: (1) whether $N*(N-1)/2$ is maximal, and (2) whether we can classify all redundant sets of inequality constraints. We answer these two questions in Section 3.

We then explore two variants of the Latin Square problem where the domain constraints are modified to reduce/increase the set of values that any cell can take. The results are reported in Sections 4 and 5.

Latin Squares can also be seen as a puzzle, like Sudoku: a certain number of entries is given (the clues) and solving the puzzle consists in filling it out completely while respecting the constraints [2]. It is also referred to as the *quasi-group completion problem*. The minimal number of clues to make a solution unique is interesting in itself, and it has been studied for instance in [4] for Latin Square puzzles. The recent paper [5] on Sudoku shows clearly the relation of unavoidable sets to the minimal number of needed clues, and also to the notion of *stability* of a solution. A solution is said to be *stable* if no single cell in it can be changed while remaining a solution, that is, if the distance (i.e., the amount of variables in the CSP or entries in the puzzle that differ) between any two solutions is at least 2. For both Latin Squares and Sudoku, the minimal distance between two solutions is four, making every solution stable. Sections 4 and 5 also discuss the stability of the variants of the Latin Square problem.

While our proofs are formal, some of the intuition behind the proofs resulted from running a constraint logic program for small values of N. We used B-Prolog [8] for this, although any Prolog with constraints would be adequate.

2 Redundant *all_different* constraints

It is easy to show that one *all_different* constraint can always be removed from *LatinSquare(N)* without affecting the set of solutions, since each *all_different* constraint is implied by all other *all_different* constraints. This can be proved as follows. Suppose, without loss of generality due to symmetries, that we remove from *LatinSquare(N)* the *all_different* constraint for the top row. Consider any number $I \in [1..N]$. Since every column still has an *all_different* constraint, the array must contain N occurrences of the number I, one per column. Since every row, except the top one, also has an *all_different* constraint, the lower rows must contain $N-1$ occurrences of I. Thus, the remaining occurrence of I must be in the top row. Since this reasoning applies to each value in $[1..N]$, the top row must contain every $I \in [1..N]$. This means all variables in the first row are different.

Let us now prove that one cannot remove more than one *all_different* constraint without increasing the set of solutions.

Theorem 2.1. *Any CSP obtained by eliminating from* LatinSquare(N) *two*

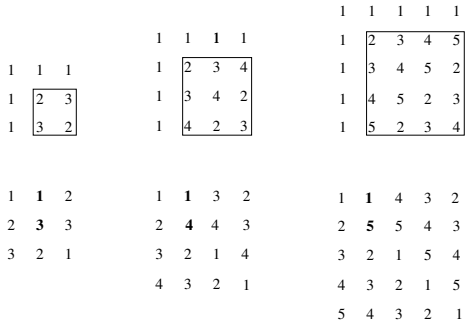


Figure 1: Extra solutions for $N = 3, N = 4$ and $N = 5$

or more all_different *constraints, has at least one more solution than* Latin-Square(N).

Proof Let us assume that the two *all_different* constraints removed cross each other (i.e., correspond to a row and a column). We can assume, without loss of generality due to symmetries, that the two constraints apply to the top row and the leftmost column. The top of Figure 1 shows one solution for such CSP with $N = 3, N = 4$ and $N = 5$ that is not a solution of the associated *LatinSquare(N)*. It is easy to see that the solutions can be straightforwardly generalized to a higher N by simply solving *LatinSquare(N-1)* with numbers $[2..N]$ (in the figure enclosed by a square) and then adding an extra column and row of 1s.

Let us now assume that the two *all_different* constraints removed are parallel to each other (i.e., correspond to two rows or two columns). Due to symmetries, we can assume without loss of generality that the two constraints apply to the two top rows. The bottom of Figure 1 shows one solution for such CSP with $N = 3, N = 4$ and $N = 5$ that is not a solution of the associated *LatinSquare(N)*. Again, it is easy to see that the solutions can be generalized to a higher N: construct first a solution by starting with the first column descending from 1 to N, and let subsequent columns be shifted down one position with respect to the previous column. Then exchange the values of $x_{1,2}$ and $x_{2,2}$: these are indicated in the figure in bold.

■

3 Redundant Inequalities

Reasoning about the redundancy of the binary inequality constraints is more complex due to the high number of such constraints. We thus proceed by analysing different values of N. The case $N = 2$ is easy to analyse by hand: any single inequality is redundant, while no set of two (or more) inequalities is.

We studied the cases $N = 3$ and $N = 4$ by letting a Prolog program generate

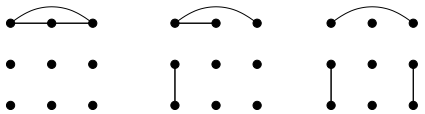


Figure 2: Maximal redundant sets of inequalities for $N = 3$

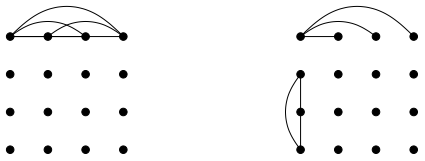


Figure 3: Maximal redundant sets of inequalities for $N = 4$

exhaustively all maximal redundant sets of inequalities up to symmetry[1].

Figure 2 shows all three solutions for $N = 3$ as a graph: each graph has a set of 9 vertices arranged as a square, with each vertex representing a variable in the 3×3 square of *LatinSquare(3)*, and each arc between two vertices representing an inequality (that involving the two variables connected by the arc). Thus, we can think of an arc as meaning *these two variables must be different*. We name such a graph *an inequality graph*: it represents a set of inequalities.

The statement that a set S of inequalities is redundant for a CSP P is saying that if S were missing from the specification of P, the solution space remains the same.

Figure 3 shows all maximal redundant sets for $N = 4$, computed using the same Prolog program.

Note that the two configurations for $N = 4$ follow the leftmost two configurations for $N = 3$. In particular, the leftmost configurations for $N = 3$ and $N = 4$ correspond to a pattern where all inequalities in the top row are missing. We will refer to this kind of configuration as *pattern 1* and denote by $R_1(N)$ the set of inequalities represented by pattern 1 for a given N. The middle configuration for $N = 3$ and the rightmost configuration for $N = 4$ correspond to a pattern where all inequalities between $x_{1,1}$ and the rest of variables in row one are missing and so are those involving any variables in the leftmost column except for $x_{1,1}$. We will refer to this kind of configuration as *pattern 2* and denote by $R_2(N)$ the set of inequalities represented by pattern 2 for a given N. In the rest of this section we will prove that these two patterns are redundant for any N and, in fact, are the only maximally redundant sets for $N > 3$.

[1] We intend not to repeat the qualification "up to symmetry"

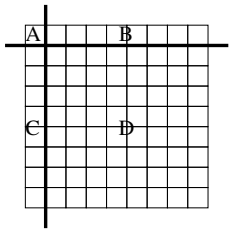
3.1 Pattern redundancy

Let us first prove that the CSPs represented by patterns 1 and 2 are redundant for all N.

Theorem 3.1. *Any CSP obtained by eliminating from* LatinSquare(N) *the inequalities in either* $R_1(N)$ *or in* $R_2(N)$ *is equivalent to* LatinSquare(N).

Proof For $R_1(N)$ the proof is straightforward, since the missing inequalities are equivalent to an *all_different* constraint in row one and, as shown before, one *all_different* constraint can always be removed from *LatinSquare(N)* without affecting the set of solutions.

For the proof about $R_2(N)$ we define 4 regions in the square as shown in the figure at the right: (1) A the upper left corner, (2) B the upper row without A, (3) C the left column without A, (4) D the rest of the square. For any filled region, we denote by $num(O)$ the set of numbers that appear in region O.



Consider a CSP P obtained by eliminating from *LatinSquare(N)* the inequalities in $R_2(N)$. It is clear that $|num(B)| = N - 1$ for any solution of P, given the inequalities P has on the first row. Let us denote by I the (only) element in $[1..N]$ that does not appear in $num(B)$. It now suffices to prove that $num(A) = I$ and that $num(C) = num(B)$.

Consider the $N-1$ columns in region $B \cup D$: since no inequality is missing in these columns, every number in $1..N$ must appear $N-1$ times in them. Since $I \notin num(B)$, region D must contain $N-1$ times number I, and $N-2$ times every number in $num(B)$. Consider now the $N-1$ rows in region $C \cup D$: since no inequality is missing from these rows, every number in $1..N$ must appear $N-1$ times in them. Since I appears $N-1$ times in D, $I \notin num(C)$. Since also every number in $num(B)$ appears $N-2$ times in region D, $num(C) = num(B)$. Finally, since the number in region A is different from all numbers in region C, $num(A) = I$. ■

3.2 Characterisation of all Pairs of Inequalities

In order to prove that the above two patterns (and their subsets) are the only possible redundant ones, we start by characterising every set S of inequalities in terms of its relationship with $R(N) = \{R_1(N), R_2(N)\}$. In particular, we say that S is *covered* by $R(N)$ if S is a (possible non-strict) subset of $R_1(N)$ or $R_2(N)$. Otherwise we say S is not covered. Note that, thanks to Theorem 3.1, we know that every covered set S results in a CSP that is equivalent to *LatinSquare(N)*.

Let us then determine whether every single pair of inequalities in *LatinSquare(N)* is covered by $R(N)$ or not. Figure 4 shows 8 missing equality graphs for *LatinSquare(4)*. Lemma 1 proves that that no other pair exists (up to symmetry). The boxed numbers in the graphs will be used to refer to particular

sets of missing pairs, i.e., S_i represents the set containing the two inequalities missing according to graph i.

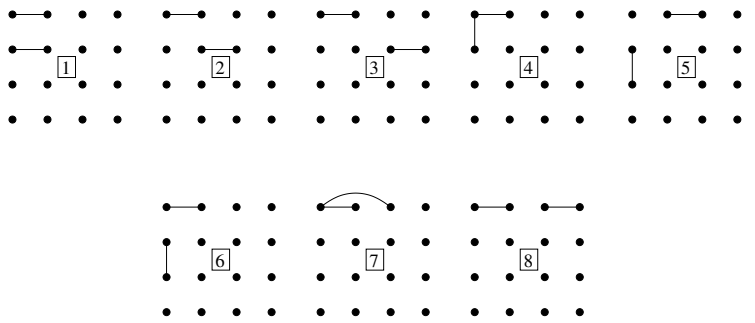


Figure 4: All possible pairs of inequalities for $N = 4$, up to symmetry

Lemma 1. Any pair of different inequalities from *LatinSquare(N)* corresponds, up to symmetry, to those in one of the sets $S_i, i \in 1..8$.

Proof There are three possibilities.

1. If the two inequalities affect variables from parallel lines (say rows): then their variables can share the same two columns (represented by S_1), only one column (S_2), or none (S_3). A similar reasoning can be made for parallel columns, due to symmetry.

2. If the two inequalities affect variables from crossing lines: then they can either share a variable (S_4), share none (S_5), or have three of their four variables in the same column/row (S_6).

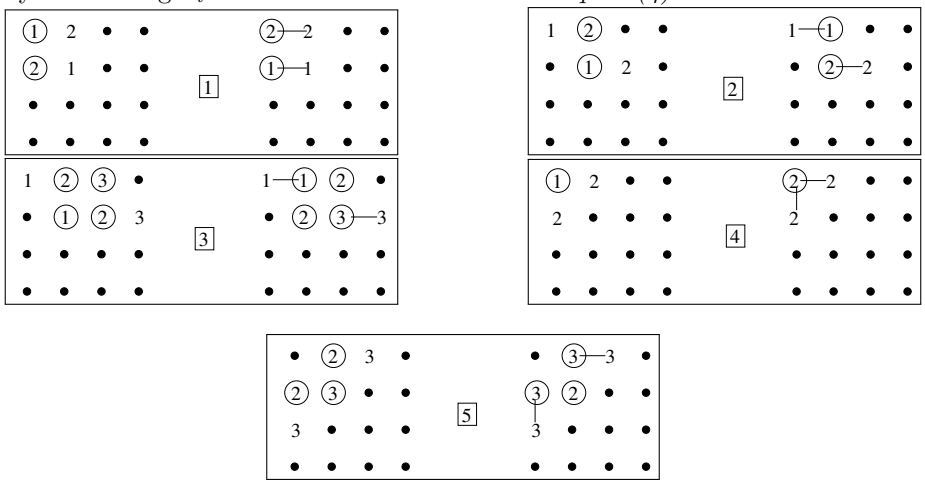
3. If the two inequalities affect variables in the same line (say a row): then the inequalities can either share a variable (S_7) or not (S_8). Again, a similar reasoning can be made for a column due to symmetry. ■

Note that no $S_i, i \in 1..5$ is covered, while all $S_i, i \in 6..8$ are covered (S_7 and S_8 by $R_1(N)$; S_6 by $R_2(N)$). Let us prove that every $S_i, i \in 1..5$ represents a *bad* pair, i.e., when missing from *LatinSquare(N)* it yields a CSP that is not equivalent to *LatinSquare(N)*. We find this fact (that a single pair of inequalities can modify the problem so fundamentally) quite surprising.

Theorem 3.2. *Any CSP obtained by eliminating from* LatinSquare(N) *the inequalities in any one of the sets $S_i, i \in 1..5$ has at least one more solution than* LatinSquare(N) *for all $N > 3$.*

Proof We first prove the case $N = 4$: we present graphical proofs for each S_i in the style of [3], where each proof consists of two pictures. The left picture represents a correct solution to *LatinSquare(4)* where only a few numbers are explicitly given (it is easy to see that such an initial solution exists for each

of them). The right picture is obtained from the left one by altering only the encircled numbers, and it represents a solution to the CSP obtained by eliminating S_i that is not a solution to *LatinSquare(4)* (as shown by the violated inequalities displayed in the picture as an arc between two numbers). Together, the two pictures of box i show that there exists a solution to the CSP obtained by eliminating S_i that is not a solution of *LatinSquare(4)*.



While the pictures show the proof for $N = 4$, it is straightforwards to see that they apply to any $N > 3$, since one can simply add more dots for extra columns and rows. One merely needs to convince oneself that there is always a solution of *LatinSquare(N)* with $N > 3$ starting with the numbers given in the left-hand side picture of each box (See Appendix 6). ■

Note that, as a result of the above theorem, any CSP that misses at least one bad pair, that is, that misses a set of inequalities S such that $S_i \subseteq S, i \in 1..5$, is also not equivalent to *LatinSquare(N)*. Together, the two previous theorems ensure that any equivalent CSP can only miss inequalities that are represented by S_6, S_7 and S_8.

3.3 Characterisation of all redundant Sets of Inequalities

The above characterisation of pairs can then be used to prove that the two patterns of $R(N)$ cover all possible redundant sets of inequalities.

Theorem 3.3. *Every set S of inequalities from* LatinSquare(N) *with $N > 3$ either contains a bad pair or is covered by $R(N)$.*

Proof If S has only one element, it is clearly covered by $R(N)$ and does not contain a bad pair. If S has two elements, Lemma 1 ensures they either form a bad pair, or one covered by $R(N)$. Let us then assume S has at least 3 elements. We simply need to prove that if S contains no bad pair, it is covered by $R(N)$.

If all inequalities in S affect variables in the same line (i.e., column or row), then S is clearly covered by $R_1(N)$. Let us assume then that the inequalities

affect variables in different lines. If the lines are parallel, then S would contain a bad pair (either S_1, S_2 or S_3). Since we are assuming it does not, then the affected variables must appear all in exactly two crossing lines. Let us assume, without loss of generality, that the lines are column 1 and row 1. By Lemma 1 we know that any crossing pair in S will be in S_4, S_5 or S_6. Since we are assuming S has no bad pairs, this means that (a) S_6 must be in S (up to symmetry), (b) S cannot contain any inequality involving variable $x_{1,1}$ and a variable in column 1 (or it would contain S_4), and (c) S cannot only contain inequalities affecting variables in the first row unless they involve $x_{1,1}$ (or it would contain S_5). Thus, S is only allowed to have inequalities from $R_2(N)$. ■

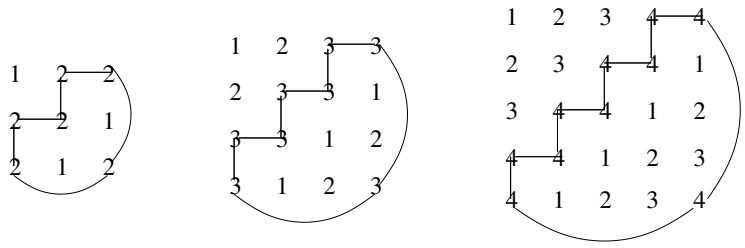
It follows from this theorem that the two sets $R_1(N)$ and $R_2(N)$ are maximally redundant.

4 Tightening the domain constraints

Up to now, we have kept the domain constraints for LatinSquare(N) intact, i.e., all variables can take values in the domain $1..N$. In this and the next section, we discuss a CSP in which the domain is either smaller, or larger.

Let us denote by *LatinSquare(N,I)* a CSP with the same variables and inequality (or *all_different*) constraints as *LatinSquare(N)* but where the domain of the variables is [1..I]. By tightening of the domain constraints we mean using an I such that $I < N$. Clearly, *LatinSquare(N,N-1)* has no solutions. Therefore, one cannot tighten the domain constraint and be satisfiable without giving up some inequality constraints. This section discusses the minimal number of inequalities one needs to eliminate in order for the problem to be satisfiable, and some of the characteristics intrinsic to the resulting set of solutions

The figure below shows 6, 8 or 10 inequalities that, when missing from *LatinSquare(3,2)*, *LatinSquare(4,3)* and *LatinSquare(5,4)*, respectively, allow them to have solutions.

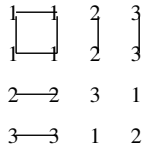


The key insight is that for a square of size N, one must break each *all_different* constraint since, otherwise, one needs a domain of size N. Therefore, at least $2 * N$ inequalities need to be removed. This shows that the above examples are also minimal: no inequality can be put back. The above generalizes as follows.

Theorem 4.1. *The minimal number of inequalities that need to be removed from* LatinSquare(N,(N-1)) *so that it is satisfiable is* $2 * N$.

Proof Immediate given that (a) any complete *all_different* constraint would not be satisfiable and, thus, $2 * N$ such constraints need to be broken, and (b) the figures show how to build a set S of inequalities that when eliminated from *LatinSquare(N,(N-1))*, allow the displayed variable/value mapping to become a solution. ■

Note that there are different ways in which the $2 * N$ inequalities can be removed, as shown by the figure below for $N = 4$.



The pictures suggest that variables which are allowed to be equal, must be equal in any solution. This is indeed the case, since making one pair different results in an *all_different* constraint on that line, which is not satisfiable. It follows that the solution set of two different sets of 2*N removed inequalities - one for each line in the square - have empty intersection.

The generalization of the above result to smaller domains is straightforward: if the domain is reduced by i elements, i arcs from each line (i.e., a total of $2 * i * N$) must be removed and the solution sets are disjoint. Formally:

Theorem 4.2. *The minimal number of inequalities that need to be removed from* LatinSquare(N,I)*, for $I < N$, so that it is satisfiable, is $2 * (N - I) * N$. Given two satisfiable CSPs obtained by removing different sets of $2 * (N - I) * N$ inequalities from* LatinSquare(N,I)*, their solution sets are disjoint.*

Proof Immediate given the previous results. ■

Note that the example solutions above are stable: one needs to change at least two variables in a solution to obtain a new solution. This generalizes to every solution s, since we know that every line in s contains all the numbers from 1 to I and, therefore, changing any single number violates the constraints in that line.

5 Relaxing the domain constraint

Let us now consider *LatinSquare(N,I)* where $I > N$ and, in particular, the case $I = N + 1$, which can be generalised to $I = N + i$ for $i > 0$. Clearly, *LatinSquare(N,N+1)* has all the solutions of *LatinSquare(N)* plus many others. As mentioned before, all solutions for *LatinSquare(N,I)*, with $I \leq N$ are stable. Interestingly, this is not the case for LatinSquare(3,4), as the example in Figure 5 shows.

In fact, we can generalise this as follows:

Theorem 5.1. *No solution to* LatinSquare(N,N+1) *is stable.*

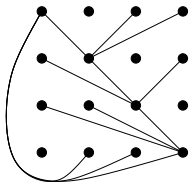
1	2	3		1	2	3
2	3	**4**		2	3	**1**
3	1	2		3	1	2

Figure 5: Two solutions to LatinSquare(3,4) differing in only one position

Proof Assume *Sol* is a solution to *LatinSquare(N,N+1)* for some N. By definition of the problem, *Sol* has N^2 variables each with a number from the domain $[1..(N+1)]$. Thus, not every number can occur N times in *Sol*, since $N*(N+1) > N^2$. For some number I in $[1..(N+1)]$, there must be both a row i and a column j in which I does not appear, since there are only N rows and N columns. A new solution can then be obtained by simply assigning the variable $x_{i,j}$ to value I. Therefore, *Sol* is not stable. ■

One can rephrase this theorem as *No amount of clues (except for all clues) makes a solution to LatinSquare(N,N+1) unique.*

Stability can be restored to the solutions of *LatinSquare(N,N+1)* by adding new inequality constraints. Adding all possible inequality constraints, i.e., adding inequality constraints between every two variables in the problem, restores stability trivially, because the problem becomes over-constrained. One can do better than that by reasoning as follows: in *LatinSquare(N,N+1)*, every variable is involved in a clique of inequalities of size N. If we add just enough inequalities so that every variable is involved in a clique of inequalities of size $N+1$, every solution is stable. Involving every variable in an $N+1$ clique can be done by adding the inequality graph of the figure below for $N=4$:



As the inequality graph shows, we add an inequality constraint between a variable $x_{i,i}$ on the diagonal and each of the variables on row(i-1) for each i in [1..4], where row(-1) is taken to be row(4). This adds $N*(N-1)$ inequalities and allows for solutions for N up to (at least) 14. Some solutions are shown further down for $N = 2, 3, 4, 5$. We do not know whether there are solutions for all $N > 14$, but we conjecture there are. We do not know either whether less than $N*(N-1)$ added inequalities can result in stability of every solution.

Let us denote by *LatinSquare*$^+$*(N,N+1)* the CSP obtained by adding to *LatinSquare(N,N+1)* the above inequalities and let $K(N)$ denote the smallest distance between any two solutions of a given problem. It is clear that $K(N) = 4$ for *LatinSquare(N)*[2]. For the case of *LatinSquare*$^+$*(N,N+1)* we have already shown above that its solutions are stable and, therefore, $K(N) > 1$. For $N = 2$ and $N = 3$, the problem has only one solution:

```
1 2        1 2 3
```

[2]Since changing one value requires changing at least one other value in the same row and column.

```
2 3        3 4 1
           4 3 2
```

respectively, up to value-symmetry, and it is easy to see that $K(2) = 2$ [3] and $K(3) = 4$ [4]. For $N > 3$, there is always more than one solution up to value-symmetry. For $N = 4$ we wrote a small Prolog program that exhaustively searches through the solutions to find $K(N)$. It determined that $K(4) = 3$, as exemplified by the following pair of solutions, where a dot indicates an unchanged cell:

```
1 2 3 4          . . . .
2 5 1 3          . . . .
3 1 4 5          . . . 2
4 3 5 2          . . 2 5
```

One can check that the second solution can not be obtained from the first by applying a value symmetry: some 5's are swapped with 2, but not all.

The same Prolog program determined that $K(5) = 2$, as shown by the following pair of solutions:

```
1 2 3 4 5          . . . . .
2 6 1 3 4          . . . . .
3 4 5 2 6          . . . . .
4 5 6 1 3          . . . . .
5 3 4 6 2          6 . . 5 .
```

While we expect $K(N)$ to be 2 for all $N > 4$, this is still only a conjucture.

6 Discussion and Conclusion

In this paper we have studied the redundancy relationships (also called implication or entailment) of the constraints used to specify the Latin Square program of size N, denoted as *LatinSquare(N)*. We have given all maximal redundancy sets found both when the problem is specified using *all_different* constraints and when it is specified using binary inequality constraints. Of particular interest have been the binary inequalities, because of the high number of such constraints needed in the specification and the complexity of their relationships. We have characterised all possible sets of binary equalities as either covered by one of our two maximal redundant sets or containing a bad pair, i.e., a pair of inequalities that – if missing – allows the problem to have solutions that are not in those of *LatinSquare(N)*. We find the fact that a single pair of inequalities can modify the problem so fundamentally, quite surprising.

We have also studied the effect on the problem of tightening and relaxing the domain constraints. In particular, we have determined the minimal number of inequalities that need to be removed from the tightened version for it to have

[3] One can swap the numbers 1 and 3: they each occur once.

[4] By swapping two values that occur minimally, e.g., 1 and 4.

a solution, and the fact that if two different sets of inequalities are removed, the solution set of the tightened problems will be disjoint. Further, we have noted how these solutions are stable. For the relaxed version, we have noted that none of its solutions is stable and have given tight upper bounds on the number of inequalities needed to be added to the problem to recover stability for the solutions. Finally, we have given the minimal distance for the cases of $N < 6$ and conjectured that for every $N > 5$ the distance is 2.

We hope that by studying particular CSP's (like Latin Square here, and Sudoku elsewhere [3]) we will get a better grasp on how (binary) inequality constraints entail other inequality constraints: at this point we have barely scratched the surface. In particular, we have no knowledge regarding *irregular* problems. Irregularity can result from two sources: (1) the inequality graph has only a few *automorphisms*, if any; (2) the domain of each variable is not the same. Both cases need more attention and success on either issue will lead to a deeper understanding of the propagation properties of the inequality constraint. As Latin Square and its variants are a case of overlapping *all_different* constraints, it is worth exploring it further from the point of view of [1], which gives a generalization of Hall's theorem for two overlapping *all_different* constraints. The difficulty comes from the fact that many problems – including Latin Squares and Sudoku – have many more than 2 overlapping *all_different* constraints. The study of minimality of constraints, uniqueness of solutions (for a set of clues) and stability of solutions is essential for the understanding of CSP's. This understanding also hinges crucially on the domain of the constrained variables: Sections 5 and 4 offer an approach to the domain issue in the context of Latin Squares, but it is clear that we are far from a comprehensive solution.

References

[1] C. Bessiere, G. Katsirelos, N. Narodytska, C.-G. Quimper, and T. Walsh. Propagating conjunctions of alldifferent constraints. *CoRR*, abs/1004.2626, 2010.

[2] C. J. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8(1):25 – 30, 1984. ISSN 0166-218X. doi: 10.1016/0166-218X(84)90075-1.

[3] B. Demoen and M. Garcia de la Banda. Less constraints is still sudoku. Conditionally accepted TPLP, 15 February, 2012.

[4] A. Keedwell. Critical sets in latin squares: An intriguing problem. *The Mathematical Gazette*, 85(503):239–244, 2001.

[5] G. McGuire, B. Tugemann, and G. Civario. There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem. *CoRR*, abs/1201.0749, 2012.

[6] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[7] W. J. van Hoeve. The alldifferent constraint: A survey, 2001. URL `http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:cs/0105015`.

[8] N.-F. Zhou. The language features and architecture of B-Prolog. *TPLP*, 12 (1-2):189–218, 2012. ISSN 1471-0684.

Appendix

The partial solutions given as the left-hand side of the proof boxes in Theorem 3.2 can be completed *LatinSquare(N)* as follows. For the (4) and (5) cases, one can simply fill the first row with numbers 1 to N in that order, and then fill the each row by shifting the previous one cyclically to the left. The partial solutions of boxes (2) and (3) are obtained in the same way, except that the shifting is to the right, rather than the left.

Case (1) is slightly more involved. For an even N we can divide the Latin Square into 2×2 squares of swapped numbers as follows: fill the first row with numbers 1 to N in that order, fill any $2 * i$ row by shifting the top row by $2 * i$ numbers to the left, and fill every other row by swapping every pair of numbers in the previous row. The picture on the right shows this solution for $N = 6$.

1	2	3	4	5	6
2	1	4	3	6	5
3	4	5	6	1	2
4	3	6	5	2	1
5	6	1	2	3	4
6	5	2	1	4	3

The case for N odd is shown bellow for $N = 5$, $N = 7$ and $N = 9$. Each picture shows the solution itself on the left, and a the part of it that is essential for the generalisation on the right. Each solution is achieved as follows: the first row and column are numbered from 1 to N in ascending order from $x_{1,1}$, and every ascending diagonal from an element $x_{i,1}$ in the first column is given the same value as $x_{i,1}$, except for $x_{2,2}$ which is always 1. Then, we number the last column and row from $N - 1$ to 4 starting from $x_{n,n}$, and every ascending diagonal from an element $x_{n,i}$ in the last row is given the same value as $x_{n,i}$. This leaves us with the 3 ascending diagonals that start from $x_{n,2}$, $x_{n,3}$ and $x_{n,4}$. The last two rows of such diagonals (marked `A` and `B`) in the figure, are always the same. The rest follows the repeating `X` and `Y` pattern shown in the figure. Clearly, the pattern generalises for any odd $N > 9$.

```
1 2 3 4 5          1 2 3 4 5
2 1 4 5 3          2 1     3
3 4 5 2 1          3     1 2
4 5 1 3 2          4   2 3 1  A
5 3 2 1 4          5 3 1 2    B

1 2 3 4 5 6 7      1 2 3 4 5 6 7
2 1 4 5 6 7 3      2 1         3
3 4 5 6 7 2 1      3           2 1
4 5 6 7 1 3 2      4       1 3 2  X
5 6 7 2 3 1 4      5     2 3 1    Y
6 7 1 3 2 4 5      6   1 3 2      A
7 3 2 1 4 5 6      7 3 2 1        B

1 2 3 4 5 6 7 8 9  1 2 3 4 5 6 7 8 9
2 1 4 5 6 7 8 9 3  2 1             3
3 4 5 6 7 8 9 2 1  3               1 2
4 5 6 7 8 9 1 3 2  4             3 2 1  X
5 6 7 8 9 2 3 1 4  5           1 2 3    Y
6 7 8 9 3 1 2 4 5  6         3 2 1      X
7 8 9 2 1 3 4 5 6  7       1 2 3        Y
8 9 1 3 2 4 5 6 7  8     2 3 1          A
9 3 2 1 4 5 6 7 8  9 3 1 2              B
```