Constraints (2013) 18:471–477
DOI 10.1007/s10601-013-9147-1

LETTER

Redundant disequalities in the Latin Square problem

Bart Demoen · Maria Garcia de la Banda

Published online: 6 August 2013
© Springer Science+Business Media New York 2013

Abstract This paper studies the redundancy properties of the constraints used when formulating the well known Latin Square problem. This problem is often formulated using either $(N-1) * N^2$ binary disequalities or $2 * N$ *all_different* global constraints. Both formulations contain redundant constraints. A complete classification of all redundant sets of constraints, be they binary or global, is performed for any N.

Keywords Constraint programming · Redundant constraints

1 Introduction

The search for a solution to a constraint problem can sometimes be sped up by either adding or removing *redundant* constraints (also called implied or entailed constraints; see for instance pages 376 and 431 in [8]), that is, constraints that do not change the set of solutions to the problem. Therefore, it is useful to be able to identify redundant constraints in a problem specification (or *model*), and also to infer new, redundant constraints that could be added to it. Redundant and implied constraints, and their effects on performance, have been studied for instance in [10] and [5] in the context of Linear Programming, and in [1] and [7] in Constraint Programming. In particular, [7] studied the addition of constraints to the usual definition of the cardinality matrix problem (which includes the Latin Square) to improve its performance. Our motivation is however different: our aim is to study the redundancy

This research was partly sponsored by he Australian Research Council grant DP110102258, and by the Research Foundation Flanders (FWO) through projects *WOG: Declarative Methods in Computer Science* and *G.0221.07*

B. Demoen (✉)
KU Leuven, Department of Computer Science, Leuven, Belgium
e-mail: bart.demoen@cs.kuleuven.be

M. Garcia de la Banda
Faculty of Information Technology, Monash University, VIC 3145, Australia

properties of the constraints already present in the problem, rather than infer new redundant constraints.

While disequalities are commonly used to model constraint problems, determining whether a disequality is redundant or not is not easy due to, among other things, their lack of transitivity and their dependence on the domains of the variables. For instance, from a chain of disequalities $x_1 \neq x_2 \neq \cdots \neq x_n$ between *boolean* variables, one may conclude that $x_1 \neq x_4$ (amongst others), but if the domains have a larger cardinality, this conclusion is no longer valid.

This paper studies the redundancy properties of the disequality constraints used for modelling the Latin Square [2] problem of size N, which requires filling out an N^2 square with numbers from 1 to N in such a way that each row and column contains every number exactly once. A common model of Latin Square as a constraint satisfaction problem 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^2 *domain* constraints indicating that the domain of each variable x_{ij} is $[1..N]$, and (c) $2N$ *all_different* global constraints [4, 6] of N variables each, one for the variables in each column and in each row. We refer to this model, that is, to the set of variables, domain constraints and *all_different* constraints, as *LatinSquare(N)*.

An alternative model substitutes each of the *all_different* global constraints by the conjunction of the $N * (N-1)/2$ binary disequality constraints on its input variables. For example, *all_different*$(\{x, y, z\})$ is substituted by the conjunction of constraints $x \neq y$, $x \neq z$, and $y \neq z$. We refer to this alternative model as *LatinSquareBin(N)*. It has the same number of variables and domain constraints as *LatinSquare(N)*, but $N^2 * (N-1)$ disequalities rather than $2N$ *all_different* global constraints.

We provide a complete classification of all redundant and non-redundant sets of constraints used in $LatinSquare(N)$ and $LatinSquareBin(N)$. To achieve this, we first classify any pair of disequalities as either redundant or non-redundant. This works well for Latin Square because the number of different pairs of disequalities is small when considered up to symmetry, that is, when considered up to the spatial symmetries of a square (any row, column and diagonal symmetries). Once this is done, we prove that any set of disequalities either contains at least one non-redundant pair and is, therefore, non-redundant, or otherwise is redundant. We believe this method will work also for other problems in which a limited number of small patterns of disequalities provably occurs in any non-redundant set of disequalities. Constraint satisfaction problems with many symmetries seem good candidates. 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 [11] for this, although any Prolog with constraints would be adequate.

2 Redundancy of sets of binary disequalities in *LatinSquareBin(N)*

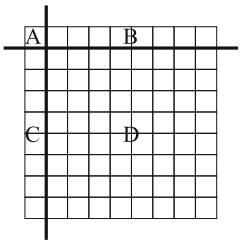
Our results for disequalities can be easily visualized using a constraint graph where nodes represent the variables in *LatinSquareBin(N)* and edges represent the disequalities. We define two particular subsets of these edges:

$$R_1(N) = \{(x_{1i}, x_{1j}) | 1 \leq i < j \leq N\}$$

$$R_2(N) = \{(x_{11}, x_{1j}) | 1 < j \leq N\} \cup \{(x_{i1}, x_{j1}) | 1 < i < j \leq N\}$$

Fig. 1 $R_1(4)$ and $R_2(4)$

Informally, $R_1(N)$ contains all disequalities between any two variables in row 1, while $R_2(N)$ contains all disequalities in row 1 involving the top-left variable (x_{11}), and also all the disequalities in column 1 not involving x_{11}. Figure 1 shows these sets for $N = 4$.



This section proves not only that both $R_1(N)$ and $R_2(N)$ are redundant sets of disequalities, but also that, up to symmetry, they are the only *maximal redundant* sets. That is, the only ones that cannot be extended without becoming non-redundant. Note that when we say "up to symmetry" we refer to the symmetries of the square, that is, those formed by exchanging any two rows, any two columns, reflecting the square upon any diagonal, or any combination of these. While Latin Square has other symmetries (e.g., those formed by exchanging any two values), these are irrelevant, since the symmetries we are considering are those of the associated constraint graph only, not all those of the Latin Square problem.

2.1 $R_1(N)$ and $R_2(N)$ are redundant sets of disequalities

Lemma 1 *For all $N > 0$, $R_1(N)$ is a redundant set of disequalities.*

Proof Consider the model obtained from *LatinSquareBin(N)* by removing the disequalities in $R_1(N)$. Let s be any solution of this modified model, and I any number in $[1..N]$. Since in every column of s all elements are known to be different, s must contain N occurrences of the number I, one per column. Since in every row, except the top one, all elements are also known to be different, 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 $I \in [1..N]$. As a result, all variables in the first row of any solution s must be different and, therefore, $R_1(N)$ is redundant. □

Lemma 2 *For all $N > 0$, $R_2(N)$ is a redundant set of disequalities.*

Proof Let us define four regions of any solution as shown in the figure at the right: A as the upper left cell, B the upper row without A, C the left column without A, and D the rest of the square. Consider the model obtained by removing from

LatinSquareBin(N) the disequalities in $R_2(N)$. It is easy to see that for any solution s and any number I in region B of s, the number of occurrences of I in B is 1, since there are disequalities between each pair of variables in B. Also, I occurs $N-1$ times in $B \cup D$, since there are disequalities between each pair of variables in any of its columns. Therefore, I occurs $N-2$ times in D. Further, I also occurs $N-1$ times in $C \cup D$, since there are disequalities between each pair of variables in any of its rows. Therefore, I occurs once in C.

As a result, the set of numbers occurring in C and occurring in B are the same, which means *all_different(C)* holds for any s. Further, since the number in A cannot appear in B, all disequalities in $R_2(N)$ hold for any s and, therefore, $R_2(N)$ is redundant. □

Let $R(N) = \{R_1(N), R_2(N)\}$ and S denote any set of disequalities. We say that S is *covered* by $R(N)$ if, up to symmetry, S is a (possibly non-strict) subset of $R_1(N)$ or $R_2(N)$. Otherwise, we say S is not covered. It is clear that any set S covered by $R(N)$ is redundant.

2.2 Non-redundant pairs of disequalities

Clearly, any single disequality is covered by $R(N)$. We thus turn our attention to pairs of disequalities. Figure 2 shows all pairs of disequalities in LatinSquareBin(4) up to symmetry. We denote them by P_i, $i = 1..8$ as numbered in the figure. For $N > 4$, the exact same eight pairs exist, and up to symmetry, no more. This can be proven by exhaustively considering all possibilities as follows: two different disequalities have a variable in common or not. In the former case, the three variables lie either in the same row (P_7) or not (P_4). In the latter case, the four variables lie either in the same row (P_8), in two different rows (P_1, P_2 and P_3), or in three different rows (P_5 and P_6).

For $N = 3$, P_3 and P_8 do not exist, and for $N = 2$, only P_1 and P_4 exist.

Clearly, pairs P_6, P_7 and P_8 are covered by $R(N)$ and, therefore, they are redundant. None of the remaining pairs is covered by $R(N)$ and, as we prove, they are all non-redundant.

Theorem 1 *Consider the model obtained by removing from* LatinSquareBin(N) *the disequalities in some* P_i, $i \in 1..5$. *For* $N > 3$, *this modified model has at least one more solution than* LatinSquareBin(N).

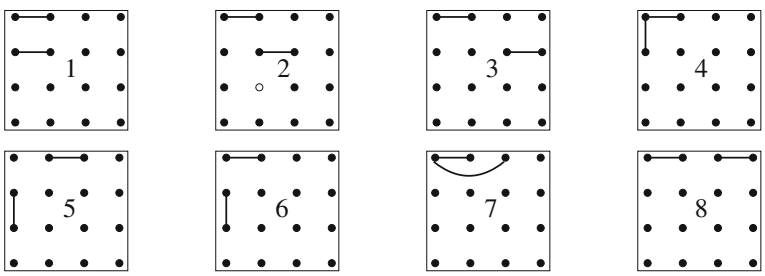
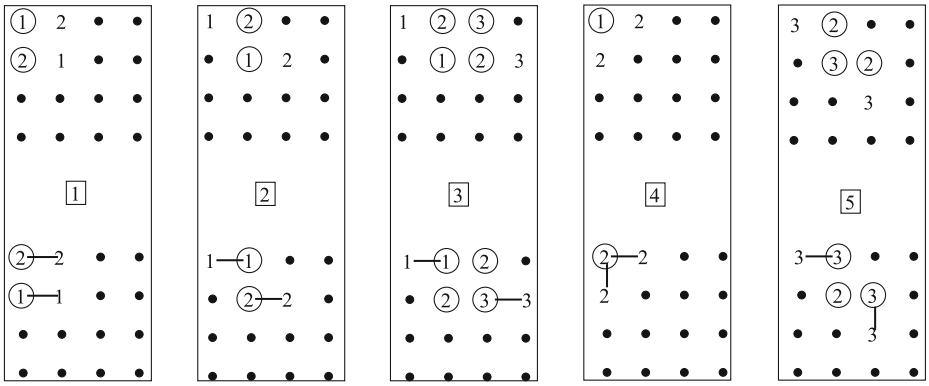


Fig. 2 All possible pairs of disequalities for $N = 4$, up to symmetry

Proof We first prove the case $N = 4$: we present graphical proofs for each P_i in the style of [3], where each proof consists of a box with two pictures. The top picture represents a solution to *LatinSquareBin(4)* where only a few numbers are explicitly given (it is easy to see that each top picture can be completed to several solutions). The bottom picture is obtained from the top one by altering only the encircled numbers, and it represents a solution to the model obtained by eliminating P_i, that is not a solution to *LatinSquareBin(4)* (as shown by the violated disequalities 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 model obtained by eliminating P_i, that is not a solution of *LatinSquareBin(4)*.



While the pictures above give a proof for $N = 4$, it is straightforward 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 prove that there is always a solution of *LatinSquareBin(N)* with $N > 3$ starting with the numbers explicitly given in the top picture of each box. This can be done easily for boxes 2 up to 5 by rotating the successive rows. For box 1, one can use Theorem 2 from [9]. □

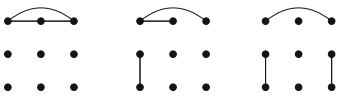
2.3 Up to symmetry, $R_1(N)$ and $R_2(N)$ are the only maximal redundant sets

Theorem 2 *Every set S of disequalities from* LatinSquareBin(N) *with $N > 3$ either contains a non-redundant pair or is covered by $R(N)$.*

Proof There are three cases for S. (1) All disequalities in S affect variables in the *same line* (i.e., column or row). In this case S is clearly covered by $R_1(N)$ and does not contain a non-redundant pair. (2) Some disequalities of S appear in different *parallel lines*. In this case, S must contain a non-redundant pair from either P_1, P_2 or P_3, and is thus not covered by $R(N)$. (3) All disequalities of S must now appear in exactly *two crossing* lines. Let us assume, without loss of generality, that the lines are column 1 and row 1. By Fig. 2 we know that any crossing pair in S must be in P_4, P_5 or P_6. If S contains a pair in P_4 or P_5, then S contains a non-redundant pair and is not covered by $R(N)$. Else, S is only allowed to have pairs of disequalities derived (via symmetry) from P_6 and, thus, they are all in $R_2(N)$. □

It follows from the above theorem that $R_1(N)$ and $R_2(N)$ are the only two maximal redundant sets of disequalities for $N > 3$.

Fig. 3 All maximal redundant sets of disequalities for $N = 3$



The special case $N = 3$: The reasoning used in Theorem 1 to prove that P_1 is a non-redundant pair is not valid for $N = 3$, since the initial configuration used in box 1 cannot be completed to a solution of *LatinSquareBin(3)*. However, for $N = 3$ it is easy to exhaustively enumerate all maximal sets of disequalities, which are shown in Fig. 3. Note that the rightmost set contains P_1 and therefore (by Theorem 1) cannot be generalized to $N > 3$.

3 Redundancy of *all_different* constraints in *LatinSquare(N)*

Theorem 2 directly implies the following result:

Corollary 1 *Exactly one all_different constraint in the formulation of* LatinSquare(N) *is redundant.*

Proof Every single *all_different* constraint is clearly redundant, as every such constraint is equal to $R_1(N)$ up to symmetry and, thus, covered by $R(N)$. Moreover, no two *all_different* constraints together can be covered by $R(N)$, since they always would contain a non-redundant pair: either in P_1, P_2 and P_3 if the constraints appear in parallel lines, or in P_4 and P_5 if they cross. Therefore, no combination of two *all_different* constraints is redundant. □

4 Conclusion

This paper studies the redundancy relationships of the binary disequality constraints and global *all_different* constraints used to model the Latin Square program of size N. In particular, we have identified all maximal redundant sets found both when the problem is specified using *all_different* constraints, and when it is specified using binary disequality constraints. For the former case, we have proved that every *all_different* constraint is redundant and no two such constraints are redundant. For the latter case, we have characterized all possible sets of binary equalities as either covered by one of our two maximal redundant sets, or containing a non-redundant pair. That is, a pair of disequalities that – if missing – allows the problem to have solutions other than those of *LatinSquare(N)*. We find the fact that a single pair of disequalities can modify the problem so fundamentally, quite surprising. A small experiment showed that taking out the redundant constraints from the problem description reduces performance by an order of magnitude even for problem sizes as small as $N = 6$, and that this reduction grows with N.

The method used to prove the above results is based on first classifying any pair of disequalities $(P_1, \cdots, P_8)$ as either redundant or non-redundant, and then proving that any set of disequalities S either contains at least one non-redundant pair and is, therefore, non-redundant, or otherwise is redundant. We believe this method will work also for other problems in which a limited number of small patterns of

disequalities provably occurs in any non-redundant set of disequalities. Note that both the results and the methodology are quite different from those in [3], where we proved for 3×3-Sudoku that up to six *all_different* constraints (out of the usual 27 with which the problem is defined) are redundant, and we gave a full classification of such sets of six. We also found that many subsets of 162 disequalities are redundant for 3×3-Sudoku and conjectured that no more is possible. No results for NxN-Sudoku for general N were obtained. In contrast, for the Latin Squares problem, a full classification is obtained not only for the redundant *all_different* constraints, but also for the redundant binary disequality constraints. Moreover, the results were obtained for all problem sizes N. Further, the methodology used in [3] was based on a series of constructive lemmas where given a set of disequalities, a new redundant disequality was inferred.

We hope that by studying particular constraint problems (like Latin Square here, and Sudoku elsewhere [3]), we will get a varied toolset of methods with which to determine the redundancy properties of disequality constraints in a wide set of problems.

Acknowledgements We thank Ian Wanless for pointing us to relevant literature on Latin Squares, and anonymous reviewers for suggestions that improved the presentation.

References

1. Bessière, C., Coletta, R., Petit, T. (2007). Learning implied global constraints. In Veloso, M.M. (Ed.), *International joint conference on artificial intelligence* (pp. 44–49).
2. Colbourn, C., Dinitz, J., Wanless, I. (2007). Latin squares. In Colbourn, C., Dinitz, J. (Eds.), *The CRC handbook of combinatorial designs* (pp. 135–152). Chapman & Hall/CRC, Boca Raton.
3. Demoen, B., & Garcia de la Banda, M. (2013). Redundant sudoku rules. *Journal of Theory and Practice of Logic Programming*. doi:10.1017/S1471068412000361
4. Gent, I.P., Miguel, I., Nightingale, P. (2008). Generalised arc consistency for the alldifferent constraint: An empirical survey. *Artificial Intelligence, 172*(18), 1973–2000. doi:10.1016/j.artint.2008.10.006.
5. Greenberg, H.J. (1996). Consistency, redundancy, and implied equalities in linear systems. *Annals of Mathematics and Artificial Intelligence 17*(1), 37–83.
6. van Hoeve, W.J. (2001). The alldifferent constraint: a survey. arXiv:0105015. Accessed 24 Aug 2013
7. Régin, J.C., & Gomes, C.P. (2004). The cardinality matrix constraint. In *Principles and practice of constraint programming–CP 2004* (pp. 572–587). Springer.
8. Rossi, F., van Beek, P., Walsh, T. (Eds.) (2006). *Handbook of constraint programming*. Elsevier.
9. Ryser, H. (1951). A combinatorial theorem with an application to Latin rectangles. *Proceedings of the American Mathematical Society, 2*(4), 550–552.
10. Telgen, J. (1978). Redundant constraints in linear programming problems. *Operations Research Verfahren, 28*, 420–433.
11. Zhou, N.F. (2012). The language features and architecture of B-Prolog. *Theory and Practice of Logic Programming, 12*(1–2), 189–218.