

Latin Squares with Graph Properties

Hans Zantema^{1,2}

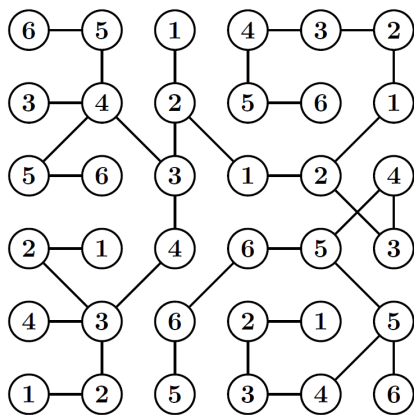
¹ TU Eindhoven, Department of Computer Science, P.O. Box 513,
5600 MB Eindhoven, The Netherlands h.zantema@tue.nl

² Radboud University Nijmegen, Institute for Computing and Information Sciences,
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands

Abstract. In a Latin square of size n every number from 1 to n occurs exactly once in every row and every column. Two neighbors (straight or diagonal) in a Latin square are connected by an edge if their values differ at most 1. We investigate Latin squares for which this underlying graph has typical graph properties, like being connected, being a tree or being Hamiltonian. Searching for such Latin squares shows up combinatorial explosion. Nevertheless, by exploiting current SAT/SMT solving, we explicitly find instances for several variants, or prove that they do not exist.

1 Introduction

Consider the following picture:



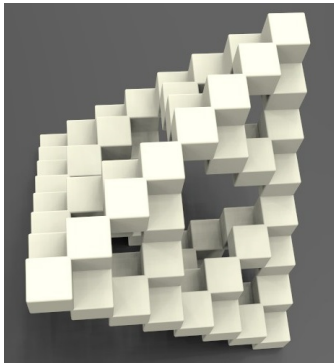
It has a number of remarkable properties:

- It is a *Latin square* of size 6, that is, every row and every column contains the numbers from 1 to 6, all exactly once.
- Two neighbors (in horizontal, vertical or diagonal direction) are connected by an edge if and only if the numbers differ by at most one.
- The resulting graph is a *tree*, that is, for every two nodes there is exactly one path from the one node to the other.

In this short paper we discuss how to generate such a Latin square with additional properties, or prove that it does not exist, by means of SAT solving. The total number of Latin squares of size 6 is 812851200, in which a direct search

Proceedings of the 21st RCRA workshop on *Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion* (RCRA 2014).
Vienna, Austria, July 17–18, 2014.

might be feasible, but we also consider Latin squares of higher size for which a direct search is blocked by combinatorial explosion. On the one hand our work is a case study in solving problems combining combinatorial and graph theoretical constraints, on the other hand it is the basis of generating and solving variants of the popular sudoku puzzle as we will see later in this paper. Where a direct search serves well for generating and solving standard sudoku puzzles, this does not hold for the variants presented here.



The question originally arose by Hans Kuiper who wanted to represent Latin squares and sudoku's in a three dimensional image by representing every number by a unit cube on the height coinciding with its number. Then connectedness is an obvious physical requirement for this three dimensional image, exactly coinciding with connectedness of the graph structure presented above. The picture on the left from [6] shows such a three dimensional image for an example of size 8. For size 9 with the extra requirement of sudoku this was a non-trivial job, succeeded by Hans Kuiper

and Walt van Ballegooijen in the paper [6] at the Bridges conference in 2013. Their paper was our starting point: can we exploit SAT solving for finding solutions in this area?

The key issue is how to describe graph properties like connectedness, tree structure or Hamiltonian as requirements in a SAT/SMT formula such that the solvers can deal with it. We want to stress that this is not about algorithms to decide whether a given graph has the particular graph property, but about specifying the property in such a way that by SAT/SMT solving particular graphs can be constructed satisfying both the specified graph property and some extra properties like Latin square requirements.

We present characterizations of the graph properties such that combined with Latin square requirements solutions are found efficiently by current tools. The example at the start of this introduction is an encoding of such a solution.

In Section 2 we present the theory and encoding. In Section 3 we discuss our implementation. In Section 4 we discuss how this machinery was the basis of our Sudoconnect project: a class of puzzles combining sudoku and the underlying graph structure. We conclude in Section 5.

2 Graph Theory

For an undirected graph (V, E) we will characterize some graph properties in a way suitable for encoding in SAT. Let $n = \#V$ be the number of nodes, write $V = \{1, 2, \dots, n\}$. Edges in E are pairs of nodes; as the graph is undirected we identify (i, j) and (j, i) , and assume $(i, i) \notin E$ for $i = 1, 2, \dots, n$. We inductively define $V_0 = \{1\}$ and $V_k := V_{k-1} \cup \{i \in V \mid \exists j \in V_{k-1} : (i, j) \in E\}$ for $k > 0$. So V_k consists of the nodes that are reachable from 1 in at most k steps. Since

there are n nodes in total V_k may strictly extend V_{k-1} at most $n - 1$ times, by which we obtain $V_k = V_{n-1}$ for $k > n - 1$.

A graph is called *connected* if for every two nodes i, j there is a path from i to j . As such a path may be obtained from concatenating a path from i to 1 and a path from 1 to j , and a path from i to 1 may be reversed to a path from 1 to i , a graph is connected if and only if for every node i there is a path from 1 to i , or stated equivalently, $i \in V_k$ for some k . As $V_k = V_{n-1}$ for $k > n - 1$, we conclude the following.

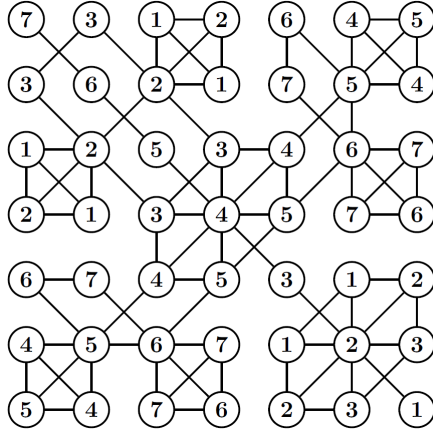
Theorem 1 *The graph (V, E) is connected if and only if $V_{n-1} = V$.*

For encoding this characterization in SAT let the graph be described by n^2 boolean variables a_{ij} for $i, j = 1, 2, \dots, n$, where a_{ij} describes whether $(i, j) \in E$. For describing undirectedness we always add the requirements $\bigwedge_{1 \leq i < j \leq n} a_{ij} \leftrightarrow a_{ji}$ and $\bigwedge_{i=1}^n \neg a_{ii}$.

For describing connectedness we introduce n^2 extra boolean variables b_{ik} for $i = 1, 2, \dots, n$ and $k = 0, 1, \dots, n - 1$, and define $C(m)$ to be the formula

$$b_{10} \wedge \bigwedge_{i=2}^n \neg b_{i0} \wedge \bigwedge_{i=1}^n b_{im} \wedge \bigwedge_{i=1}^n \bigwedge_{k=1}^m (b_{ik} \leftrightarrow (b_{i,k-1} \vee \bigvee_{j=1}^n (a_{ij} \wedge b_{j,k-1}))).$$

So b_{ik} is true if and only if $i \in V_k$. As the formula moreover contains $\bigwedge_{i=1}^n b_{im}$, for $m = n - 1$ this specifies $V_{n-1} = V$. Hence by Theorem 1 we conclude that the graph on n nodes $1, 2, \dots, n$ specified by a_{ij} for $i, j = 1, 2, \dots, n$ is connected if and only if boolean values for b_{ik} exist for which the formula $C(n - 1)$ yields true.



The requirement $V_{n-1} = V$ in Theorem 1 can be strengthened to $V_m = V$ for some $m < n - 1$. In fact then one describes that every node is reachable in at most m steps from node 1. Also note that the formula described above then only deals with b_{ik} for $k = 0, 1, \dots, m$, so decreases in size when decreasing m . The picture on the left shows a Latin square of size 7 obtained in this way, where $m = 3$ and the special node 1 is chosen to be the middle one. So every node is reachable from the middle node in at most three steps.

An undirected graph is called a *tree* if it is connected and acyclic. Here acyclic means that there is no cycle, that is, a non-empty path from a node to itself without duplicate nodes. A standard characterization of trees, see e.g., [5], Theorem 11.5, or [1], Theorem 7.2, states that a connected graph is a tree if and only if $\#E = \#V - 1$. This characterization is easily expressed in SMT or by

pseudo-boolean constraints. However, when doing so this performs much worse than the characterization we will propose now. A reason for this is that when in the SAT machinery searching for a solution many a_{ij} 's already got a value and may cause a cycle, in the counting approach this is not detected hence the search continues. Our approach results in a substantial speedup by backtracking due to early cycle detection.

Theorem 2 *The graph (V, E) is a tree if and only if the following three properties hold:*

1. $V_{n-1} = V$,
2. for every $k = 1, \dots, n-1$, $1 \leq i < j \leq n$ for which $i, j \in V_k \setminus V_{k-1}$ one has $(i, j) \notin E$, and
3. for every $k = 1, \dots, n-1$, $i = 1, \dots, n$ for which $i \in V_k \setminus V_{k-1}$ there is at most one $j \in V_{k-1}$ such that $(i, j) \in E$.

Proof. First assume that (V, E) is a tree. If (1) does not hold then by Theorem 1 (V, E) is not connected, contradiction. If (2) does not hold, then there exists $(i, j) \in E$, while there are paths from 1 to i and from 1 to j , both of length k and not containing the edge (i, j) . This gives rise to a cycle, contradiction. If (3) does not hold, then $(i, j), (i, j') \in E$ exist such that $j \neq j'$, and there are paths from 1 to j and from 1 to j' , both of length $k-1$ and not containing the edges $(i, j), (i, j')$. This gives rise to a cycle, contradiction. This proves the ‘only if’-part of the theorem.

Conversely, assume (1), (2) and (3) hold. Then by Theorem 1 (V, E) is connected. For every $i \in V$ define k_i to be the smallest k for which $i \in V_k$. We define $\phi : V \setminus \{1\} \rightarrow E$ by choosing $\phi(i) = (i, j) \in E$ for $j \in V_{k_i-1}$. By definition of V_k such a j always exist, and by (3) it is unique. By construction $k_j = k_i - 1$, and ϕ is injective. Further, according to (2) ϕ is surjective. So ϕ is bijective, so $\#E = \#V - 1$. Combined with connectedness and the standard characterization of trees we conclude that (V, E) is a tree. $\square$

This characterization can be encoded in SAT as follows. By Theorem 2 the graph on n nodes $1, 2, \dots, n$ specified by a_{ij} for $i, j = 1, 2, \dots, n$ is a tree if and only if boolean values for b_{ik} exist for which the formula $C(n-1) \wedge T_2 \wedge T_3$ yields true. Here $C(n-1)$ is defined as before, T_2 is defined to be

$$\bigwedge_{k=1}^{n-1} \bigwedge_{1 \leq i < j \leq n} (\neg b_{ik} \vee b_{i,k-1} \vee \neg b_{jk} \vee b_{j,k-1} \vee \neg a_{ij}),$$

describing requirement (2) of Theorem 2, and T_3 is defined to be

$$\bigwedge_{k=1}^{n-1} \bigwedge_{i=1}^n \bigwedge_{1 \leq j < j' \leq n} \neg(b_{ik} \wedge \neg b_{i,k-1} \wedge a_{ij} \wedge a_{ij'} \wedge b_{j,k-1} \wedge b_{j',k-1}),$$

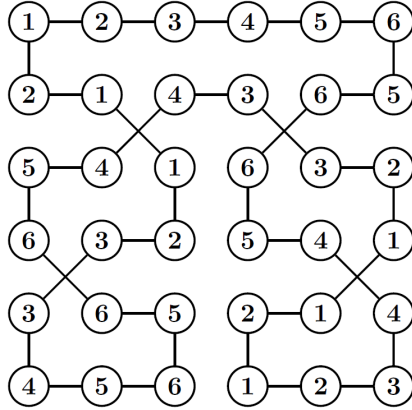
describing requirement (3) of Theorem 2. These formulas apply for trees in a general setting; in the case of Latin squares in which a_{ij} is false for all i, j not

being neighbors the corresponding parts of T_2 and T_3 may be pruned away. In this way the example from the introduction was constructed.

Among all connected graphs, trees can be seen as the sparsest as they have the least number $n - 1$ of edges required for a graph on n nodes to be connected. After removal of any single edge, the graph is not connected any more. On the other side of the spectrum one can think of *bi-connected graphs*: graphs that remain connected after removing any node with all its directly connected edges. Even stronger than bi-connected is *Hamiltonian*: a graph is called Hamiltonian if it admits a Hamilton cycle, that is, a cycle containing every node exactly once, see [5, 1]. The following characterization is an immediate consequence of this definition.

Theorem 3 *The graph (V, E) is Hamiltonian if and only if $E' \subseteq E$ exists such that (V, E') is connected and for every $i = 1, \dots, n$ there are exactly two values $j \in V$ such that $(i, j) \in E'$.*

For encoding this in SAT for connectedness we can use a variant of Theorem 1: for defining V_k based on E' the value k only needs to run until $m = \lfloor n/2 \rfloor$ as in a cycle every node can be reached in at most m steps from the start node 1, while the requirement $V_{n-1} = V$ is replaced by $V_m = V$. Encoding that every node is connected to exactly two others can either be done directly in SAT or by counting in SMT.



The example shown in the picture left is found in this way: note that many edges exist that are not part of the Hamilton cycle; they are not drawn. Also note that this example has a lot of symmetry: for all i, j we have $c_{ij} = 7 - c_{i, 7-j}$ (where c_{ij} is the value on position (i, j)), and even the Hamilton cycle is symmetrical around the vertical axis.

3 Implementation

As a SAT/SMT solver we used the tool **Yices**, [2]. For encoding Latin squares there are two basic approaches:

1. Introduce n^3 boolean variables b_{ijk} for $i, j, k = 1, \dots, n$, where b_{ijk} states that position (i, j) contains number k . Now correctly filling the square is expressed by stating that for every (i, j) there is exactly one k such that b_{ijk} holds. The Latin square requirements are expressed by stating that for every (i, k) there is exactly one j such that b_{ijk} holds and for every (j, k) there is exactly one i such that b_{ijk} holds. Every such “exactly one” is expressed in two parts: “at least one” by a single disjunction, and “at most one” by stating that no two distinct entries are both true.

2. Introduce n^2 integers c_{ij} for $i, j = 1, \dots, n$, where c_{ij} is the value at position (i, j) . For all of these values we specify $1 \leq c_{ij} \leq n$; the Latin square requirements are expressed similar as in case 1.

For both approaches any numbering of the n^2 nodes is chosen, and the relationship between a_{ij} for $i, j = 1, \dots, n^2$ expressing the graph edges and the just defined b_{ijk} or c_{ij} is expressed in an obvious way. For both approaches the graph requirements are expressed as introduced in Section 2. Approach (1) is in pure propositional SAT; approach (2) makes use of SMT, using the theory of linear inequalities. For finding Latin squares with graph properties, both approaches are feasible, while often the first is slightly more efficient. To give an indication, using a standard Linux platform and the encoding based on Theorem 2, a Latin square of size 6 of which the underlying graph is a tree like in the example in the introduction, is found by approach (1) in 4 seconds and by approach (2) in 27 seconds. All examples in this paper were found by approach (2), by inspecting the values of a_{ij} in the resulting satisfying assignment.

This approach also serves to establish that particular Latin squares do not exist. For instance, by adding the requirement that no two diagonal edges cross to the formula encoding a Latin square of size 6 of which the underlying graph is a tree, the formula is established to be unsatisfiable by **Yices**, showing that no such Latin square exists without crossing diagonal edges.

4 Sudoconnect

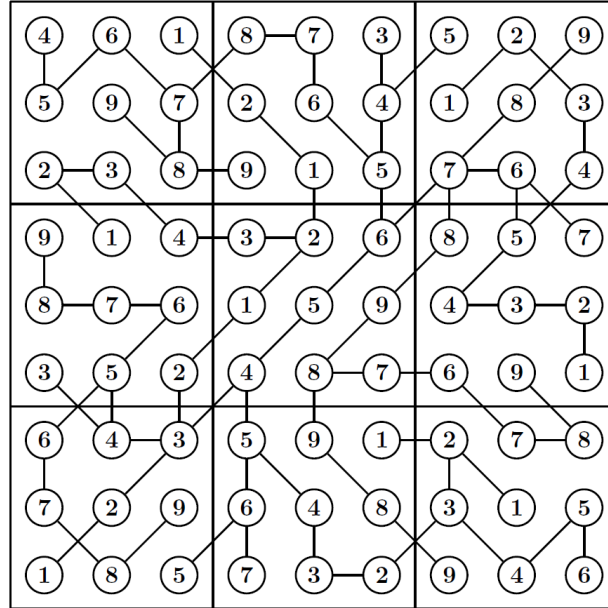
For Latin squares of size $n = k^2$, the square can be split up into n small squares of size k , and for these small squares the requirement can be added that they also contain the numbers 1 to n , all exactly once. For $n = 9$ this coincides with the rules for the popular sudoku puzzle. The question whether there exist Latin squares with this extra requirement for which the underlying graph is connected was the key issue of [6]. Such a Latin square we call a *connected sudoku*. In [6] many solutions for this problem are found by an ad hoc technique of swapping in partial solutions. However, their approach tends to fail when adding more requirements, like symmetry, or moreover requiring that both diagonals contain the numbers 1 to n , all exactly once. By our SAT approach we could easily deal with such extensions. This was the starting point of the Sudoconnect project, where we designed sudoku puzzles having several solutions as a normal sudoku, but having a unique solution with the extra requirement that the underlying graph is connected. So in solving these puzzles both the sudoku rules and connectedness should be exploited. An extensive series of 66 such puzzles are available from <http://www.win.tue.nl/~hzantema/sudc.html>. They all were designed as follows. First a connected sudoku was generated by SAT solving exploiting Theorem 1. In order to get several of them, also a few requirements of the shape $c_{ij} = k$ were added for randomly chosen $i, j, k = 1, \dots, 9$. Here c_{ij} is the value at position (i, j) . Next a formula was generated containing

- all requirements of a connected sudoku,

- $\neg \bigwedge_{i=1}^9 \bigwedge_{j=1}^9 c_{ij} = k_{ij}$ in which k_{ij} is the value on position (i, j) in the particular connected sudoku,
- $\bigwedge_{(i,j) \in A} c_{ij} = k_{ij}$ for some set A of positions, typically showing up a nice pattern and having around 20 elements.

In case the SAT solver states this formula is unsatisfiable, we conclude that the particular connected sudoku is the unique solution of the puzzle obtained by giving the values of this connected sudoku on the positions from A , and a new puzzle has been created. In case the formula is satisfiable, the set A may be extended and the process repeated until the resulting formula is unsatisfiable. As solving the puzzles requires both the sudoku rules and the connectedness requirements, these puzzles most times turn out to be substantially harder than normal sudoku puzzles.

By definition, a connected sudoku satisfies the sudoku rules and has a connected underlying graph. A natural question is whether more requirements can be added, like some symmetry, or requirements on the diagonals, in such a way that there are still solutions. A question asked by Jack van Wijk is: can we find a combination of extra requirements in such a way that up to symmetry exactly one connected sudoku exists? An answer on this question is given in the next picture



Up to symmetry it is the only connected sudoku satisfying the following properties:

- it is symmetric in the sense that $c_{10-i,10-j} = 10 - c_{ij}$ for all $i, j = 1, \dots, 9$,
- the underlying graph is a tree,
- every node is reachable from the middle node in at most 10 steps,

- both diagonals contain the number 1 to 9, all exactly once, and
- the diagonal from left bottom to right top contains the numbers 1 to 9 in increasing order.

This was established by finding a satisfying assignment to the formula combining all requirements (in around 1 minute), and checking that the same formula combined with the negations of this solution and its mirrored version, is unsatisfiable (in around 2 minutes). Finding this result in this way shows the power of SAT solving.

5 Conclusions

By exploiting SAT/SMT solving we found several Latin squares of which the underlying graph has a particular graph property, like being connected, being a tree or being Hamiltonian. For doing so we developed characterizations for these graph properties suitable for encoding in SAT/SMT. In fact these characterizations are unrelated to Latin squares: they can be used in any setting in which requirements expressed in SAT/SMT should be extended by these graph properties.

There are more examples of SAT/SMT being successful in graph theory: in [3] a long-standing lower bound for the Ramsey number $R(4, 8)$ was improved by SAT solving. For more results on Ramsey numbers by SAT see <http://opal.inf.kyushu-u.ac.jp/~fujita/ramsey.html>. For an application of SAT to graph coloring see [4].

Acknowledgments: We thank Walt van Ballegooijen, Nick Heijmink, Hans Kuiper, Jasper Piers and Jack van Wijk for many fruitful discussions on this topic. Among these people extra thanks go to Walt for solving and grading several dozens of sudoconnect puzzles by hand, finally leading to the mentioned series of 66.

References

1. M. Aigner. *Discrete Mathematics*. AMS, 2007.
2. B. Dutertre and L. de Moura. The Yices SMT solver. Available at <http://yices.csl.sri.com/>.
3. H. Fujita. A new lower bound for the Ramsey number $R(4, 8)$. *CoRR*, abs/1212.1328, 2012.
4. A. Van Gelder. Another look at graph coloring via propositional satisfiability. *Discrete Applied Mathematics*, 156(2):230–243, 2008.
5. E. G. Goodaire and M. M. Parmenter. *Discrete Mathematics with Graph Theory*. Prentice Hall, 1998.
6. H. Kuiper and W. van Ballegooijen. 3d sudoku puzzle with 81 connected cubes. In George W. Hart and Reza Sarhangi, editors, *Proceedings of Bridges 2013: Mathematics, Music, Art, Architecture, Culture*, pages 407–410, Phoenix, Arizona, 2013. Tessellations Publishing. Available online at <http://archive.bridgesmathart.org/2013/bridges2013-407.pdf>.