Ludoku: A Game Design Experiment

Cameron Browne, RIKEN Institute

This article provides a practical example of designing a game from scratch, using principles outlined in previous articles in this column: where to start, what to aim for, trouble-shooting the design, and how to evaluate the outcome. The resulting game, called Ludoku, is a Sudoku variant that simplifies the basic Sudoku design while introducing new strategies without adding undue rule complexity.

1 Introduction

IN previous instalments of this *Games Design Patterns* column, I have tried to outline practices conducive to good game and puzzle design [3, 4, 9, 10, 11]. In this instalment, I put my own words into practice, to show how they may be applied to design a new puzzle game.

This article describes the game thus derived, called Ludoku, then goes on to summarise the design process, the game's strengths and weaknesses, and the general success of the exercise.

1.1 Ludoku

Ludoku is a Japanese-style logic puzzle [6] derived from Sudoku [8]. Figure 1 shows a typical challenge with 17 starting hints. The complete rules for playing Ludoku are given in the following blue box.

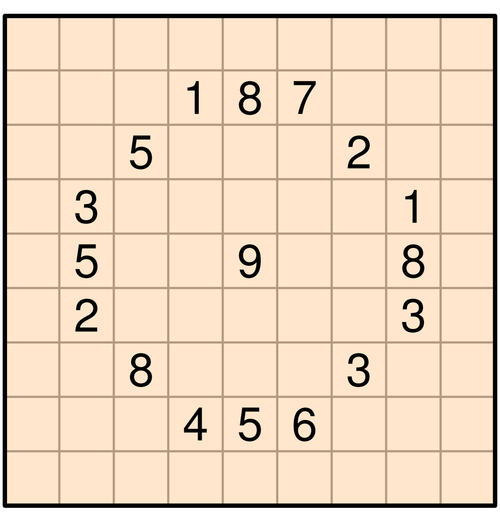

Figure 1. A Luduko challenge with 17 hints.

Ludoku is played on a 9×9 square grid, with some hint values shown. The aim is to fill the grid with numbers 1..9 such that:

1. No number is repeated in any row.

2. No number is repeated in any column.

3. The diagonal neighbours of a number do not repeat that number *or each other*.

Rule 3, the local diagonal neighbourhood rule, is illustrated in Figure 2. Consider the region formed by the diagonal neighbours of the central cell with the value 4 (shaded). No other cell in this region can also contain a 4 (left), and no other cells in this region can contain repeated numbers of any value (right).

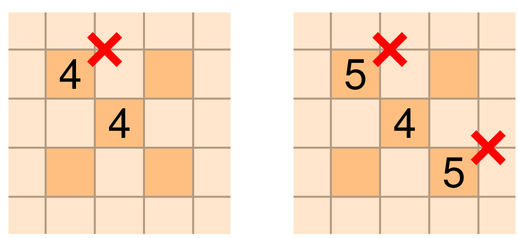

Figure 2. Illegal diagonal neighbours in Ludoku.

This exact Sudoku variant has not been proposed before to my knowledge. Nikoli, the proprietary owner of Sudoku and world's foremost publisher of it and other Japanese logic puzzles, confirm that this design has no precedent that they know of.[1]

2 Design Process

The design process that led to Ludoku followed the basic advice outlined in previous articles in the *Game Design Patterns* series, as follows.

[1]Private communication with Nikoli's chief editor Yoshinao Anpuku.

Browne, C., 'Ludoku: A Game Design Experiment', *G&PD*, vol. 3, no. 2, 2017, pp. 35–46. © 2017

2.1 Reinvent the Wheel

The article 'Reinvent the Wheel' [3] suggests starting the design process with a known design that has proven to be good and then look for ways to modify it. This gives an entry point into the design space that is known to be fruitful.

I chose Sudoku as my starting point for this exercise, as Japanese-style logic puzzles [6] are my favourite type of solitaire puzzle and Suduko is the most widely known example of this genre. Figure 3 shows an example.

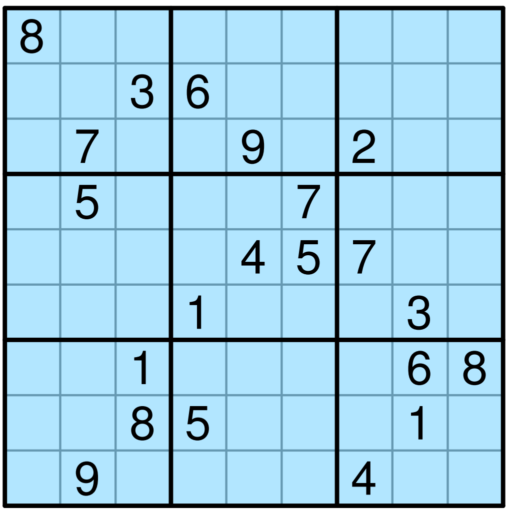

Figure 3. The 'World's Hardest Sudoku' [2].

The fact that so many Sudoku variants already exist [5] suggests that this is a rich region of the game design search space, while the danger is that so many designers have already explored this region that it could be exhausted. Is there scope for yet another Sudoku variant with something new and meaningful to offer?

2.2 Explore the Design Space

Now that the starting point had been decided, I applied the strategy outlined in the article 'Explore the Design Space' [4] and considered the degrees of freedom that could be modified.

The Sudoku Dragon web site[2] lists some of the degrees of freedom that designers have modified over the years to create new Sudoku variants. In almost all cases, Sudoku variants add complexity to the rules (e.g. Diagonal Sudoku [5]), grid design (e.g. Killer Sudoku [6]) or both (e.g. Try [9]), in order to introduce additional constraints and solution strategies. However, I wanted to go in the other direction and *simplify* the design.

I decided to explore variants on a plain 9×9 square grid with the usual 3×3 Sudoku subregions removed, as shown in Figure 4. This would simplify the design at least, and hark back to the puzzle's origins as a Latin square [8].

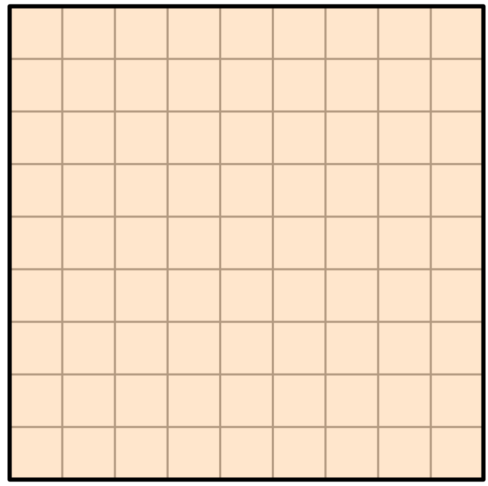

Figure 4. A plain 9×9 square grid.

2.3 Make the Design Do the Work

So what Sudoku-based rules would such a simplified design support? According to the article 'Make the Design do the Work' [9], the rules of a game should be as transparent and intuitive as possible, and flow naturally from the design of the equipment. Instead of giving the player many rules to remember, as few rules as possible should be defined and the design should enforce the rest.

There is not much to work with in a 9×9 square grid apart from orthogonal and diagonal adjacency. Sudoku makes good use of orthogonal adjacency in its row and column rules (Rules 1 and 2 above), so can diagonal adjacent be exploited in a similar way? The simplest such rule change, most in keeping with the existing rules, would be:

3. No number is repeated along any diagonal line.

However, it turns out that such fully diagonal Sudoku packings can only occur on 5×5 and 7×7 square grids, as discussed in Appendix A. This is probably one reason that Diagonal Sudoku [5] only involves the two diagonal constraints across the full board between opposite pairs of corners. I therefore tried a reduced version of this rule:

3. The diagonal neighbours of a number do not repeat that number.

[2] http://www.sudokudragon.com/sudokuvariants.htm

This allows full packings on the 9×9 square grid, exploits diagonal adjacency and is consistent with the game's other rules. Further, this rule effectively replaces the local 3×3 sub-grids in Sudoku with local 3×3 'X'-shaped regions, as highlighted in Figure 2, thus maintaining conceptual consistency with the original game at a more fundamental level. However, a problem with this new rule soon became apparent.

2.4 Bug or Feature?

The article 'Bug or Feature?' [10] promotes awareness of apparent bugs in designs that can be turned around to produce useful features. The problem with the rule listed immediately above is that it allows opposite diagonal neighbours of a cell to have the same number, such as the repeated value 5 in Figure 2 (right). This created some cognitive dissonance, as such repeated diagonals just looked wrong, and felt in violation of the spirit of the diagonal constraint. Each time it occurred, I had to mentally go back over the rules and confirm that it was indeed legal, disrupting the flow of the game.

The solution was to simply forbid such cases, as follows:

3. The diagonal neighbours of a number do not repeat that number *or each other*.

This rule change removed the problem in a consistent and elegant way without adding undue complexity, and introduced new strategies (see Appendices B.2.2, B.2.3 and B.2.5). Turning this bug into a feature was a clear improvement and gave the final rule set shown in Section 1.1.

2.5 Embed the Rules

The article 'Embed the Rules' [11] describes the benefits of having the design of a game's equipment implicitly enforce its rules as much as possible, in order to simplify the rule set and make the design more *poka-yoke* (i.e. mistake-proof). It could be argued that this new Sudoku variant violates this principle by instead simplifying the equipment (by removing the sub-grids) and adapting the rules to suit.

However, note that little complexity is added to the game. The original Sudoku rule 3 (that no number is repeated in any 3×3 sub-grid) is simply replaced by the new rule 3 (that the diagonal neighbours of a number do not repeat that number or each other) and the original local Sudoku constraints (3×3 sub-grids) are replaced by new local constraints (3×3 'X' regions). Further, given that the new diagonal rule implicitly exploits an additional property of the square grid – diagonal adjacency – I would argue that the new design embeds its rules in the equipment at least as much as the original Sudoku design.

Now that the equipment and rules of the new variant had been decided, the game required a name. I chose 'Ludoku' as a contraction of 'Local Sudoku', with the additional bonus that *ludo* is the Latin root for 'play'.

3 Analysis

This section provides a brief analysis of Ludoku and how it differs from Sudoku.

3.1 Distinguishing Features

The most obvious difference between Ludoku and Sudoku is the absence of the 3×3 sub-grids. These are instead effectively replaced with the implicit 3×3 'X' regions due to the new diagonal neighbourhood rule.

Figure 5 shows the three basic region types in Ludoku: rows, columns and 'X' regions. It is worth distinguishing between *global* regions (i.e. rows and columns) that contain each of the numbers 1..9 when completed, and *local* regions ('X' regions) that will only contain five of the numbers 1..9 when completed.

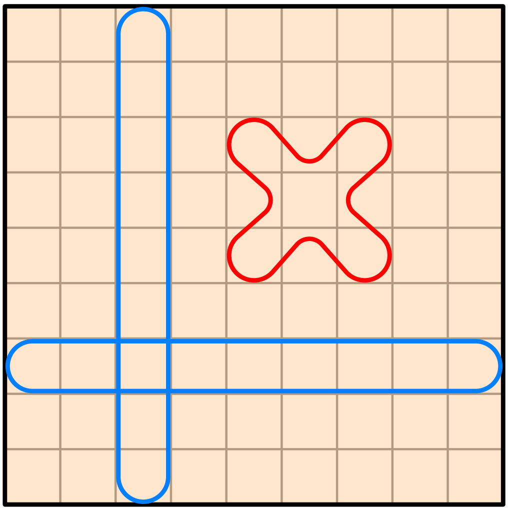

Figure 5. Row, column and 'X' regions.

A key difference between Sudoku's sub-grids and Ludoku's 'X' regions is that no number may be repeated in a Sudoku sub-grid (Figure 6, left) while such formations do not necessarily violate the diagonal neighbourhood rule in Ludoku (Figure 6, right).

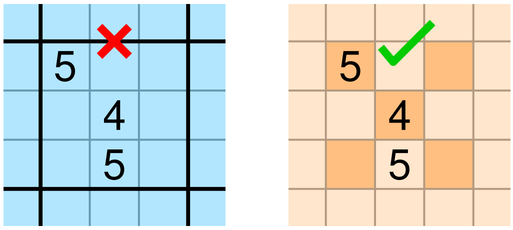

Figure 6. A key difference between Sudoku and Ludoku.

Ludoku's 'X' regions provide weaker constraints for performing deductions than Sudoku's sub-grids, which has implications for the game's strategic depth. Note, however, that there are only nine 3×3 sub-grids in a Sudoku grid while there are 77 'X' regions in a Ludoku grid, one centred on each cell minus the four corners (which are subsets of the 'X's at diagonally adjacent cells). Sudoku has $9 + 9 + 9 = 27$ constraint regions in total to work with while Ludoku has $9 + 9 + 77 = 95$. This far greater number of weaker constraints outweighs any potential loss.

Another feature that highlights the fundamental difference between these two games is that no Sudoku challenge can start with fewer than 17 hints and still remain uniquely deducible,[3] while there exist deducible 15-hint Ludoku challenges, as shown in Figure 7. There may exist deducible Ludoku challenges with even fewer hints; a complete search/analysis has not been done.

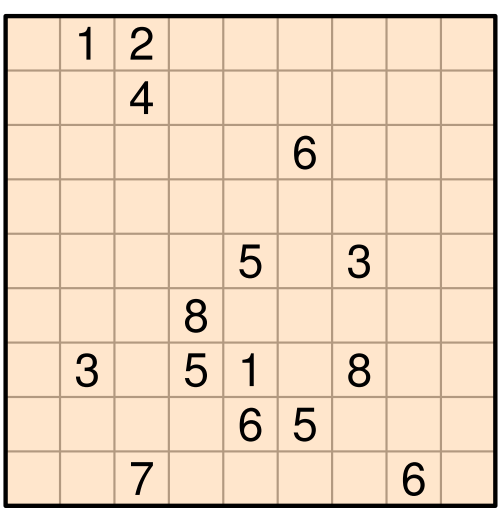

Figure 7. A deducible 15-hint Ludoku challenge.

3.2 Strategic Depth and Deducibility

Ludoku allows most of the basic Sudoku strategies to be applied (except for those specific to the 3×3 sub-grids) plus the addition of several new strategies. Some of these are listed in Appendix sections B.1 and B.2, respectively. In terms of number of strategies, Ludoku could well be strategically deeper than Sudoku.

Importantly, Ludoku balances global constraints provided by the row and column regions and the local constraints provided by the 'X' regions. Such interaction between global and local constraints appears to be central to the success of many logic puzzles.

A logic puzzle is described as *deducible* if it can be solved by applying logical deductive steps to produce a unique solution [13]. Ludoku succeeds in allowing deducible challenges that are interesting to solve, much like Sudoku, using the strategies listed in Appendix B.

The greater number of regions – 95 as opposed to 27 – makes Ludoku harder than Sudoku in general, as players must remain vigilant over a greater number of potential deduction points throughout the game. This greater mental effort is reduced to a manageable level through the judicious use of relevant strategies that encapsulate the side-effects of the new constraints, but there is no denying that Ludoku is hard; the more difficult examples can take an hour or two to solve.

Even the annotated 7×7 sample game listed in Appendix C requires knowledge of the relevant strategies and significant forward planning. For example, consider the sequence of deductive steps leading to the instantiation of the value `2` in Figure 36. This sequence relies on several different regions, both local and global, and apparently unrelated candidate values `4`, `1` and `3` before the eventual `2` is instantiated.

This increased difficulty in Ludoku is both a blessing and a curse. Sudoku enthusiasts looking for new challenges with novel strategies that will push their skills might enjoy Ludoku, but it is unlikely that the average player looking for a mild diversion will persist with it.

3.3 Challenge Design

It is preferable to design Ludoku challenges with their starting hints in symmetrical patterns. Sudoku publisher Nikoli have long maintained that handcrafted challenges are superior to those generated algorithmically [14], and symmetric hint placement is an indicator of handcrafted design. Even when challenges *are* generated by computer, incorporating symmetry can help give the impression of handcrafted design [15]. Symmetric hint placement is especially important in Ludoku, as the absence of 3×3 sub-grids makes the starting hints the only way to give challenges structure.

[3] Proven by McQuire *et al.* [12] in a 7.1 million hour search performed over one year on a supercomputer cluster.

3	2	4	4		4	4	2	3
2	3	5		6		5	3	2
4	5		7	8	7		5	4
4		7	9		9	7		4
	6	8		12		8	6	
4		7	9		9	7		4
4	5		7	8	7		5	4
2	3	5		6		5	3	2
3	2	4	4		4	4	2	3

Figure 8. Cell totals for the 'diamond' design.

5	4	4		6		4	4	5
4	5	6		4		6	5	4
4	6		7	4	7		6	4
		7	11	4	11	7		
6	4	4	4	4	4	4	4	6
		7	11	4	11	7		
4	6		7	4	7		6	4
4	5	6		4		6	5	4
5	4	4		6		4	4	5

Figure 9. Cell totals for the 'asymptotes' design.

1	3	3	3	7	3	3	3	1
3	7	5				5	7	3
3	5		6	7	6		5	2
3		6	8	5	8	6		3
7		7	5		5	7		7
3		6	8	5	8	6		3
3	5		6	7	6		5	2
3	7	5				5	7	3
1	3	3	3	7	3	3	3	1

Figure 10. Cell totals for the 'circle' design.

Figures 8, 9 and 10 show some templates for generating Ludoku challenges with symmetric hint patterns. Black cells indicate positions of starting hints, and the number in each empty cell shows the total number of starting hints that the cell shares a constraint region with, indicating the amount of deductive information available to each cell. If a cell shares constraint regions with eight or more starting hints, then its value can be immediately instantiated if those starting hints contain eight different numbers. Higher values indicate greater constraint, and in logic puzzles it is usually beneficial to focus on the point of most constraint at each step.

The different distributions of cell totals give each pattern a different character. For example, the 'diamond' design shown in Figure 8 has a high-value cell at its centre with 12 shared starting hints that is likely to be deduced, but the available information dissipates quickly the farther a cell is from the centre. Solving challenges based on this pattern would typically involve focussing on the centre then solving outwards. The 'asymptotes' design shown in Figure 9, conversely, has a low-value centre cell surrounded by four high-value neighbours that would be the sensible starting points for solution.

The 'circle' design shown in Figure 10 is more rounded, so to speak, with a reasonably homogenous distribution of cell totals over most of its area, apart from the outermost cells. This is the most pleasing design found so far, both aesthetically and in terms of deductive flow during solution of the challenges that it produces.

4 Generation

The Ludoku challenges shown in this paper, and printed throughout this issue, were generated algorithmically using the following approach:

1. Generate a random packing of numbers that satisfies the Ludoku region constraints.
2. Choose a starting hint set as follows (with equal probability):
   (a) A pre-defined pattern (Figure 1).
   (b) Iteratively removing hints in a symmetric pattern (Figure 24).
   (c) Iteratively removing single hints (Figure 7).
3. If the final hint pattern provides a deducible challenge, then:
   (a) Evaluate the challenge.
   (b) Store the challenge to file.

In steps 2(a) and 2(b), hint patterns were iteratively reduced as long as the challenge remained deducible using the strategies listed in Appendix B. This process generates around one deducible challenge per second per thread on a typical laptop computer.

Challenges were evaluated for quality by recording the sequence of strategies applied and applying the following calculation:

$$quality = variety + degree - help \qquad (1)$$

where *variety* is the number of times the player must apply a different strategy to progress in the solution (to encourage interplay between strategies), *degree* is the minimum number of times any one strategy is applied (to encourage the use of all strategies), and *help* is based on the number of starting hints (to reward fewer hints).

This quality estimate gives some indication of the strategic depth and difficulty of challenges, but does not always capture how truly difficult a challenge is for the human player. This measurement was used to indicate potentially interesting challenges, with high-scoring examples then being hand-tested for more accurate evaluation.

Note that the automated solution process was based entirely on the strategies listed in Appendix B and not the more thorough *deductive search* technique devised to solve and evaluate player difficulty for general deduction problems [13]. This is because the strategies implemented already encoded the key deductive steps that players could be expected to make for this game, and the challenges thus generated already proved difficult enough without considering higher levels of deductive embedding. 'Easy' 9×9 challenges can still take 20-30 minutes while 'hard' challenges can take up to 1-2 hours to manually solve.

5 Conclusion

Previously described *Game Design Patterns* were successfully applied to create the Ludoku deduction puzzle, an apparently novel Sudoku variant that simplifies the board design and introduces new strategies without adding undue rule complexity. Ludoku could be strategically deeper than Sudoku but its distribution of local constraints over the entire grid (rather than concentrated in just nine sub-grids) makes it harder work for players and more difficult to solve.

On the positive side, I find Ludoku to be an interesting puzzle that is absorbing and enjoyable – if challenging! – to solve. On the negative side, it will probably be *too* challenging for most players, and is in the end just another Sudoku variant. However, I am generally satisfied with the result of this game design experiment.

Acknowledgements

This work was conducted while the author was a member of the RIKEN Institute's Advanced Intelligence Project (AIP).

References

[1] *The Times Japanese Logic Puzzles: Hitori, Hashi, Slitherlink and Mosaic*, London, Harper Collins, 2006.

[2] Collins, N., 'World's Hardest Sudoku: Can You Crack It?', *The Telegraph: Science*, 28 June 2012.

[3] Browne, C., 'Reinvent the Wheel', *Game & Puzzle Design*, vol. 3, no. 1, 2017, pp. 51–62.

[4] Browne, C., 'Explore the Design Space', *Game & Puzzle Design*, vol. 1, no. 2, 2015, pp. 71–83.

[5] 'Sudoku Rules', *Conceptis Puzzles*, 2018. http://www.conceptispuzzles.com/index.aspx?uri=puzzle/sudoku/rules

[6] Lewis, L., 'Too Good for Fiendish? Then Try Killer Su Doku', *The Times*, 31 August 2005.

[7] Browne, C., 'Try: A Hybrid Puzzle/Game', *Game & Puzzle Design*, vol. 1, no. 2, 2015, pp. 21–27.

[8] Rosenhouse, J. and Taalman, L., *Taking Sudoku Seriously: The Math Behind the World's Most Popular Pencil Puzzle*, Oxford, Oxford University Press, 2011.

[9] Browne, C., 'Make the Design Do the Work', *Game & Puzzle Design*, vol. 2, no. 2, 2016, pp. 27–40.

[10] Browne, C., 'Bug or Feature?', *Game & Puzzle Design*, vol. 2, no. 1, 2016, pp. 61–69.

[11] Browne, C., 'Embed the Rules', *Game & Puzzle Design*, vol. 1, no. 1, 2015, pp. 60–70.

[12] McGuire, G., Tugemanny, B. and Civario, G., 'There is No 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem via Hitting Set Enumeration', *arXiv*, 2013. http://arxiv.org/pdf/1201.0749.pdf

[13] Browne, C., 'Deductive Search for Logic Puzzles', in *Proceedings of Computational Intelligence & Games (CIG '13)*, Niagara Falls, IEEE, 2013, pp. 1–8.

[14] Kanamoto, N., 'A Well-Made Sudoku is a Pleasure to Solve', *Nikoli*, 2001. http://www.nikoli.co.jp/en/puzzles/why_hand_made.html

[15] Browne, C., 'Metrics for Better Puzzles', in El-Nasr, M. S., Drachen, A., Canossa, A. and Isbister, K. (eds.), *Game Analytics: Maximizing the Value of Player Data*, Berlin, Springer, 2013, pp. 769–800.

Cameron Browne is a Research Scientist at the RIKEN Institute's Advanced Intelligence Project (AIP) in Tokyo. His research interests include artificial intelligence and automated game design. **Address:** Mitsui Building, 15th floor, 1-4-1 Nihonbashi, Tokyo 103-0027, Japan. **Email:** cameron.browne@riken.jp

Appendix A. There is No Fully Diagonal 9×9 Sudoku

The question raised in Section 2.3 is whether an $n \times n$ square grid can be filled with numbers $1..n$ such that no number is repeated in any row, column or along any diagonal line within the grid. Let us call such a packing a *fully diagonal Sudoku packing*. Two $n \times n$ square grids that allow such packings are sizes $n = 5$ and $n = 7$.

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

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

Figure 11. Fully diagonal 5×5 Sudoku packings.

Figure 11 shows the two possible unique fully diagonal Sudoku packings on the 5×5 square grid (other packings may be derived by permutations of the number sets 1..5). These correspond to cyclic packings in which the offset distance d for each row r and column c is given by:

$$d = (2r + c)\ \%\ n \tag{2}$$

$$d = (3r + c)\ \%\ n \tag{3}$$

Figure 12 shows the four possible unique fully diagonal Sudoku packings on the 7×7 square grid (other packings may be derived by permutations of the number sets 1..7). These correspond to cyclic packings in which the offset distance d for each row r and column c is given by:

$$d = (2r + c)\ \%\ n \tag{4}$$

$$d = (3r + c)\ \%\ n \tag{5}$$

$$d = (4r + c)\ \%\ n \tag{6}$$

$$d = (5r + c)\ \%\ n \tag{7}$$

This problem is identical to the n^2-*Queen Colouring Problem*, for which it has been shown that it is not possible to superimpose nine different solutions to the *n-Queens Problem* on a 9×9 grid.[4] Hence, it would not be possible to derive any fully diagonal 9×9 Sukoku challenges using the initial rule set outlined in Section 2.3.

1	2	3	4	5	6	7
6	7	1	2	3	4	5
4	5	6	7	1	2	3
2	3	4	5	6	7	1
7	1	2	3	4	5	6
5	6	7	1	3	3	4
3	4	5	6	7	1	2

1	2	3	4	5	6	7
5	6	7	1	2	3	4
2	3	4	5	6	7	1
6	7	1	2	3	4	5
3	4	5	6	7	1	2
7	1	2	3	4	5	6
4	5	6	7	1	2	3

1	2	3	4	5	6	7
4	5	6	7	1	2	3
7	1	2	3	4	5	6
3	4	5	6	7	1	2
6	7	1	2	3	4	5
2	3	4	5	6	7	1
5	6	7	1	2	3	4

1	2	3	4	5	6	7
3	4	5	6	7	1	2
5	6	7	1	2	3	4
7	1	2	3	4	5	6
2	3	4	5	6	7	1
4	5	6	7	1	2	3
6	7	1	2	3	4	5

Figure 12. Fully diagonal 7×7 Sudoku packings.

Appendix B. Solution Strategies

This appendix describes some key strategies for solving Ludoku challenges. A general rule of thumb for tackling logic problems can also save time: *focus on the most constrained point at each step.*

B.1 Regular Sudoku Strategies

The following basic Sudoku strategies also apply to Ludoku.

B.1.1 Eliminate by Region

When the value of a cell is known, then that value can be eliminated as a candidate from all other cells with which it shares a region. For example, the value 3 shown in Figure 13 can be eliminated from the other cells shown.

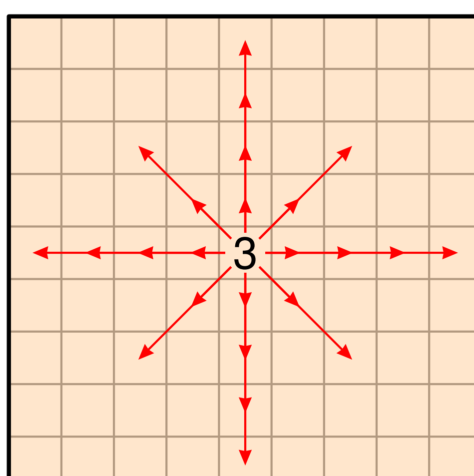

Figure 13. Remove candidate 3s from other cells.

[4] Vašek Chvátal, 'Colouring the Queen Graphs': http://users.encs.concordia.ca/ chvatal/queengraphs.html

B.1.2 Instantiate by Cell

If the number of candidate values for a given cell has been reduced to a single possibility, then that value can be instantiated at that cell. For example, Figure 14 shows a cell that must be 9.

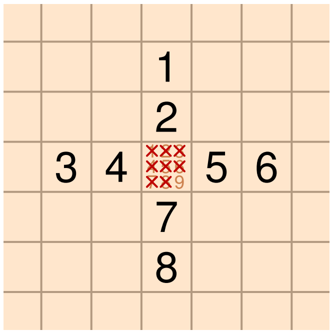

Figure 14. The centre cell must be 9.

B.1.3 Instantiate by Region

If a given value can only occur in one possible cell within a global region, i.e. row or column, then that value can be instantiated at that cell. For example, Figure 15 shows a case in which the cell marked '**?**' within the row must take the value 4.

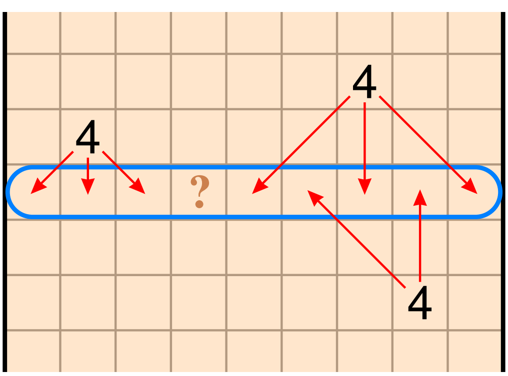

Figure 15. The cell marked '**?**' must be 4.

B.1.4 Pairs of Pairs by Region

If the candidate sets for two cells within a region are reduced to the same two candidate values, then those values can be removed as candidates from all other cells within that region. For example, the row shown in Figure 16 has two cells reduced to candidates 4 and 5, hence these values can be eliminated from other cells in the region.

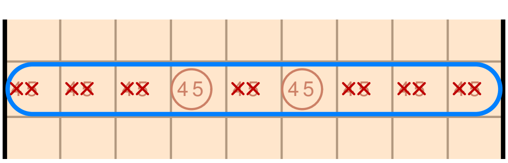

Figure 16. 4 and 5 can be eliminated as shown.

B.1.7 Cross-Elimination

If two candidate values can only occur at the same two positions in any two rows or columns, then that value can be eliminated from those positions in any other columns or rows.

For example, consider Figure 17, which shows the coverage of known 4s in this example. It may not seem that any other 4s can be immediately instantiated from here.

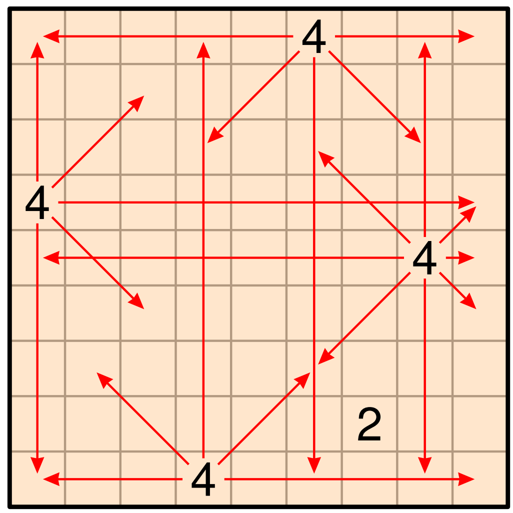

Figure 17. Coverage of known 4s.

However, 4s only occur on the same two rows (three and seven) of the two columns highlighted in Figure 18. The value 4 can therefore be eliminated from other cells along these two rows as shown, allowing another 4 to be instantiated.

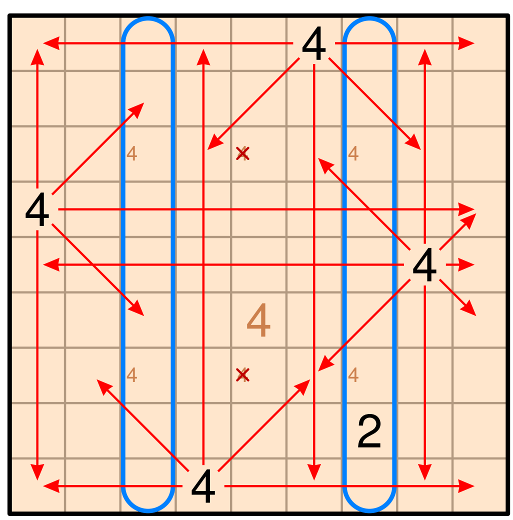

Figure 18. 4s can be eliminated from two rows.

B.2 Ludoku-Specific Strategies

The following strategies, particular to Ludoku, are based on local diagonal relationships between cells. The rationale behind these is to eliminate candidate placements that that would incorrectly eliminate neighbouring values from their respective row or column due to local diagonals.

B.2.1 1-Step Pairs

If a given value can only occur in two adjacent cells within a given row or column, then that value can be eliminated from diagonally adjacent cells, as shown in Figure 19.

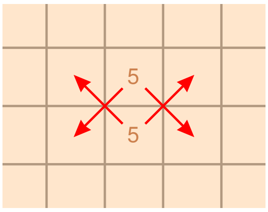

Figure 19. 5s can be eliminated.

B.2.2 2-Step Pairs

If a given value can only occur in two cells separated by one intervening cell within a given row or column, then that value can be eliminated from diagonally adjacent cells up to two steps away, as shown in Figure 20.

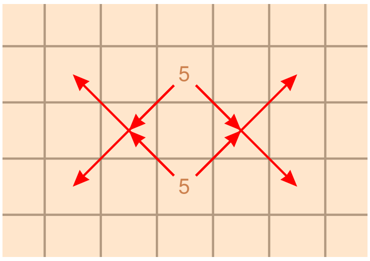

Figure 20. 5s can be eliminated.

B.2.3 4-Step Pairs

If a given value can only occur in two cells separated by three intervening cells within a given row or column, then that value can be eliminated from diagonally adjacent cells exactly two steps away, as shown in Figure 21.

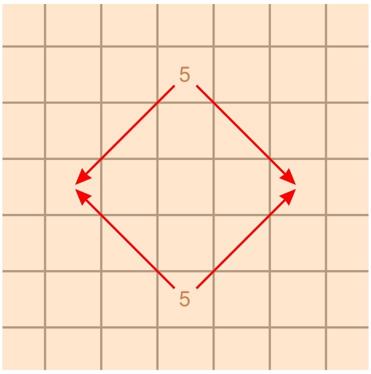

Figure 21. 5s can be eliminated.

B.2.4 1-Step Triplets

If a given value can only occur in three consecutive cells within a given row or column, then that value can be eliminated from the common diagonally adjacent cells, as shown in Figure 22.

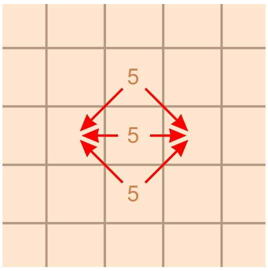

Figure 22. 5s can be eliminated.

B.2.5 2-Step Triplets

If a given value can only occur in three cells within a given row or column, each separated by an intervening cell, then that value can be eliminated from diagonally adjacent cells exactly two steps away, as shown in Figure 23.

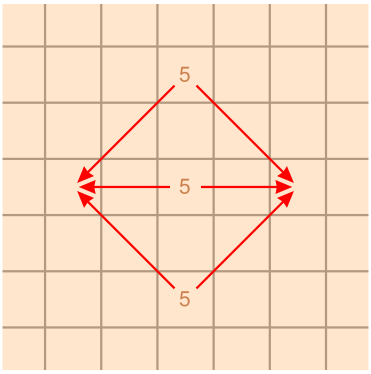

Figure 23. 5s can be eliminated.

Appendix C. Worked Example

This appendix provides a worked example of a 7×7 Ludoku challenge (Figure 24) that shows most of the deductive strategies listed in Appendix B in action. Note that even though this challenge is smaller than the standard size, it is still quite difficult. Also note the constant interplay between local and global constraints in allowing deductions.

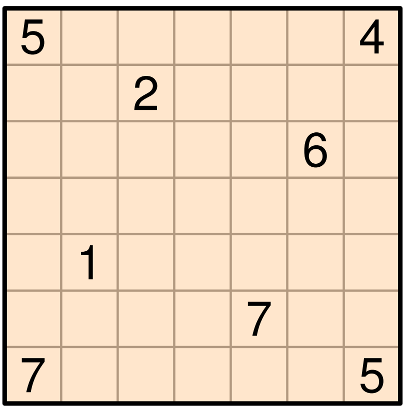

Figure 24. Example 7×7 Ludoku challenge.

A 7 can immediately be instantiated by region, along the third row (Figure 25). Note that rows are numbered from bottom to top.

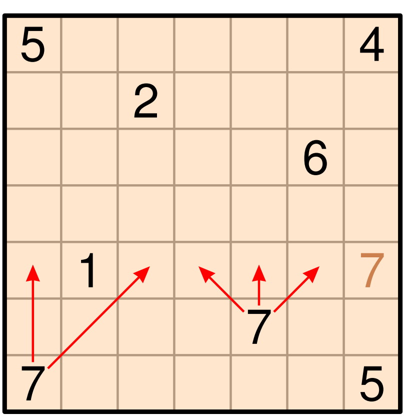

Figure 25. 7 can be instantiated.

A 6 can then be instantiated along the seventh column (Figure 26).

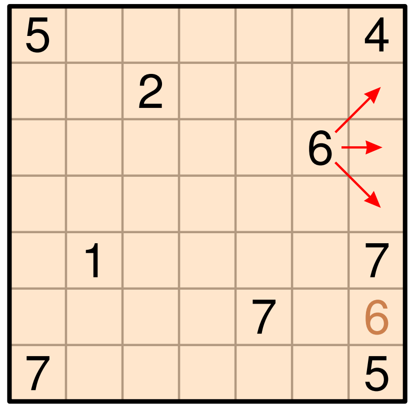

Figure 26. 6 can be instantiated.

The coverage of known 7s means that 7 can only occur in a 2-step pair along the fourth row (Figure 27)...

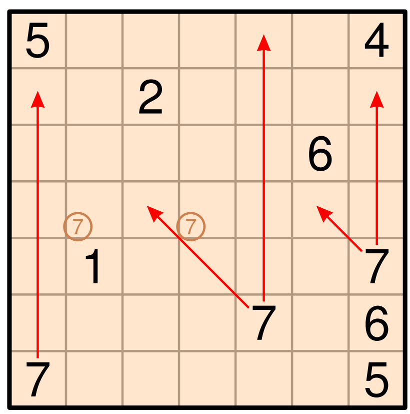

Figure 27. A 2-step pair of 7s.

...allowing two potential 7s to be eliminated from the sixth row and a further 7 to be instantiated (Figure 28).

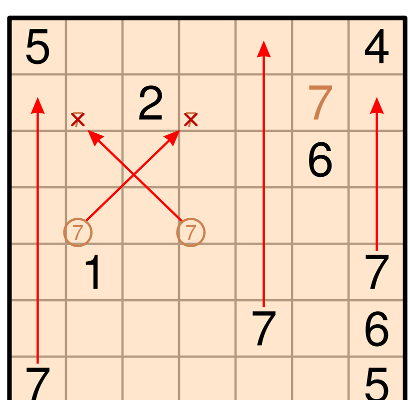

Figure 28. 2-step pair elimination to give a 7.

This new 7 eliminates one of the 2-step pair to allow another 7 to be instantiated (Figure 29)...

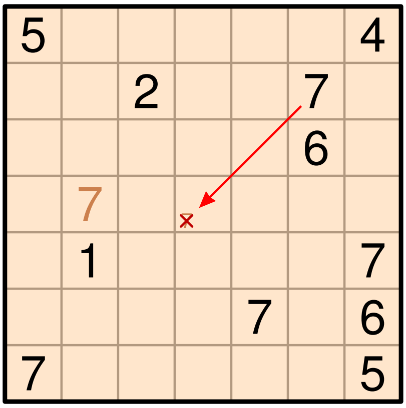

Figure 29. Another 7 can be instantiated.

...which in turn allows a 7 to be instantiated on the fifth row (Figure 30). The final 7 can then be trivially instantiated on the seventh row.

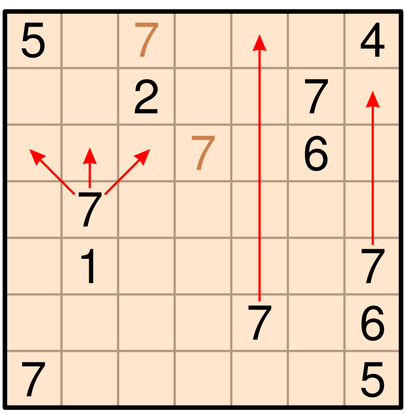

Figure 30. Another 7 can be instantiated.

Only two 5s can exist in the sixth column in a 1-step pair, allowing the elimination of candidate 5s from neighbouring cells (Figure 31).

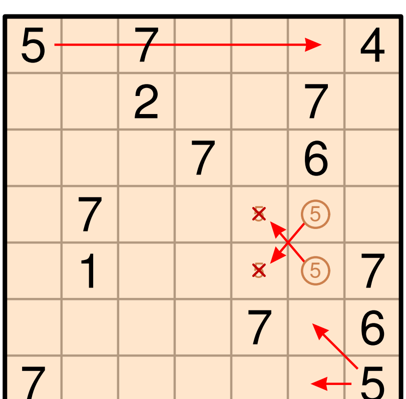

Figure 31. A 2-step pair of 5s.

This produces another 1-step pair of 5, in the fifth column, which eliminates another neighbouring candidate 5 (Figure 32).

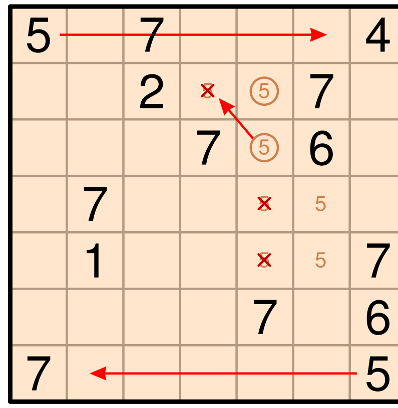

Figure 32. Another 2-step pair of 5s.

This allows a 5 to be instantiated on the six row (Figure 33).

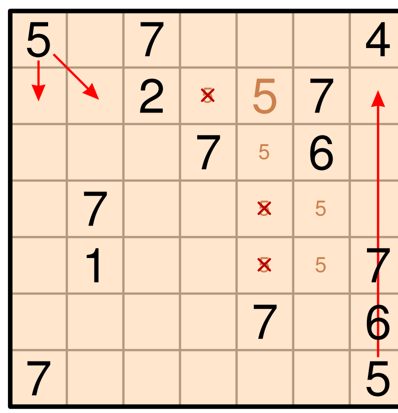

Figure 33. 5 can be instantiated..

A similar process can be applied to deduce the positions of the remaining 5s (Figure 34).

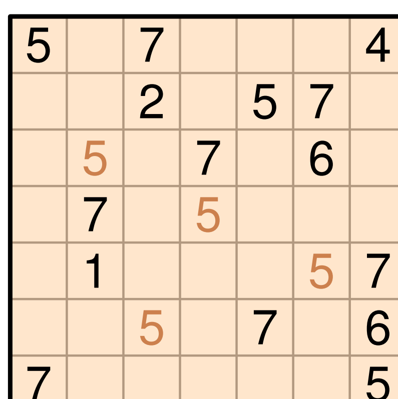

Figure 34. Positions of remaining 5s.

Candidate 4s can then be reduced to a 2-step pair in the fifth row, which reduces candidate 4s to a 1-step pair in the third row, which eliminates a neighbouring candidate 4 above (Figure 35).

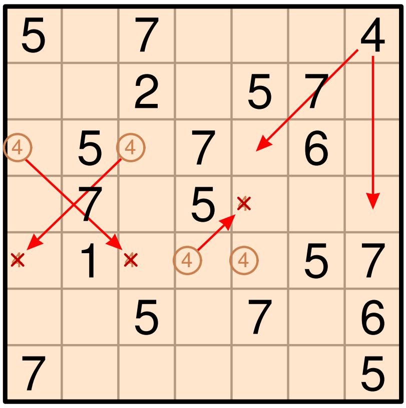

Figure 35. 2-step pairs of 4s.

The two cells circled in Figure 36 can then be reduced to candidates 1 and 3, and the cell thus triangulated must take the value 2.

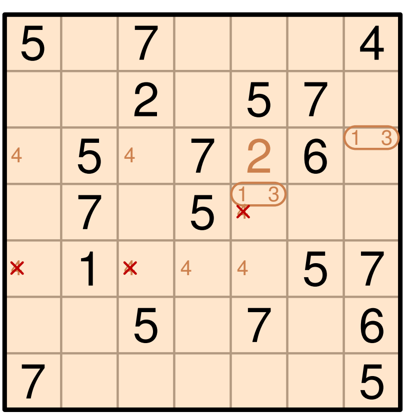

Figure 36. Instantiation due to pairs of pairs.

This leads to the immediate instantiation of a nearby 2 (Figure 38)...

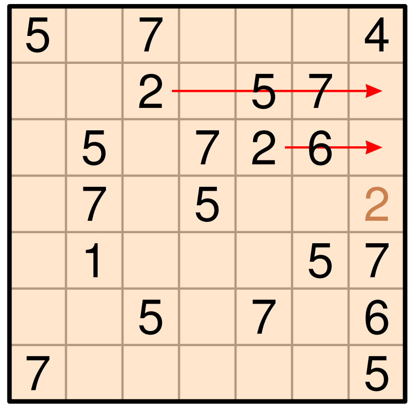

Figure 37. Trivial instantiation of a 2.

...and the deduction of another 2, through an elimination due to a 1-step pair (Figure 38).

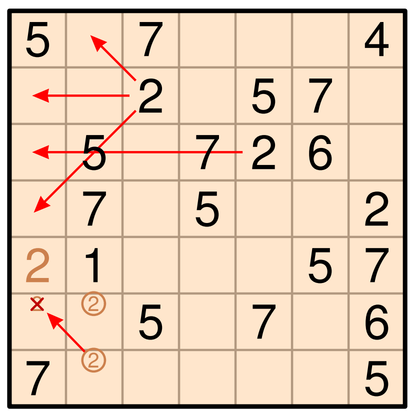

Figure 38. More involved deduction of a 2.

And so on, until the final solution (Figure 39).

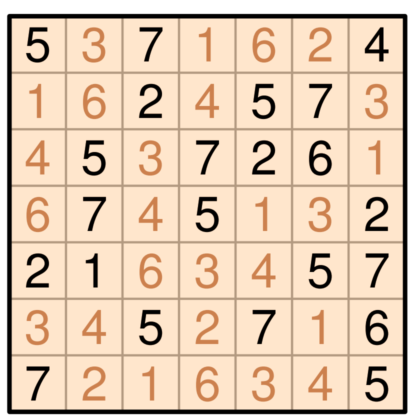

Figure 39. Final solution.