

# About 4x4 sudokus

by

**Siro Brotón Gutiérrez**

**zID: z5627460**

Email: [z5627460@ad.unsw.edu.au](mailto:z5627460@ad.unsw.edu.au)



**UNSW**  
A U S T R A L I A

## **Contents**

|          |                                |          |
|----------|--------------------------------|----------|
| <b>1</b> | <b>Introduction</b>            | <b>2</b> |
| <b>2</b> | <b>Generating the families</b> | <b>2</b> |
| <b>3</b> | <b>Family relationships</b>    | <b>3</b> |
| <b>4</b> | <b>Inputs and solutions</b>    | <b>4</b> |
| <b>5</b> | <b>Results</b>                 | <b>5</b> |
| <b>6</b> | <b>Other figures</b>           | <b>6</b> |

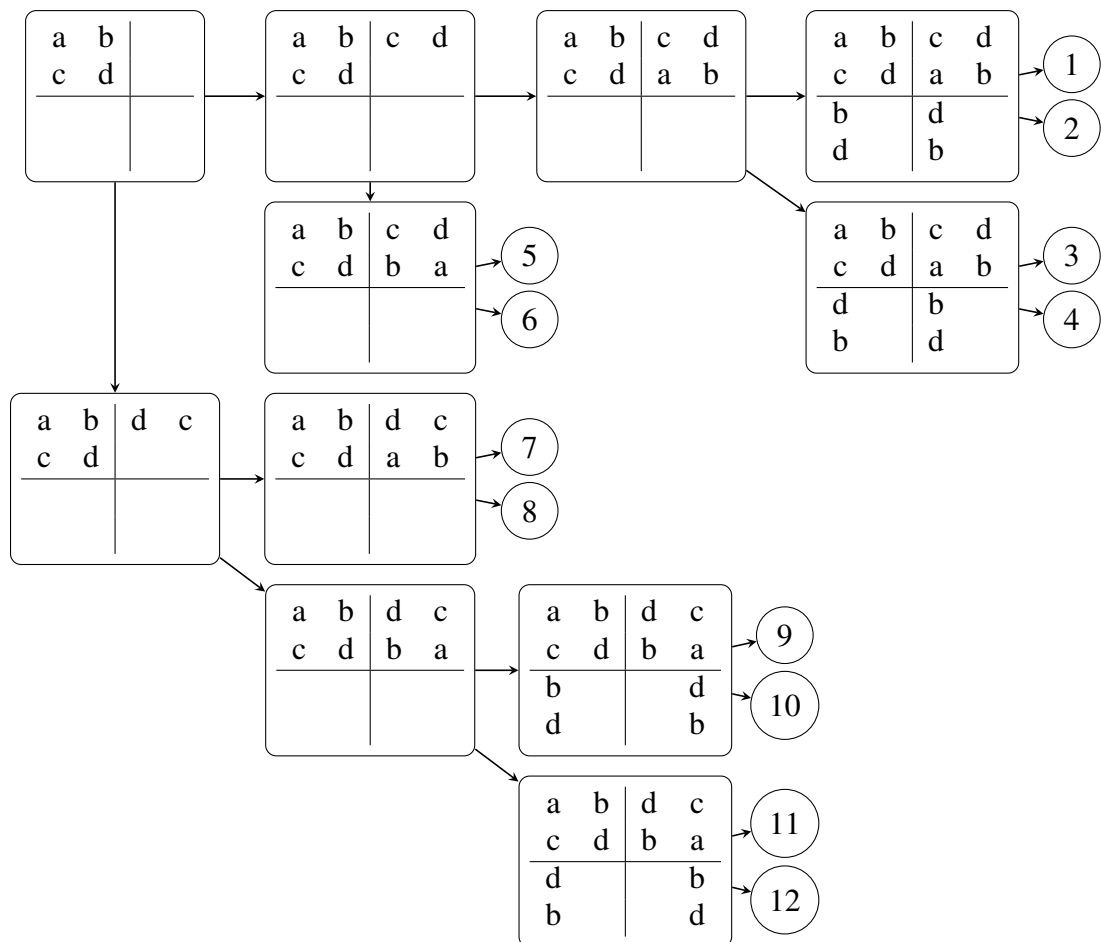
# 1 Introduction

In this paper I plan to dive deeper into the different families of solutions for a 4x4 sudoku, how they relate to one another and how they help find the fitting solution for any given input. For this purpose we will first identify all the families, analyze their properties and find how these can help us connect them. We are also going to investigate about what kinds of input can be given so that there is only one solution and the general steps to finding it.

## 2 Generating the families

The first thing we can notice is that the exact symbols of a solution don't really matter, so we are going to replace them in such a way that 2 solutions with the same general structure will have the same representation. We can do this by fixing the top-left quadrant so that it is always the same,  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ , and substitute the rest of the numbers of the solution following the code created by that first quadrant substitution. Doing this we get that both of the solutions shown in the previous assignment would fall into the same family because they have the same representation when we change the numbers to letters.

The next step is be to find all possible families, which we can do with a tree:



Because each of the 4 letters can be any of the 4 numbers we get a 4! solutions in each family for a total of 288 possible solutions.

### 3 Family relationships

Now that we have identified all families we can analyze the position of each of the letters in each of the families to better understand the relationships between them. Connecting the positions of every instance of a letter gives us a table from which we can extract some useful information (See tables 1 and 2 in section Other figures).

We can differentiate 4 types of shape: rhombus (we will call them r1 and r2 in order of appearance), squares (they will be s1 and s2), kites (they will be k1-k8) and triangles (these will be t1, t2, t3 and t4). Notice that some shapes are incompatible, meaning that they never appear together in a solution (like squares with triangles or certain kites with rhombuses). Also any 2 shapes that appear from the same letter (r1, k1, k5 and t1 are all shapes formed by As) are also automatically incompatible because they would both occupy the same square at least in the first quadrant (r1, k1, k5 and t1 all need the A in the top-left).

(THIS IS PROBABLY NOT USED IN THE FINAL SOLUTION BUT I FIND IT POSSIBLY USEFUL. YOU CAN SKIP TO SOLUTIONS) Also notice that in each type of shape there is a way to go from any of its elements to any other applying clockwise rotations and vertical or horizontal reflections (See figure below), and so any 2 families of solutions with the same set of components (for example any 2 kites with any 2 triangles; or 1 rhombus, 1 square and 2 kites) can also be transformed into one another with those same rotations and reflections. For convenience I'm going to define some abbreviations for the operations:

- c = clockwise rotation
- rh = horizontal reflection (reflect on a vertical axis)
- rv = vertical reflection (reflect on an horizontal axis)
- ref = any of the previous two reflections
- any = any operation

If we apply all possible transformations to all the shapes we can draw the relations between them in a graph like this one:

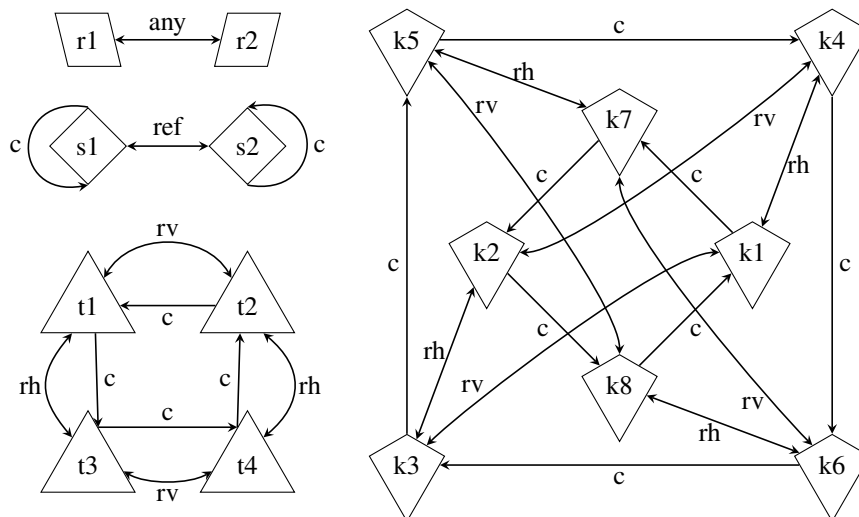


Figure 1: Relations between shapes

We can see that if by applying the same transformation to all the shapes of a family they coincide with the shapes of another family, then applying that transformation to the family as a whole would result in that other family. For example: family 2 has {k1, s1, k2 and r2}. If we apply a rotation to all the shapes we would get {k7, s1, k8, r1} which

coincides with the shapes of family 7, meaning a rotation of a solution in family 2 gives a solution of the family 7.

Mapping all of these relationships is an extremely tedious job, but doing it we can get to the realization that there are only 5 "true families" (See figure below). We can also realize that in all these graphs the "longest shortest path" between any 2 nodes has length 2 (even if we discard one of the reflections), meaning that we can just "memorize" one solution of each family and when we are given a solution we can relate it to one in memory in 2 steps or less (4 operations at most with breadth first search).

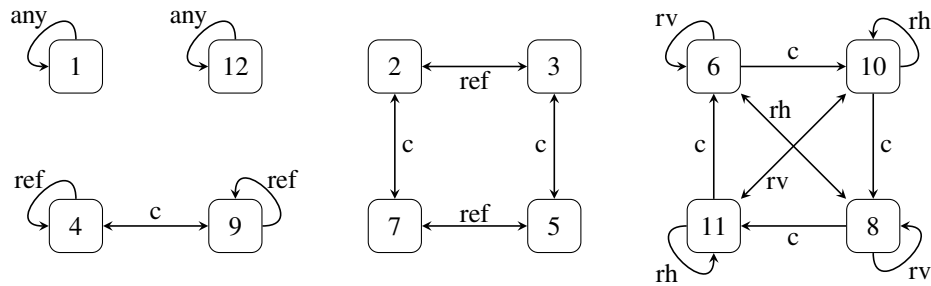


Figure 2: Relations between families

## 4 Inputs and solutions

Now that we know how all possible solutions that work we have to study how the different inputs to our model might behave. We will define an input as correct if there is at least one solution reachable from that input by filling the grid. We will also define an input as minimal if by removing any of the numbers the solution ceases to be unique. Turns out the minimal input with the lowest amount of numbers has only 4, but because I don't have a proof or a consistent way to create those the model will normally use non-minimal inputs for testing and then hope it still works for the extreme cases.

The main objective of this project was to find a way to solve a sudoku in a more or less reasonable time, so after all that preamble we can finally start creating our solver seeing different approaches. I highly encourage following my attempt to explain them with pen and paper.

- The first approach would be a depth-first search and trying combinations until we reach a solution. Completely unfeasible.
- The second approach is to use our knowledge about solutions to try to fit them over the input and if each letter coincides with every instance of a number then we output the known solution with the letters changed to the corresponding number. This approach needs an average of 6,5 attempts to guess the correct result. Let's see an example:

We get this input 

|   |   |
|---|---|
| 4 | 3 |
| 3 | 2 |

 and superimpose one of the known solutions, for example 

|   |   |   |   |
|---|---|---|---|
| a | b | c | d |
| c | d | a | b |
| b | a | d | c |
| d | c | b | a |

 (1). We can see that the *Ds* overlap with both 3s but also a 4

which means this can't be a solution. We keep trying until we get to

|   |   |   |   |
|---|---|---|---|
| a | b | d | c |
| c | d | b | a |
| d | a | c | b |
| b | c | a | d |

(11). See that now the *B*s only overlap 2s, the *C*s overlap all the 3s and the *D*s overlap only 4s. Now change all the *A*s with 1s, *B*s with 2s, *C*s with 3s and *D*s with 4s. We have found a solution but we don't know if it is the only one.

Knowing that there exist isomorphisms relating the families we can optimize memory usage if we also build a good function to rotate and reflect matrices and just store one example of each of the families. This adds complexity and probably reduces performance.

In this approach we would have 2 steps: superimpose solutions and compare until we find a solution; "translate" our theoretical solution to an actual solution with actual numbers.

- The third approach relies in the same concept of superimposing but now we do it with shapes instead of the whole family. We will say a shape "fits" if the squares it occupies have the same number and there are no more instances of that number. In this approach we have the potential of eliminating up to 3 families with one comparison and said comparisons are way cheaper. To minimize the number of comparisons we should avoid discarding a family multiple times, which we can do by checking sets of incompatible shapes. We can start by trying to fit all the shapes of one of the letters and then seeing what families we have left. If needed we can try the same for a different letter. After 2 letters (8 comparisons) we usually have found a solution but we can do one more if needed (the last letter is never needed. Proof is left to the reader). Let's see an example:

We get this input

|   |   |   |
|---|---|---|
| 2 | 3 |   |
|   | 1 |   |
|   |   | 3 |

. Try the shapes produced by the letter A (r1, k1,

k5 and t1): r1 doesn't fit because it overlaps a 2 and a 3, discard {1,3,7}; k1 fits; k5 doesn't fit because it overlaps a 2 and a 1, discard {5,9,11}; t1 fits. We have reduced the possible families to {2,4,6,8,10,12}. Try the *B*: s1 doesn't fit because it doesn't overlaps all 3s, discard {2}; k3 doesn't fit for the same reason, discard {4,8}; same for t2, discard {6,12}; k6 fits. We are left with {10}. We have a unique solution.

If we also build into the model some of the incompatibilities we can skip some of the shapes, but I don't think this will improve performance in a significant enough way to justify the time needed to implement it.

In this approach we would also have 2 steps: superimpose shapes and discard families; translate the theoretical solutions to the real solution.

## 5 Results

I firmly believe that the third approach is faster than the other two so that's the one I plan on implementing, but I can't give states more detailed than the steps of the algorithm without spending hours drawing. The main problem I have found is that if you were to do the same procedure for 3x3 sudokus you would get around  $10^{16}$  families, so it is not very scalable.

## 6 Other figures

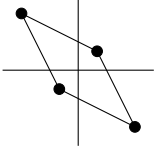
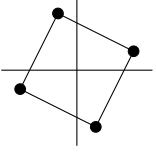
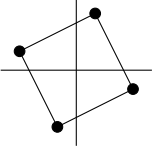
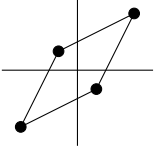
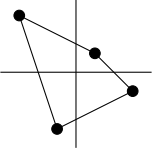
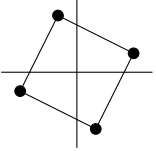
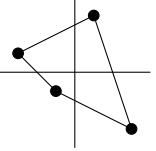
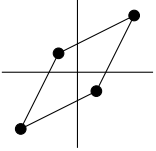
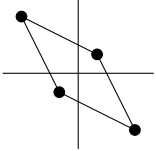
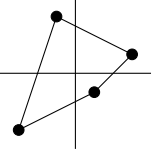
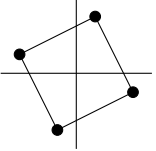
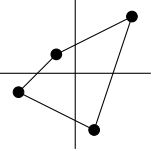
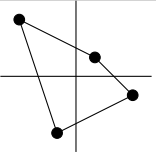
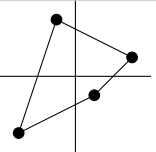
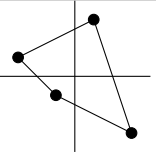
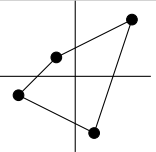
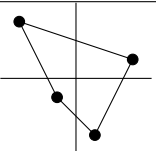
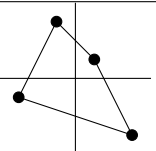
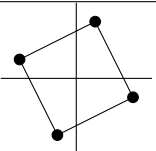
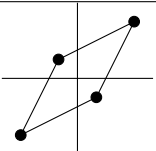
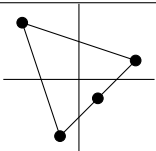
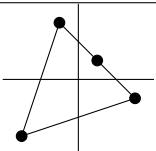
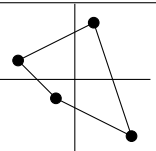
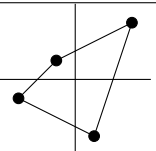
| family |   |   |   |   | a   |    | b   |    | c  |    | d   |    |
|--------|---|---|---|---|---|----|---|----|--|----|---|----|
| 1      | a | b | c | d |    | r1 |    | s1 |    | s2 |    | r2 |
|        | c | d | a | b |   |    |   |    |  |    |   |    |
|        | b | a | d | c |   |    |   |    |  |    |   |    |
|        | d | c | b | a |   |    |   |    |  |    |   |    |
| 2      | a | b | c | d |    | k1 |    | s1 |    | k2 |    | r2 |
|        | c | d | a | b |   |    |   |    |  |    |   |    |
|        | b | c | d | a |   |    |   |    |  |    |   |    |
|        | d | a | b | c |   |    |   |    |  |    |   |    |
| 3      | a | b | c | d |    | r1 |    | k3 |    | s2 |    | k4 |
|        | c | d | a | b |   |    |   |    |  |    |   |    |
|        | d | a | b | c |   |    |   |    |  |    |   |    |
|        | b | c | d | a |   |    |   |    |  |    |   |    |
| 4      | a | b | c | d |   | k1 |   | k3 |   | k2 |   | k4 |
|        | c | d | a | b |   |    |   |    |  |    |   |    |
|        | d | c | b | a |   |    |   |    |  |    |   |    |
|        | b | a | d | c |   |    |   |    |  |    |   |    |
| 5      | a | b | c | d |  | k5 |  | k6 |  | s2 |  | r2 |
|        | c | d | b | a |   |    |   |    |  |    |   |    |
|        | b | a | d | c |   |    |   |    |  |    |   |    |
|        | d | c | a | b |   |    |   |    |  |    |   |    |
| 6      | a | b | c | d |  | t1 |  | t2 |  | k2 |  | k4 |
|        | c | d | b | a |   |    |   |    |  |    |   |    |
|        | d | c | a | b |   |    |   |    |  |    |   |    |
|        | b | a | d | c |   |    |   |    |  |    |   |    |

Table 1: Decomposition of families 1

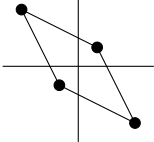
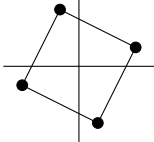
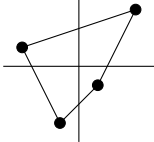
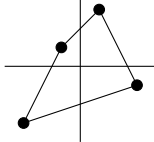
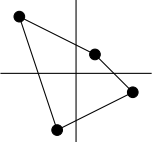
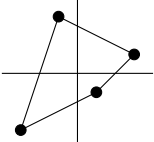
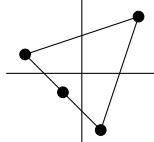
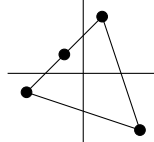
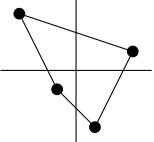
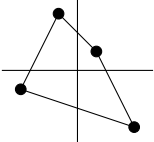
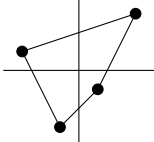
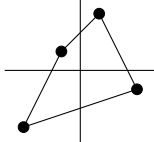
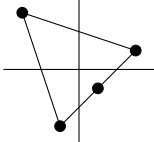
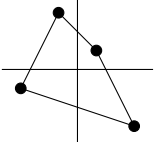
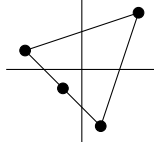
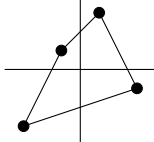
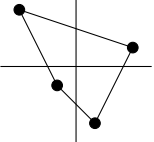
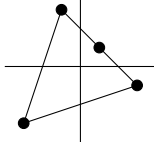
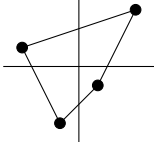
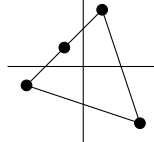
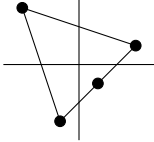
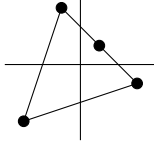
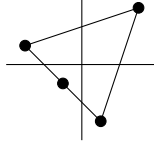
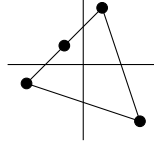
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
| <div>7</div> <table><tr><td>a</td><td>b</td><td>d</td><td>c</td></tr><tr><td>c</td><td>d</td><td>a</td><td>b</td></tr><tr><td>b</td><td>a</td><td>c</td><td>d</td></tr><tr><td>d</td><td>c</td><td>b</td><td>a</td></tr></table>  | a | b | d | c | c | d | a | b | b | a | c | d | d | c | b | a | <div></div> <div>r1</div> <div></div> <div>s1</div> <div></div> <div>k7</div> <div></div> <div>k8</div>         |
| a   | b | d | c |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| c   | d | a | b |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| b   | a | c | d |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| d   | c | b | a |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| <div>8</div> <table><tr><td>a</td><td>b</td><td>d</td><td>c</td></tr><tr><td>c</td><td>d</td><td>a</td><td>b</td></tr><tr><td>d</td><td>c</td><td>b</td><td>a</td></tr><tr><td>b</td><td>a</td><td>c</td><td>d</td></tr></table>  | a | b | d | c | c | d | a | b | d | c | b | a | b | a | c | d | <div></div> <div>k1</div> <div></div> <div>k3</div> <div></div> <div>t3</div> <div></div> <div>t4</div>         |
| a   | b | d | c |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| c   | d | a | b |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| d   | c | b | a |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| b   | a | c | d |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| <div>9</div> <table><tr><td>a</td><td>b</td><td>d</td><td>c</td></tr><tr><td>c</td><td>d</td><td>b</td><td>a</td></tr><tr><td>b</td><td>a</td><td>c</td><td>d</td></tr><tr><td>d</td><td>c</td><td>a</td><td>b</td></tr></table>  | a | b | d | c | c | d | b | a | b | a | c | d | d | c | a | b | <div></div> <div>k5</div> <div></div> <div>k6</div> <div></div> <div>k7</div> <div></div> <div>k8</div>     |
| a   | b | d | c |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| c   | d | b | a |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| b   | a | c | d |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| d   | c | a | b |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| <div>10</div> <table><tr><td>a</td><td>b</td><td>d</td><td>c</td></tr><tr><td>c</td><td>d</td><td>b</td><td>a</td></tr><tr><td>b</td><td>c</td><td>a</td><td>d</td></tr><tr><td>d</td><td>a</td><td>c</td><td>b</td></tr></table> | a | b | d | c | c | d | b | a | b | c | a | d | d | a | c | b | <div></div> <div>t1</div> <div></div> <div>k6</div> <div></div> <div>t3</div> <div></div> <div>k8</div> |
| a   | b | d | c |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| c   | d | b | a |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| b   | c | a | d |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| d   | a | c | b |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| <div>11</div> <table><tr><td>a</td><td>b</td><td>d</td><td>c</td></tr><tr><td>c</td><td>d</td><td>b</td><td>a</td></tr><tr><td>d</td><td>a</td><td>c</td><td>b</td></tr><tr><td>b</td><td>c</td><td>a</td><td>d</td></tr></table> | a | b | d | c | c | d | b | a | d | a | c | b | b | c | a | d | <div></div> <div>k5</div> <div></div> <div>t2</div> <div></div> <div>k7</div> <div></div> <div>t4</div> |
| a   | b | d | c |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| c   | d | b | a |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| d   | a | c | b |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| b   | c | a | d |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| <div>12</div> <table><tr><td>a</td><td>b</td><td>d</td><td>c</td></tr><tr><td>c</td><td>d</td><td>b</td><td>a</td></tr><tr><td>d</td><td>c</td><td>a</td><td>b</td></tr><tr><td>b</td><td>a</td><td>c</td><td>d</td></tr></table> | a | b | d | c | c | d | b | a | d | c | a | b | b | a | c | d | <div></div> <div>t1</div> <div></div> <div>t2</div> <div></div> <div>t3</div> <div></div> <div>t4</div> |
| a   | b | d | c |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| c   | d | b | a |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| d   | c | a | b |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
| b   | a | c | d |   |   |   |   |   |   |   |   |   |   |   |   |   |  |

Table 2: Decomposition of families 2