Journal of Information Processing Vol.20 No.3 680–685 (July 2012)

[DOI: 10.2197/ipsjjip.20.680]

Regular Paper

Squaring the Square with Integer Linear Programming

Sascha Kurz[1,a)]

Received: July 28, 2011, Accepted: March 2, 2012

Abstract: We consider so-called "squaring the square" puzzles where a given square (or rectangle) should be dissected into smaller squares. For a specific instance of such problems we demonstrate that a mathematically rigorous solution can be quite involved. As an alternative to exhaustive enumeration using tailored algorithms we describe the general approach of formulating the problem as an integer linear program.

Keywords: tiling problems, squaring the square, electrical circuits, integer linear programming, modeling

1. Introduction

Consider a floor tiler with the task of tiling a rectangular room using square tiles from a given set only. Questions arising are whether it is possible at all and if so to ask for minimum cost solutions given a target function such as the number of used tiles or the sum of the actual buying prices for each tile.

Even without this economic interpretation the underlying geometric idea of dissecting a rectangle into smaller squares was the source for many classical geometric puzzles such as the following "Problem 173" in Ref. [5]:

> "For Christmas, Mrs. Potipher Perkins received a very pretty patchwork quilt constructed of 169 square pieces of silk material. The puzzle is to find the smallest number of square portions of which the quilt could be composed and show how they might be joined together. Or, to put it the reverse way, divide the quilt into as few square portions as possible by merely cutting the stitches."

More formally, the generalized version of this problem can be stated as follows: For a given integer n, determine the minimum number $s(n)$ of squares in a tiling of an $n \times n$-square using $i \times i$-squares with $1 \leq i \leq n-1$, i.e. using only other integral squares.

Provided by Martin Gardner this puzzle first appeared in a puzzle magazine edited by Sam Loyd in 1907 and later on in the famous "Sam Loyd's Cyclopedia of 5,000 Puzzles, Tricks, and Conundrums", see www.mathpuzzle.com/loyd for an on-line version. We give the minimal solution consisting of 11 squares in **Fig. 1** and remark that it is unique up to rotations and reflections. For an overview on the (scattered) literature concerning these questions we refer to "Problem C2" in Ref. [4].

Once you have found a solution using *few* squares, by a heuristic search or simply by trial and error, it is easy to verify the validity. Even in the case where the sizes of the squares are omitted one can easily recover them by solving a linear equation system. Using the variables from **Fig. 2** in our example this equation system is given by:

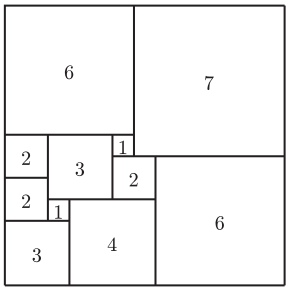

Fig. 1 Solution of the patch quilt puzzle.

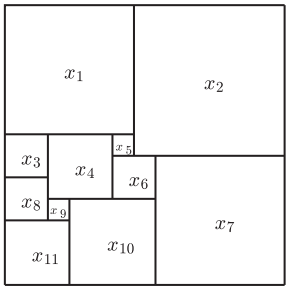

Fig. 2 Unknown sizes of the squares.

$$x_1 + x_2 = 13$$
$$x_3 + x_4 + x_5 + x_2 = 13$$
$$x_3 + x_4 + x_6 + x_7 = 13$$
$$x_8 + x_4 + x_6 + x_7 = 13$$
$$x_8 + x_9 + x_{10} + x_7 = 13$$
$$x_{11} + x_{10} + x_7 = 13$$
$$x_1 + x_3 + x_8 + x_{11} = 13$$
$$x_1 + x_4 + x_9 + x_{11} = 13$$
$$x_1 + x_4 + x_{10} = 13$$
$$x_1 + x_5 + x_6 + x_{10} = 13$$
$$x_2 + x_6 + x_{10} = 13$$
$$x_2 + x_7 = 13$$

As in most puzzles, asking for a minimal solution in some

[1] University of Bayreuth, Bayreuth, Germany
[a)] sascha.kurz@uni-bayreuth.de

sense, the most difficult part is verifying that a given solution is the minimum solution. This part is addressed by simply stating the smallest *known* solution. Rigorous mathematical proofs are explicitly given in only a few cases and even then are found only in personal communications, see e.g. mathworld.wolfram.com/MrsPerkinssQuilt.html.

If the puzzle can be formulated as a finite search space, one can in principle apply exhaustive enumeration. This is the case in the framework of the "squaring the square" context. However, this yields a drawback: sophisticated custom enumeration algorithms have to be developed and implemented in order to obtain results in a reasonable amount of time.

The purpose of this article is to demonstrate that it is not that hard to formulate such puzzles as integer linear programs. Standard software can be used in order to exactly solve these problems without the need for implementing new algorithms. The big advantage of such an approach is that it can be easily adapted to different variants of the puzzle, which we will demonstrate in the following.

1.1 Outline of the Paper

In Section 2 we give a puzzle from a mathematical competition for 14–16 year old pupils and outline a rigorous mathematical solution. The known results on the determination of the minimal number of squares $s(n)$ are outlined in Section 3. The underlying theory for these classes of puzzles is briefly addressed in Section 4. In contrast to this exhaustive enumeration approach we describe the modeling process as an integer linear program in Section 5 and end with a conclusion in Section 6.

2. A puzzle from a Mathematical Competition and a Rigorous Solution

In a mathematical team competition for 14–16 year pupils we have proposed the following task: Tile a 13×13 room using $i \times i$-squares where $i = 1, 2, \ldots, 12$.

(1) Determine the minimum number of tiles using at least one 12×12-square.
(2) Determine the minimum number of tiles using at least one 11×11-square.
(3) Determine the minimum number of tiles using at least one 10×10-square.

According to comments from the participants, finding the answers 26, 16, and 13 is not very difficult. For the first two questions possible tilings achieving these numbers are drawn in **Figs. 3** and **4**. Proving that there are no solutions using fewer squares turned out to be a much harder task for the participants.

To demonstrate the arising difficulties in proving the minimality of the stated tilings, we outline rigorous proofs for the first two cases and leave the third case to the interested reader. We assume that all squares are arranged on an integer grid.

2.1 Case 1.

Using a 12×12-square inside a 13×13-square leaves only the possibility to fill the remaining gaps with 1×1 squares. Since $13^2 - 12^2 = 25$ at least $1 + 25 = 26$ tiles are required.

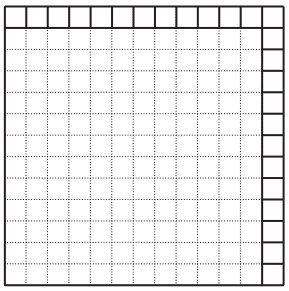

Fig. 3 Optimal solution using a 12×12-square.

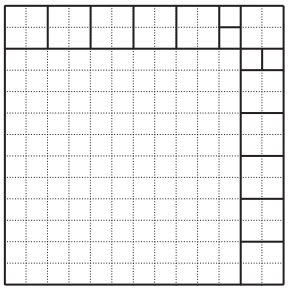

Fig. 4 Optimal solution using a 11×11-square.

2.2 Case 2.

If no corner of the inner 11×11-square coincides with a corner of the outer 13×13-square, then at least $13^2 - 11^2 = 48$ additional squares are required. Thus we can assume without loss of generality that the lower left corners coincide, as in Fig. 4. As the largest possible side length of the additional squares is 2 we can deduce that at least $\left\lceil \frac{13^2-11^2}{2^2} \right\rceil = 12$ of them are needed. Since the upper 2×13-strip can not be covered using non-overlapping 2×2-squares only the number n_2 of used 2×2-squares is at most 11 and thus the number n_1 of used 1×1-squares is given by $n_1 = 13^2 - 11^2 - 4n_2 \geq 4$. Thus we need at least

$$1 + n_2 + n_1 = 49 - 3n_2 \geq 49 - 3 \cdot 11 = 16$$

squares in total.

These ad hoc proofs can be replaced by using a slightly modified version of the integer linear program presented in Section 5. We remark that our proof for the third question is already twice as long as the one stated for case 2.

3. Squaring the Square Using as Few Squares as Possible

Our benchmark example for this article asks for the minimum number $s(n)$ of squares needed to tile an $n \times n$-square using $i \times i$-squares with $1 \leq i \leq n - 1$ only.

Since we can enlarge a given tiling of an $n \times n$-square we obviously have $s(n_1 \cdot n_2) \leq \min(s(n_1), s(n_2))$ for all $n_1, n_2 \in \mathbb{N}_{\geq 2}$. Thus it is very likely that we need to determine $s(n)$ for primes only to obtain the minimum values.

In **Table 1** we list the minimum values $s(p)$ and counts of the used squares for the first primes. These values were computed by solving the integer linear programing formulation from Section 5 with the `Gurobi` solver. Exhaustive enumerations of several kinds of squared squares are known up to $n = 29$, see e.g., www.squaring.net and several papers by A.J.W. Duijvestijn such as Ref. [7]. Heuristically found configurations, meeting our exact

Table 1 Minimal numbers $s(p)$ to tile a $p \times p$-square.

p	s(p)	used squares
2	4	1^4
3	6	$1^5 2^1$
5	8	$1^4 2^3 3^1$
7	9	$1^3 2^3 3^2 4^1$
11	11	$1^4 2^1 3^3 5^2 6^1$
13	11	$1^2 2^3 3^2 4^1 6^2 7^1$
17	12	$1^2 2^3 3^1 4^2 5^1 8^2 9^1$
19	13	$1^1 3^4 4^2 5^3 6^1 9^1 10^1$
23	13	$1^2 2^3 3^1 4^1 6^2 7^1 10^2 13^1$
29	14	$1^1 3^2 4^2 5^3 6^2 7^1 13^2 16^1$
31	15	$1^3 2^3 4^3 8^3 15^2 16^1$
37	15	$2^2 3^3 4^3 5^1 9^2 11^1 17^2 20^1$
41	15	$1^2 2^2 3^2 4^1 5^1 7^1 11^2 12^1 18^2 23^1$
43	16	$1^2 4^4 5^1 6^2 7^2 11^1 13^1 19^2 24^1$
47	16	$1^1 3^3 6^1 7^2 9^3 9^2 10^1 22^2 25^1$
53	16	$1^1 3^2 4^2 5^2 6^2 7^1 13^2 16^1 24^2 29^1$
59	17	$1^2 2^1 4^2 6^1 8^1 10^2 11^1 12^1 14^1 16^1 18^1 19^1 29^1 30^1$
61	17	$3^2 4^4 6^2 8^1 9^2 11^1 15^1 17^1 29^2 32^1$

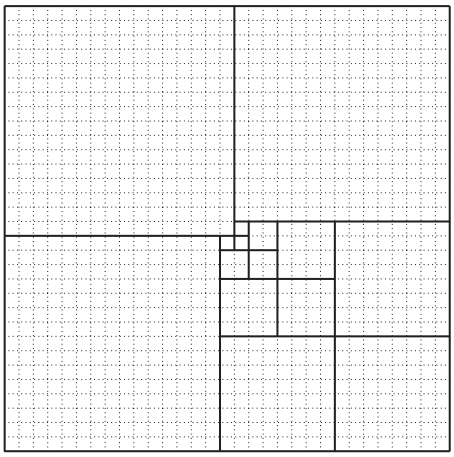

Fig. 5 An optimal tiling for a 31×31-square.

bounds, can be found in many places.

The encoding $a_1^{b_1} a_2^{b_2} \dots a_r^{b_r}$, in Table 1, means that exactly b_i squares of type $a_i \times a_i$ are used.

Looking at $s(p)$ for primes of the form $p = 2^r - 1$ reveals an interesting pattern: $1^3 2^3 4^3 \cdots \left(2^i\right)^3 \cdots \left(2^{r-2}\right)^3 \left(2^{r-1}-1\right)^2 \left(2^{r-1}\right)^1$. In **Fig. 5** we have depicted the construction for $p = 2^5 - 1 = 31$ which can be easily generalized, so that we have

$$s(2^r - 1) \le 3r$$

for all $r \ge 2$.

Trustrum [12] gave a set of general constructions showing

$$s(n) \le 6 \log_2(3n - 1) - 10 < 6 \log_2(n).$$

For the other direction Conway [3] has proven

$$s(p) \ge \log_2(p)$$

for primes p.

4. Squared Rectangles from Electrical Networks

A correspondence between a certain class of planar electrical networks and squared rectangles was observed by Brooks, Smith, Stone and Tutte [2]. Here we give a brief sketch of the approach and refer the interested reader to the expositions Refs. [7], [9] and the historical review Ref. [8]. Extensive information on the topic can also be found at www.squaring.net.

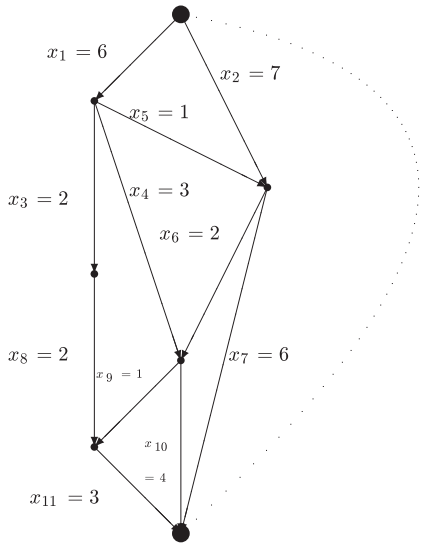

Fig. 6 Network corresponding to Fig. 1.

Given a dissection $\mathcal{D}$ of a rectangle into squares we can build up a network $\mathcal{G}$ as follows. Dropping the vertical lines of the constituent squares leaves unions of horizontal lines at the same height, which we call (horizontal) dissectors. The upper and the lower side of the outer rectangle are examples of such dissectors. As vertices of $\mathcal{G}$ we choose the horizontal dissectors of the squares and as edges the squares itself. Two vertices are joined by an edge if the corresponding two horizontal dissectors contain the lower and upper horizontal side of the corresponding square. We call the vertex corresponding to the upper side of the outer rectangle the positive pole and the vertex corresponding to the lower side the negative pole of the network.

The network corresponding to the squared 13×13-square in Fig. 1 is drawn in **Fig. 6**.

Given a dissection of a rectangle into squares such a network is uniquely defined (if the graph is drawn in a certain way) and it can be shown to be planar. Depending on the properties of the dissection more graph theoretic restrictions can be deduced. E.g., if all squares have different sizes the graph $\mathcal{G}$ complemented by the edge connecting the positive and the negative pole is a three-connected planar graph with out multiple edges. If squares of the same size are allowed, the augmented network remains at least two-connected but may contain multiple edges.

The base for an exhaustive enumeration algorithm for squared rectangles is that from each planar network we can compute the side lengths of the squares using Kirchhoff's rules. The first law states that the sum of currents in a network meeting at a point is zero. In our example of Fig. 6 this yields:

$$x_1 = x_3 + x_4 + x_5$$
$$x_3 = x_8$$
$$x_2 + x_5 = x_6 + x_7$$
$$x_4 + x_6 = x_9 + x_{10}$$
$$x_8 + x_9 = x_{11}$$

The second law states that the directed sum of the electrical poten-

tial differences in any sub-circuit is zero. Assuming unit resistors this gives:

$$x_1 + x_5 - x_2 = 0$$
$$x_5 + x_6 - x_4 = 0$$
$$x_6 + x_{10} - x_7 = 0$$
$$x_9 + x_{11} - x_{10} = 0$$
$$x_3 + x_8 - x_9 - x_4 = 0$$

In general it can be proven that the solution space of the combined equation system is one-dimensional so that one can choose the unique minimal integer solution. For our example we obtain, of course, multiples of the solution given in Fig. 6.

So by exhaustively generating the planar graphs and determining the corresponding dissection of a rectangle, one can systematically explore the search space for squared rectangles. This approach is limited to rectangles with a small number of squares. Unless one can exploit strong restrictions on the graph parameters enumerations for side lengths n with $s(n) \geq 40$ seem to be out of reach, see Ref. [1]. As already mentioned in Section 3, the author is not aware of any exhaustive enumerations of squared squares with more than 29 squares. A relatively early work using computers is Ref. [6].

5. An Integer Linear Programming Formulation

In this section we want to demonstrate that one can develop an integer linear programming formulation for the "squaring the square" puzzle quite naturally. Once we have such a formulation at hand we can apply standard software tools to compute the solution. Only minor changes are necessary to adapt the model to variants of the problem.

To figure out how to model a problem it is useful to ask some basic questions: What can we decide? In our context the answer is easy – the positions of the squares. How can we represent or encode our decisions? Drawing a geometric figure may be suitable for explanations to humans, but when talking to a computer we need something different. We may represent the chosen tiling of Fig. 1 by a table:

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

Using this representation the task is to write integers from 1 to $n - 1$ into the cells of a $n \times n$-grid forming squares for $n = 13$. If we have decided where to place the upper left corner of a 7×7-square, then the remaining 48 cell entries follow directly. So we restrict ourselves to printing the positions of the upper left corners:

6						7						
2		3			1							
					2		6					
2												
		1	4									
3												

In many cases binary decision variables are well suited to represent decisions. Therefore we introduce the binary variables $x_{i,j,h}$ having value 1 if and only if we write integer h in cell (i, j).

How do we evaluate different decisions? Here our criterion is the number of used squares which can be counted by

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{h=1}^{n-1} x_{i,j,h}.$$

What are the constraints restricting our decisions? Here we have to guarantee that each cell of the $n \times n$-grid is covered exactly by one square and that the squares completely lie inside. The first condition can be written as

$$\sum_{h=1}^{n-1} \sum_{a=0}^{\min(i-1,h-1)} \sum_{b=0}^{\min(j-1,h-1)} x_{i-a,j-b,h} = 1$$

for all $1 \leq i, j \leq n$. Understanding this constraint is facilitated by asking the following question: Under what condition is a $h \times h$-square with left upper corner at position $(i - a, j - b)$ covering cell (i, j)? The second condition can be written as $x_{i,j,h} = 0$ $\forall 1 \leq i, j \leq n$: $i + h > n + 1 \vee j + h > n + 1$.

Using a modeling language such as `ZIMPL`, see `zimpl.zib.de`, one can write this in a very compact and readable way:

```
param n:=13;
set A:={1 to n};
set B:={1 to n-1};
set S:=A cross A cross B;
var x[S] binary;
minimize target:
sum <i,j,h> in S: x[i,j,h];
subto packing:
forall <i,j> in A cross A:
sum <h> in B:
sum <a> in {0 to min(i-1,h-1)}:
sum <b> in {0 to min(j-1,h-1)}:
x[i-a,j-b,h]==1;
subto boundary:
forall <i,j,h> in S with
i+h>n+1 or j+h>n+1: x[i,j,h]==0;
```

`ZIMPL` produces an input file for an ILP solver like `CPLEX` or `Gurobi`. In general one can often quite rapidly develop a first working integer linear programming model using `ZIMPL`, see Ref. [10].

In **Table 2** we have listed the results for small values of n including the running time and the number of branch&bound nodes.

Table 2 Results and running times using `Gurobi`.

n	s(n)	seconds	b&b-nodes
11	11	0.1	39
13	11	0.3	41
17	12	2	92
19	13	9	168
23	13	19	173
29	14	930	3,341
30	4	2	1
31	15	3,148	7,409
32	4	3	1
33	6	4	1
34	4	4	1
35	8	20	10
36	4	5	1
37	15	12,634	6,911
38	4	9	1
39	6	8	1
40	4	9	1
41	15	26,887	6,520

We have omitted all cases with $n \leq 41$, where the problem was solved in the root node, i.e., where we have exactly 1 b&b-node, and the running time was less than one second. So without much effort or theoretic insight, it was possible to exactly determine the minimum number $s(n)$ of squares to tile a $n \times n$-square, compare sequence A018835 in the on-line encyclopedia of integer sequences at `oeis.org`.

We remark that by using this rather simple approach we could verify the values of Table 1 and that $s(n) = \min(s(p) \mid p|n, p \geq 2)$ holds for all $n \leq 104$.

5.1 Problem Variations

Taking the previous integer linear programming formulation as a basic module we can formulate models for variations of the "squaring the square" theme. By introducing the auxiliary variables y_i, counting the number of used $i \times i$-squares, we can express many constraints quite compactly. The relation between the x- and the y-variables can be stated as

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{i,j,h} = y_h$$

for all $1 \leq h \leq n-1$.

The three problems from Section 2 can in this way be solved by requiring $y_{12} = 1$, $y_{11} = 1$, or $y_{10} = 1$, respectively.

Since we have observed that the known minimal values $s(n)$ of our benchmark problem from Section 3 arise from values $s(p)$ for primes p dividing n, the additional requirement that the greatest common divisor of the side lengths has to be one was introduced. This can be reformulated such that for each prime $2 \leq p \leq n-1$ the side length of the squares are not all divisible by p: $\sum_{1 \leq h \leq n-1 \,:\, p \nmid h} y_h \geq 1$.

6. Conclusion

In this article we have considered a special and well known class of geometric puzzles, where rectangles have to be dissected into smaller squares. Arguing that a discovered solution is minimal in some sense can be quite involved as exemplarily demonstrated in Section 2. On the other hand performing a computer based exhaustive search can be time consuming and often requires the development of a customized algorithm and some theoretical insights in the specific problem. We have briefly outlined the well-known theory for the problem in Section 4. Based on exhaustive generation using planar graphs going beyond 40 vertices, which correspond to used squares, seems to be computationally infeasible. But we have to admit that we are not aware of any attempts to determine the exact values of $s(n)$ with some kind of restricted generation, i.e., where the search tree is pruned if one can anticipate that the achievable side lengths n will be too large.

To obtain rigorous results quickly, modeling the problem as an integer linear program and afterwards solving it with standard software seems to be a viable approach. As demonstrated in Section 5, the modeling process is more or less straightforward and can be adjusted to different problem variations relatively easily. Of course such a general approach has its computational limits but the same holds for enumeration algorithms too (whenever the search space grows exponentially, as it does in our context).

Without sophisticated methods, such as column generation, the integer linear programming approach is currently not able to push the actual computational frontiers, but seems to be an accessible method for a broader audience of puzzlers. Its great benefit is its simplicity, compared to the more involved direct method in Section 4, and its very general applicability.

Another example where an integer linear programming formulation is used to quickly solve mathematical puzzles is given in Ref. [11].

Acknowledgments The authors thanks the anonymous referees and the handling editor for their careful reading of a preliminary version of this paper. Their constructive remarks were extremely useful to improve its presentation.

References

[1] Brinkmann, G. and McKay, B.: Fast generation of planar graphs, *MATCH Commun. Math. Comput. Chem.*, Vol.58, No.2, pp.323–357 (2007).
[2] Brooks, R., Smith, C., Stone, A. and Tutte, W.: The dissection of rectangles into squares, *Duke Math. J.*, Vol.7, pp.312–340 (1940).
[3] Conway, J.: Mrs Perkin's quilt, *Proc. Camb. Phil. Soc.*, Vol.60, pp.363–368 (1964).
[4] Croft, H., Falconer, K. and Guy, R.: *Unsolved problems in geometry*, Problem Books in Mathematics, Springer-Verlag, New York (1994).
[5] Dudeney, H.: *Amusements in Mathematics*, Nelson, New York (1917).
[6] Duijvestijn, A.: Electronic computation of squared rectangles, Ph.D. Thesis, Eindhoven, p.92 (1962).
[7] Duijvestijn, A., Federico, P. and Leeuw, P.: Compound perfect squares, *Am. Math. Mon.*, Vol.89, pp.15–32 (1982).
[8] Federico, P.: Squaring rectangles and squares: A historical review with annotated bibliography, Graph theory and related topics, pp.173–196 Academic Press (1979).
[9] Kazarinoff, N. and Weitzenkamp, R.: Squaring rectangles and squares, *Am. Math. Mon.*, Vol.80, pp.877–888 (1973).
[10] Koch, T.: Rapid Mathematical Programming, Ph.D. Thesis, Technische Universität Berlin (2004).
[11] Koch, T.: Rapid Mathematical Programming or How to Solve Sudoku Puzzles in a Few Seconds, *OR*, pp.21–26 (2005).
[12] Trustrum, G.: Mrs Perkin's quilt, *Proc. Camb. Phil. Soc.*, Vol.61, No.7, pp.7–11 (1965).

Sascha Kurz was born in 1980. He received his Ph.D. from the University of Bayreuth in 2005. His current research interests are combinatorial optimization and exhaustive generation of all kinds of discrete structures like point sets with pairwise integral distances, unit-distance graphs, codes or voting systems.