Pesquisa Operacional (2022) 42: e262876 p.1-13
doi: 10.1590/0101-7438.2022.042.00262876

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
ARTICLES

ALMOST SQUARING THE SQUARE: OPTIMAL PACKINGS FOR NON-DECOMPOSABLE SQUARES

Vitor Pimenta dos Reis Arruda[1*],
Luiz Gustavo Bizarro Mirisola[2] and Nei Yoshihiro Soma[3]

Received April 7, 2022 / Accepted July 14, 2022

ABSTRACT. We consider the problem of finding the minimum uncovered area (trim loss) when tiling non-overlapping distinct integer-sided squares in an $N \times N$ square container such that the squares are placed with their edges parallel to those of the container. We find such trim losses and associated optimal packings for all container sizes N from 1 to 101, through an independently developed adaptation of Ian Gambini's enumerative algorithm. The results were published as a new sequence to The On-Line Encyclopedia of Integer Sequences®. These are the first known results for optimal packings in non-decomposable squares.

Keywords: cutting and packing, enumeration, operational research.

1 INTRODUCTION

This article addresses a problem we call 2OKPN (2-dimensional Orthogonal Knapsack Problem in an $N \times N$ square), defined by:

Given an $N \times N$ square container and the set of $N-1$ square items sized $\{1 \times 1,\ 2 \times 2,\ \cdots,\ N-1 \times N-1\}$, find a packing of a subset of the items with the minimum uncovered area (trim loss), obeying:

1. All packed items must be placed entirely inside the container;
2. no two packed items may overlap;
3. all packed items must be placed with their edges parallel to those of the container.

*Corresponding author
[1]Instituto Tecnológico de Aeronáutica 12228-900, São José dos Campos-SP, Brazil – E-mail: vitor.arruda@ga.ita.br – http://orcid.org/0000-0002-4143-8346
[2]Instituto Tecnológico de Aeronáutica 12228-900, São José dos Campos-SP, Brazil – E-mail: lgm@ita.br – http://orcid.org/0000-0002-2689-452X
[3]Instituto Tecnológico de Aeronáutica 12228-900, São José dos Campos-SP, Brazil – E-mail: soma@ita.br – http://orcid.org/0000-0003-3069-9644

2OKPN can be considered the meeting point of two related problems: the OR (Operational Research) researcher's 2-dimensional Orthogonal Knapsack Problem (2OKP) and the mathematician's Squaring the Square (STS).

2OKP (among other variations) searches for a packing of rectangular items into a rectangular container to minimize the trim loss. Therefore it is a generalization of 2OKPN to rectangles of arbitrary width and height. Although exact solution methods have been proposed — see Iori et al. (2021) for a survey — the fact that the problem is *NP-hard* motivates research on heuristics, which have less computational cost but usually find only "good" suboptimal solutions. See Wei et al. (2009), Wei et al. (2011), Leung et al. (2012) and Shiangjen et al. (2018).

STS (among other variations) looks for perfect squared squares: squares that can be tiled by distinct smaller squares with no trim loss. The classical solution approach models the packing as an electrical network, each square corresponding to a wire (Figure 1). The wire has unit resistance, and the current that flows through is numerically equal to the side length of the square. To find perfect squared squares, a graph algorithm enumerates these networks and converts them into the perfect packings they represent (Brooks et al. (1940)). Whenever this enumeration finds a perfect squared square, it corresponds to the optimal solution of some 2OKPN instance, for example Figure 1 is the optimal 2OKPN solution for $N = 110$. But since Brooks' method can only find perfectly decomposable squares (zero trim loss), it cannot be used to solve 2OKPN in general. Although Biró & Boros (1984) later generalized these networks to handle the non-perfect case, we have not attempted to adapt these methods to solve 2OKPN.

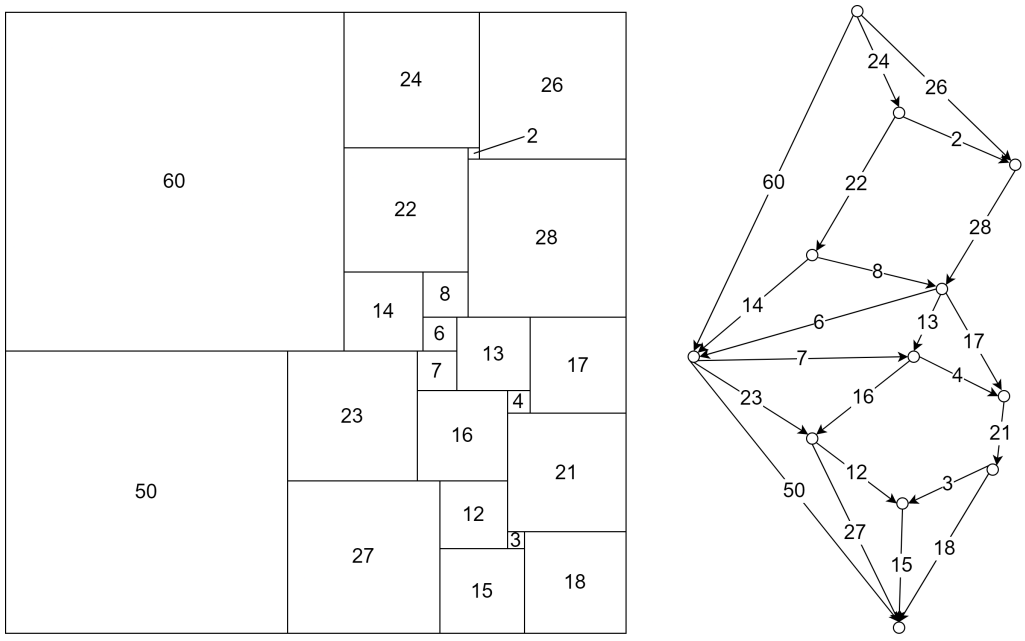

Figure 1 – Example electrical network corresponding to a perfect squared square.

2 ENUMERATIVE METHOD

Working on STS, Gambini (1999) proved that 110 is the minimum side length of a square that admits perfect decomposition. Questions like the one he answered ("what is the smallest...") generate renewed interest in STS, even though much is already known about the problem. In fact, YouTube channel Numberphile created a video featuring STS and some "what is the smallest" questions. By now, the number of views almost reaches 1 million.

Gambini's result was derived from a search algorithm henceforth called g. Given an $N \times N$ square container, g searches for a perfect decomposition by considering all possible placements of distinct smaller squares on *delimited plates* of the container. A *plate* is a line segment formed by the top edges of already placed squares (Figure 2). It is *delimited* if the plates to its left and right are both higher than it (a container wall is always considered higher). For example, in Figure 2, the leftmost and rightmost plates are the only delimited ones.

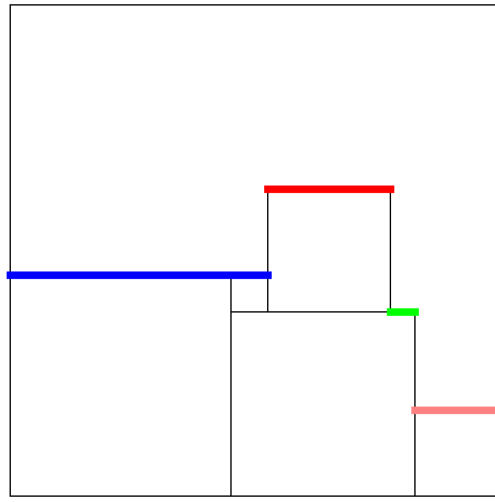

Figure 2 – Example of plates in a square container.

Gambini argues that a perfect packing (zero trim loss) can always be achieved by placing each square at the leftmost end of the smallest-width delimited plate (henceforth SWD plate). This inspires a backtracking algorithm for finding perfect packings, where each square is considered for placement at the SWD plate. The backtracking step occurs when the width of the next SWD plate is smaller than the side of any unplaced square. Figure 3 illustrates part of the execution of g for a 5×5 square container.

The original purpose of g is to find perfect dissections with zero trim loss. Therefore, for non-decomposable N-sided containers, the algorithm terminates without finding a solution. Further, in the context of 2OKPN, there is no guarantee that g will find the packing with minimum trim loss, since the backtracking step may cut away the path to such a packing.

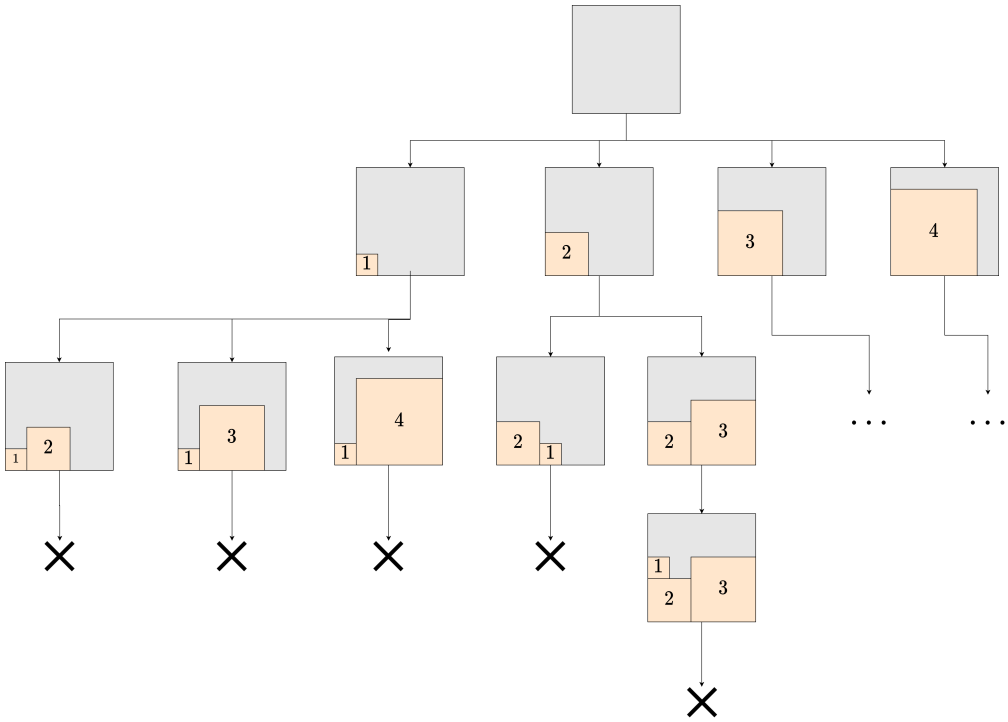

Figure 3 – Partial execution sequence of g for a 5×5 square container. The "X" means that no square fits on the current SWD, therefore backtracking occurs.

Even so, we have independently devised an adaptation to the algorithm that allowed us to solve 2OKPN. It consists of modifying the set of available squares by including extra units of the 1×1 square. For example, suppose we know $N = 100$ (which is non-decomposable) admits a packing with 1 unit trim loss. Then, by adding an extra 1×1 square to the set of available squares used by g, the algorithm will find the solution (using both 1×1 units). Removing the excess square, we are left with the packing shown in Figure 4. Hereafter we write $g(N, e_1)$ to denote the packing obtained by g on an $N \times N$ container when g is supplied e_1 extra units of the 1×1 square. Notice that the actual packing found by g may contain less than e_1 extra units.

We initially thought the idea was novel (although simple), but later found Lesh et al. (2004) had mentioned it as a suggestion (but not implemented it).

Since our objective was to solve every 2OKPN from $N = 1$ to $N = 101$, we created a method to make use of information obtained from previous solutions. We call it *SUB*, for *speculative upper bound*. To explain it, consider $e_1^{(N)}$ the minimum trim loss possible on an $N \times N$ square container, a quantity only dependent on N, and that is unknown *a priori*. When g solves an N-sided square, we take note of $e_1^{(N)}$. Later, when solving for $N+1$, we take $e_1^{(N)}$ as a speculative upper bound $\overline{e_1}^{(N+1)}$ for $e_1^{(N+1)}$. Also, we define the lower bound $\underline{e_1}^{(N+1)} = 0$ (we know by Gambini's proof that 1 is a lower bound, but we chose not to rely on this result and instead rediscover it). What

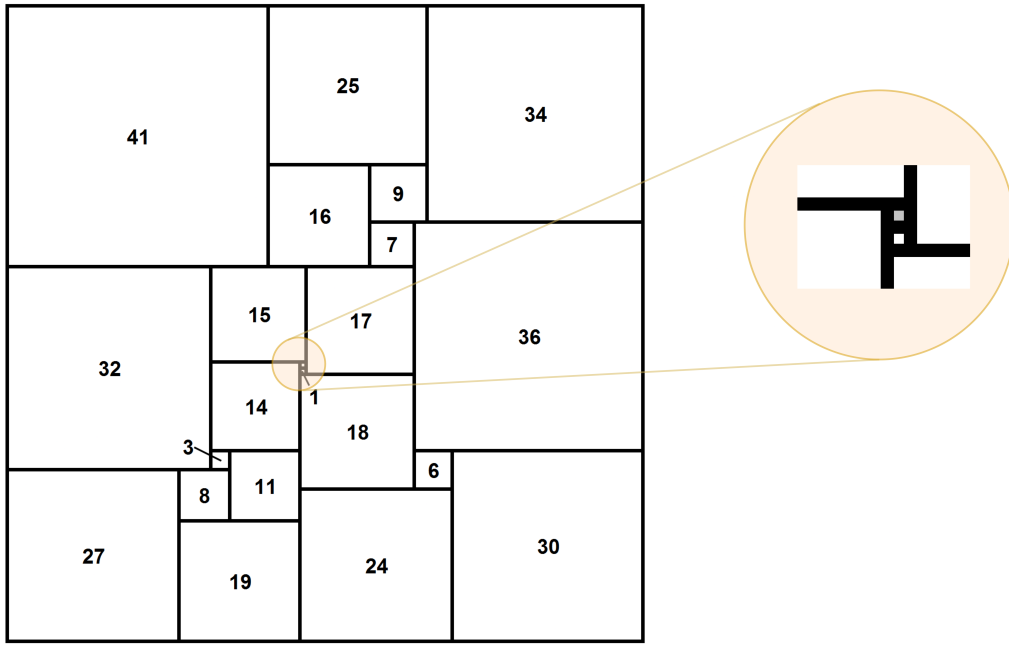

Figure 4 – Minimum trim loss packing for an $N = 100$ square container. In the detail, the trim loss is indicated by a 1×1 grey square.

follows is a search for the exact value of $e_1^{(N+1)}$ which resembles a binary search, and we describe by the procedure:

1. (Initialization): Assign $\underline{e_1}^{(N+1)} \leftarrow 0$ and $\overline{e_1}^{(N+1)} \leftarrow e_1^{(N)}$.
2. (Validate speculative upper bound): Execute $g\left(N+1, \overline{e_1}^{(N+1)}\right)$. If no solution is found, assign $\underline{e_1}^{(N+1)} \leftarrow \overline{e_1}^{(N+1)} + 1$ and $\overline{e_1}^{(N+1)} \leftarrow \max\left(2\overline{e_1}^{(N+1)}, 1\right)$, then repeat step 2. Otherwise, if g found a solution that uses e_g extra units of the 1×1 square, assign $\overline{e_1}^{(N+1)} \leftarrow e_g$ and go to step 3.
3. (Binary search): If $\underline{e_1}^{(N+1)} \geq \overline{e_1}^{(N+1)}$, go to step 4. Otherwise, execute

$$g\left(N+1, \left\lfloor \left(\underline{e_1}^{(N+1)} + \overline{e_1}^{(N+1)}\right)/2 \right\rfloor\right)$$

   If no solution is found, assign $\underline{e_1}^{(N+1)} \leftarrow \overline{e_1}^{(N+1)} + 1$. Otherwise, if g found a solution that uses e_g extra units of the 1×1 square, assign $\overline{e_1}^{(N+1)} \leftarrow e_g$. In any case, repeat step 3.
4. (Solution found): The minimum trim loss possible is $e_1^{(N+1)} = \overline{e_1}^{(N+1)}$.

Step 1 is necessary because the initial value of $\overline{e_1}^{(N+1)}$ (which is $e_1^{(N)}$) may not be a valid upper bound for $e_1^{(N+1)}$. But when step 1 completes, this is no longer an issue, and the remaining steps implement a binary search attempting to execute g as few times as possible. Note that the "max" operator in step 2 is necessary because $\overline{e_1}^{(N+1)}$ may have been 0 before the reassignment.

3 PROBLEM REDUCTION

Given an instance of 2OKPN, we call *trivial* the solution where the $N-1 \times N-1$ and 1×1 squares are the only ones packed. The trivial solution has trim loss $N^2 - ((N-1)^2+1) = 2(N-1)$. As for other solutions, consider what happens when a square with side length $J > N/2$ is packed: after it, no square larger than $N-J$ can be packed. Therefore, a lower bound L_J on the achievable trim loss is:

$$L_J = \max\left(0, N^2 - \left(J^2 + \sum_{j=1}^{N-J} j^2\right)\right) \tag{1}$$

Consequently, the $J \times J$ square can only be present in a non-trivial optimal solution if:

$$L_J < 2(N-1) \tag{2}$$

$$\implies J^2 + \sum_{j=1}^{N-J} j^2 > (N-1)^2 + 1 \tag{3}$$

$$\implies -\frac{J^3}{3} + J^2 N + \frac{3J^2}{2} - JN^2 - JN - \frac{J}{6} + \frac{N^3}{3} - \frac{N^2}{2} + \frac{13N}{6} - 2 > 0 \tag{4}$$

If N is understood as a parameter, the left-hand side of the inequality in Equation 4 is a polynomial $p_N(J)$ on the variable J. According to the Cardano formula, it has three roots j_1, j_2 and j_3:

$$j_1 = N - \frac{\sqrt{25+96N} - 11}{4} \tag{5}$$

$$j_2 = N - 1 \tag{6}$$

$$j_3 = N + \frac{\sqrt{25+96N} + 11}{4} \tag{7}$$

Since $N \in \mathbb{N}^*$, all three roots are real numbers. If we further restrict ourselves to $N > 2$, then $j_1 < j_2 < N < j_3$. Also, the derivative of $p_N(J)$ at $J = j_1$ is always negative when $N > 2$ (we omit this calculation), therefore $p_N(J)$ is positive for all $J < j_1$ and negative for all $j_1 < J < j_2$.

With this, the solution to the inequality in Equation 2 where $0 < J < N-1$ and $N > 2$ is

$$J < N - \frac{\sqrt{25+96N} - 11}{4} \tag{8}$$

In other words, squares larger than the right-hand side of Equation 8 can be removed without compromising optimality. We reduce the 2OKPN instances in this way before executing the enumerative algorithm described in the previous section.

4 RESULTS

At the time of writing, we have solved all 2OKPN instances with $1 \leq N \leq 101$ and thus proposed a new sequence (A334905) to The On-Line Encyclopedia of Integer Sequences® (OEIS Foundation Inc. (2022)). The Nth term of the sequence is the trim loss of the solution to 2OKPN. The link https://oeis.org/A334905 is the entry for the sequence, and illustrations of the optimal packings obtained (Figure 4 is an example) can be viewed at https://oeis.org/A334905/a334905_3.pdf.

All results were obtained by a single-threaded program running on a computer with Inter(R) Core(TM) i5-2410M CPU @ 2.30GHz and 4GB physical memory. In what follows, we omit instances with $N < 15$ because they have trivial optimal solution. Figure 5 illustrates the total time taken (in hours) to solve each 2OKPN instance for $15 \leq N \leq 101$ and their optimal trim losses. For the full numerical data, see Table 1. Trim loss and execution time are correlated, in the sense that, in 99% of the cases where $e_1^{(N+1)} \neq e_1^{(N)}$, solving for $N+1$ takes more (resp. less) time than solving for N when $e_1^{(N+1)}$ is greater (resp. less) than $e_1^{(N)}$.

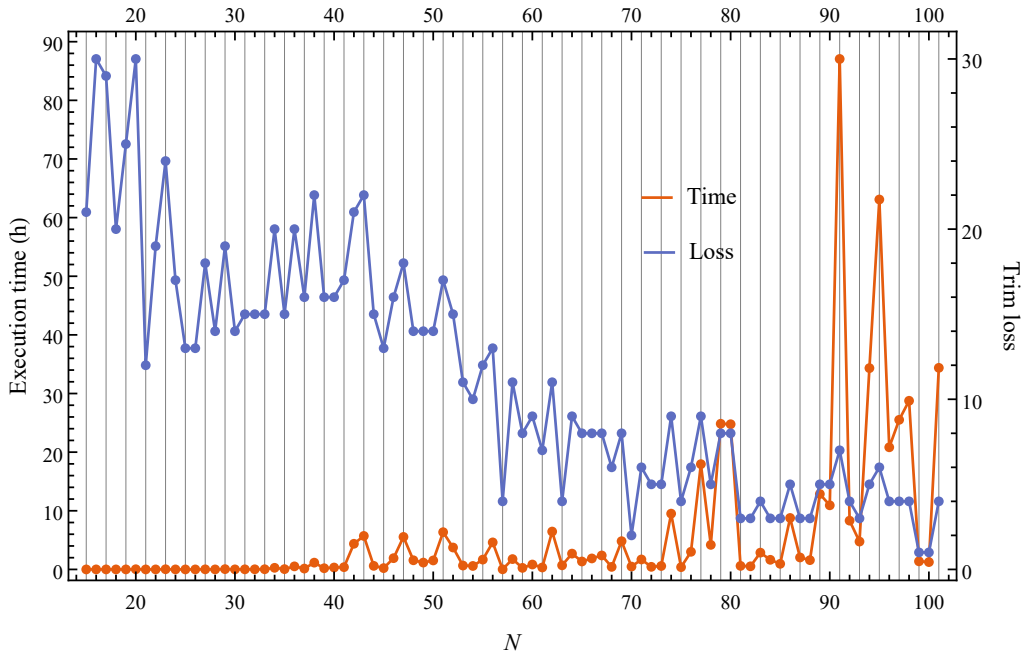

Figure 5 – Total solution time for 2OKPN instances.

Figures 6, 7 and 8 break down the duration of individual executions of g for each N (there are three separate figures for ease of presentation). Note that Figure 6 measures time in minutes, whilst Figures 7 and 8 measure time in hours. Before inspecting the empirical data, one may predict that the duplication of $\overline{e_1}^{(N+1)}$ in step 2 of SUB would likely lead to long search times because the search tree of $g(N+1, \overline{e_1}^{(N+1)})$ gets bigger as $\overline{e_1}^{(N+1)}$ increases. But this hypothesis is proved false by the aforementioned Figures, since the darkest-colored cells are almost always

Table 1 – Solution time of 2OKPN instances, in seconds.

N	Time (s)	N	Time (s)	N	Time (s)
15	1	44	2186	73	2096
16	15	45	802	74	34244
17	19	46	6915	75	1389
18	4	47	19951	76	10824
19	19	48	5618	77	64632
20	108	49	4199	78	15070
21	2	50	5511	79	89434
22	7	51	22867	80	89147
23	107	52	13451	81	2170
24	12	53	2376	82	1908
25	3	54	2128	83	10304
26	4	55	6032	84	5904
27	54	56	16590	85	3438
28	11	57	77	86	31590
29	171	58	6349	87	7340
30	33	59	945	88	5700
31	57	60	3007	89	46312
32	76	61	1196	90	39359
33	109	62	23248	91	313399
34	988	63	2517	92	29939
35	176	64	9702	93	17103
36	1874	65	4869	94	123594
37	522	66	6738	95	227112
38	4158	67	8572	96	74983
39	727	68	1556	97	91843
40	1171	69	17335	98	103538
41	1382	70	1741	99	4968
42	15744	71	6114	100	4507
43	20552	72	1588	101	123763

to the left of the bold cell in each row of these Figures. In other words, for the majority of the values of N, SUB spends more time executing $g(N,X)$ where $X < e_1^{(N)}$ than it does where $X \geq e_1^{(N)}$. This means that *proving* that a solution is optimal takes more time than *finding* this solution.

Another way to look at this is to notice that $g(N,X)$ for $X < e_1^{(N)}$ amounts to an exhaustive search, as opposed to $X \geq e_1^{(N)}$, where the search is terminated as soon as a solution is found. So even though the search tree is bigger in the latter case, its exploration stops early when a solution is found, whereas in the first case the whole tree must be traversed, taking more time.

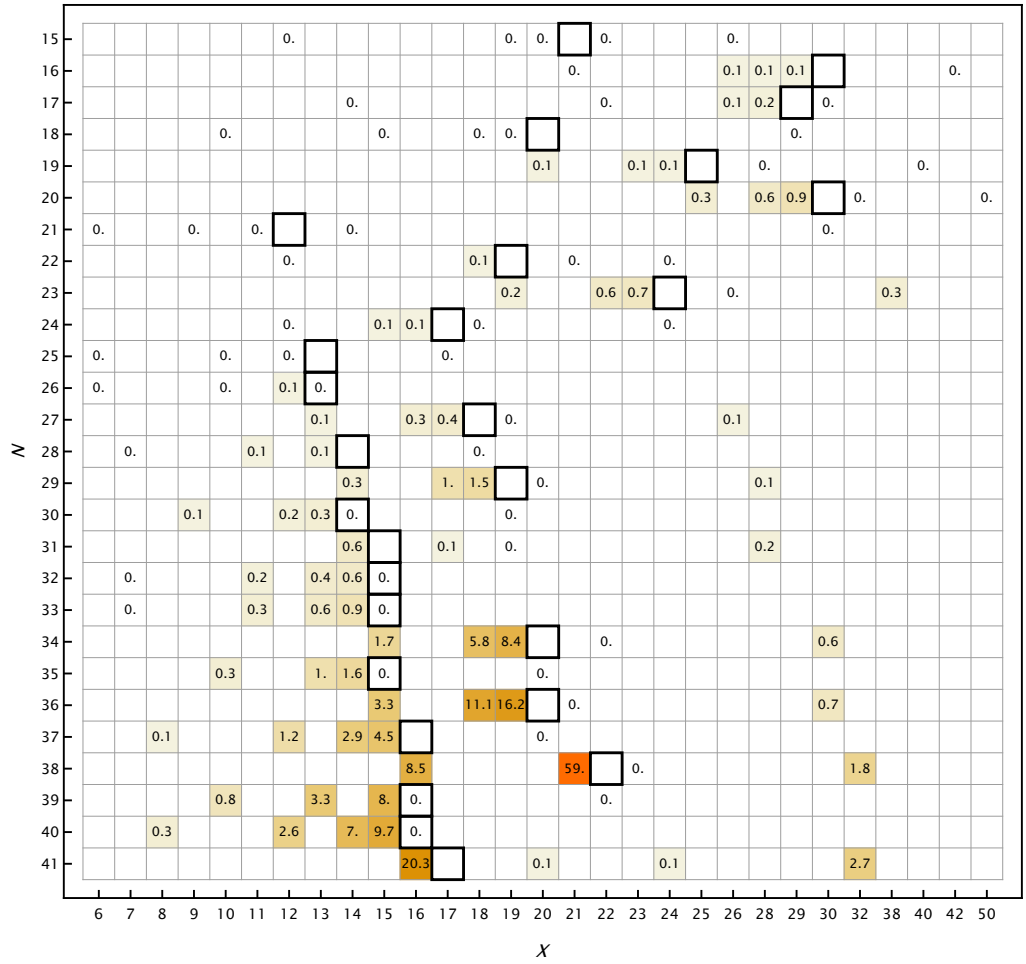

Figure 6 – Breakdown of times (in minutes) for $g(N,X)$ as executed during the solution procedure, for $N < 42$. Bold cells indicate $e_1^{(N)} = X$. Values are rounded to one decimal place.

Lastly, notice SUB will have to execute $g(N, e_1^{(N)} - 1)$ at some point to prove that $X = e_1^{(N)}$ is the optimal solution for N. Therefore, if SUB had executed $g(N,X')$ for any $X' < e_1^{(N)} - 1$ prior to this, it turns out (*a posteriori*) to have been a *waste of time*, since the search tree of $g(N,X')$ is contained in the tree of $g(N, e_1^{(N)} - 1)$.

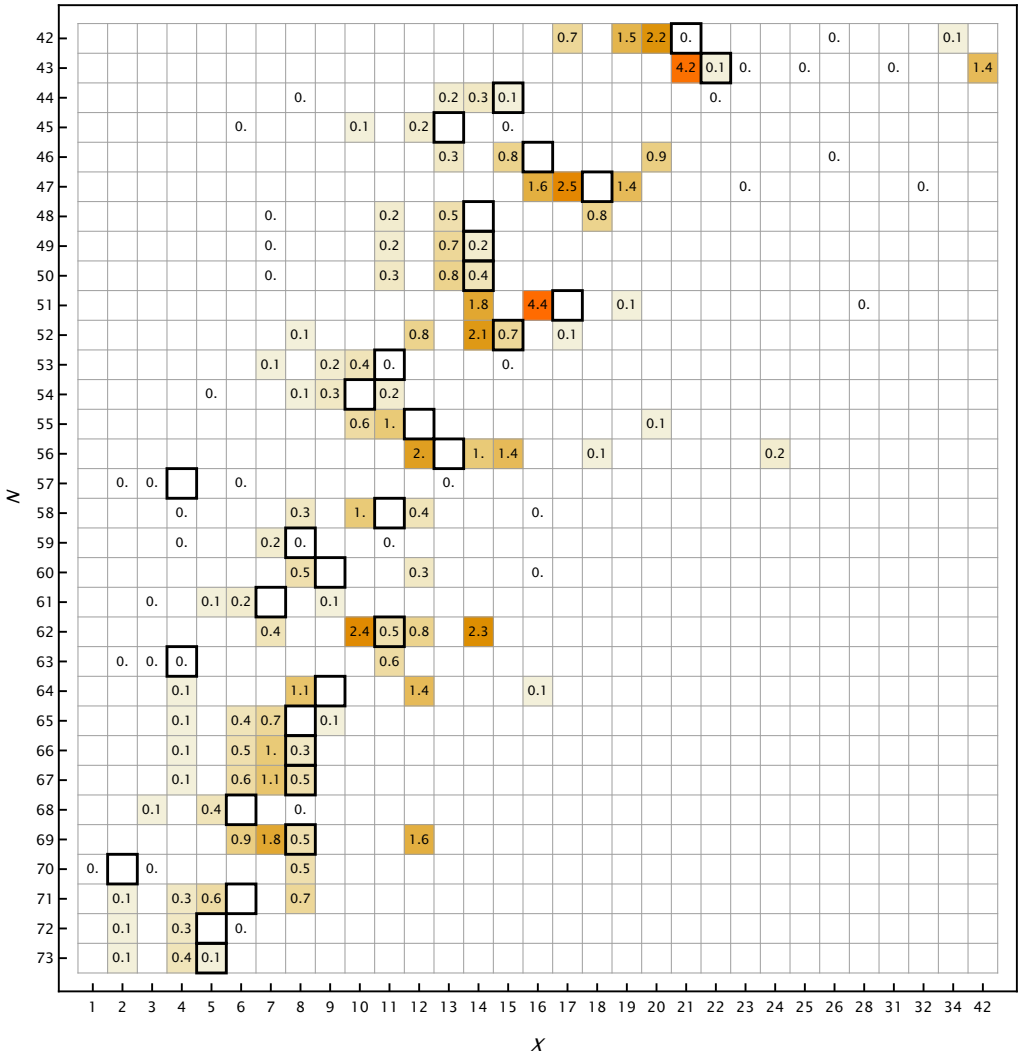

Figure 7 – Breakdown of times (in hours) for $g(N,X)$ as executed during the solution procedure, for $42 \leq N < 74$. Bold cells indicate $e_1^{(N)} = X$. Values are rounded to one decimal place.

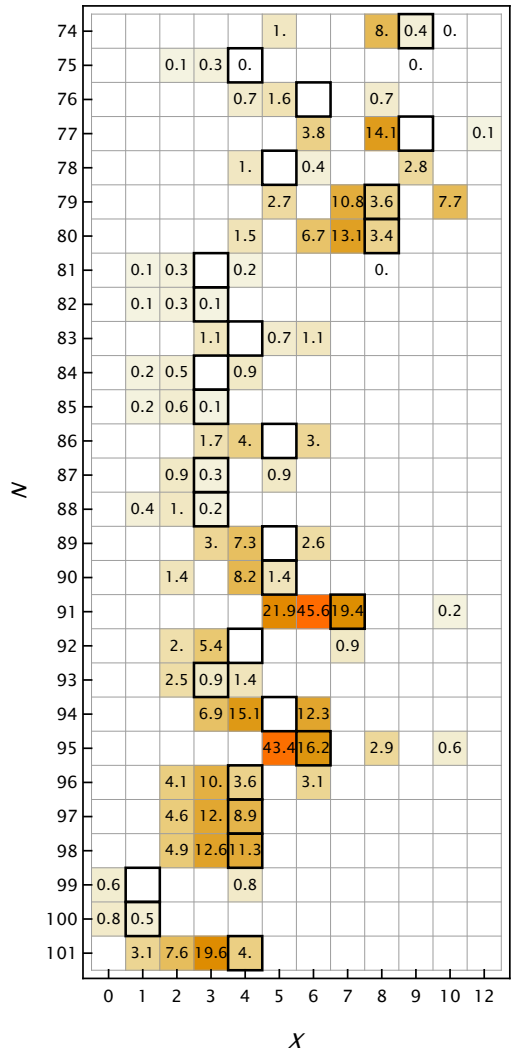

Figure 8 – Breakdown of times (in hours) for $g(N,X)$ as executed during the solution procedure, for $N \geq 74$. Bold cells indicate $e_1^{(N)} = X$. Values are rounded to one decimal place.

5 CONCLUSION

We considered the problem of finding the minimum uncovered area (trim loss) when tiling non-overlapping distinct integer-sided squares in an $N \times N$ square container. We solved it for all container sizes N from 1 to 101 using a binary search procedure called SUB, an independently developed adaptation of Ian Gambini's enumerative algorithm (here called g). The results were published as a new sequence to The On-Line Encyclopedia of Integer Sequences®. Though this text was only concerned with this particular problem, notice that the solution procedures developed here can be readily applied to the more general case of packing rectangles in a rectangular container.

We showed that it is effective to use the optimal trim loss of the previous container to initialize the $N \times N$ search. But SUB still cannot avoid calling $g(N,X)$ with X smaller than the optimal number $e_1^{(N)}$ of extra 1×1 squares, because this is necessary to refute that there exists a packing with $X < e_1^{(N)}$. The empirical data proved that these calls represent most of the search time, whilst $g(N,X')$ for $X \geq e_1^{(N)}$ takes actually less time than intuition would predict *a priori*. As future work, we suggest to use these observations as starting point to develop different solution procedures aiming to reduce the overall search time.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001.

The authors thank the anonymous referees for their helpful comments that improved the quality of the manuscript.

References

BIRÓ M & BOROS E. 1984. Network flows and non-guillotine cutting patterns. *European Journal of Operational Research*, **16**(2): 215–221.

BROOKS RL, SMITH CAB, STONE AH, TUTTE WT ET AL. 1940. The dissection of rectangles into squares. *Duke Mathematical Journal*, **7**(1): 312–340.

GAMBINI I. 1999. A method for cutting squares into distinct squares. *Discrete Applied Mathematics*, **98**(1-2): 65–80.

IORI M, DE LIMA VL, MARTELLO S, MIYAZAWA FK & MONACI M. 2021. Exact solution techniques for two-dimensional cutting and packing. *European Journal of Operational Research*, **289**(2): 399–415.

LESH N, MARKS J, MCMAHON A & MITZENMACHER M. 2004. Exhaustive approaches to 2D rectangular perfect packings. *Information Processing Letters*, **90**(1): 7–14.

LEUNG SC, ZHANG D, ZHOU C & WU T. 2012. A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. *Computers & Operations Research*, **39**(1): 64–73.

OEIS FOUNDATION INC. 2022. The On-Line Encyclopedia of Integer Sequences. Available at: https://oeis.org.

SHIANGJEN K, CHAIJARUWANICH J, SRISUJJALERTWAJA W, UNACHAK P & SOMHOM S. 2018. An iterative bidirectional heuristic placement algorithm for solving the two-dimensional knapsack packing problem. *Engineering Optimization*, **50**(2): 347–365.

WEI L, OON WC, ZHU W & LIM A. 2011. A skyline heuristic for the 2D rectangular packing and strip packing problems. *European Journal of Operational Research*, **215**(2): 337–346.

WEI L, ZHANG D & CHEN Q. 2009. A least wasted first heuristic algorithm for the rectangular packing problem. *Computers & Operations Research*, **36**(5): 1608–1614.

How to cite

ARRUDA VPR, MIRISOLA LGB & SOMA NY. 2022. Almost Squaring the Square: Optimal Packings for Non-decomposable Squares. *Pesquisa Operacional*, **42**: e262876. doi: 10.1590/0101-7438.2022.042.00262876.