Acta Univ. Sapientiae, Informatica **10,** 1 (2018) 73–85

DOI: 10.2478/ausi-2018-0005

# Exact fit problem generator for cutting and packing, revisiting of the upper deck placement algorithm

Levente FILEP
Sapientia University
Department of Economic Sciences
Miercurea Ciuc
email:
fileplevente@uni.sapientia.ro

László ILLYÉS
Sapientia University
Department of Economic Sciences
Miercurea Ciuc
email:
illyeslaszlo@uni.sapientia.ro

**Abstract.** Problem generators are practical solutions for generating a set of inputs to specific problems. These inputs are widely used for testing, comparing and optimizing placement algorithms. The problem generator presented in this paper fills the gap in the area of 2D Cutting & Packing as the sum of the area of the small objects is equal to the area of the Large Object and has at least one perfect solution. In this paper, the already proposed Upper Deck algorithm is revisited and used to test the proposed generator outputs. This algorithm bypasses the dead area problem that occurs in most of all well-known strategies of the 2D Single Knapsack Problem where we have a single large rectangle to cover with small, heterogeneous rectangle shapes, whom total area exceeds the large object's area. The idea of placing the small shapes in a free corner simplifies and speeds the placement algorithm as only the available angles are checked for possible placements, and collision detection only requires the checking of corners and edges of the placed shape. Since the proposed generator output has at least one exact solution, a series of optimization performed on the algorithm is also presented.

**Computing Classification System 1998:** I.2.8. [Problem Solving, Control Methods, and Search]: Subtopic - Heuristic methods
**Mathematics Subject Classification 2010:** 68T20, 90B40
**Key words and phrases:** Cutting stock problem, problem generator, genetic algorithm

## 1 Introduction

Cutting and Packing (C&P) problems are of great interest in operation research. From storage filling to memory allocation, the knapsack problem appears in many forms. As these problems are NP-complete (non-deterministic polynomial time), meta-heuristic approaches, such as Genetic Algorithms (GA), are popular as they provide good-enough solutions.

Problem generators are seen as a tool to overcome limitations imposed by the absence of appropriate test problems [14] in the field of C&P. The problem generator proposed and presented in this paper aims to fill the gap in the area by providing problems with at least one exact solution which, besides benchmarking, can be used to improve upon existing algorithms.

The upper deck placement GA, which was presented by one of the authors in the 2nd ESICUP meeting [6], is revisited in this paper and improvement attempts are made using the output of the proposed problem generator.

### 1.1 Problem generators

There are many problem generators developed for C&P. For our problem of two-dimensional rectangle cutting problem, the existing generator is the 2DCPackGen [14] developed by Silva, Oliveira and Wsher, downloadable from [17]. Since each of these generates shapes whose summed areas exceeds the surface area there is no best-known solution to these. While this is not a problem, since the aim of meta-heuristic algorithms is to provide a good enough solution for this kind of problems, it is a problem if we need to know of better solutions to further optimize the placing algorithms. Using brute force to try out all possible combinations to obtain the best possible solution its not a practical option. On the 2ndESICUP meeting, Alvarez-Valdez [1] defined the problems with perfect solutions as Jigsaw problems. T. Inamichi, et al. [9] proposed algorithms for perfect packing problems. Mumford and Wang [13] uses a perfect guillotine-cut example to check the performance of their algorithms. The first approach to the Jigsaw generator problem was from Illys and Fbin [7], but it was not implemented. This paper presents such an exact fit or in other words a perfect matching 2D, non-guillotine cutting stock problem generator.

### 1.2 Cutting stock problems

According to the typology of cutting stock problems [16], our work falls into the two-dimensional Single Knapsack Problem (2D-SKP) category. Previous algorithms to solve this problem are the Bottom Left algorithm (BL) [10], the improved BL algorithm [12], the BL Fill algorithm [4]. There are other, newer algorithms proposed and presented, but these are the first best approaches.

## 2 Proposed problem generator

In contrast to existing generators which generate shapes with a given distribution, usually gamma, until the combined area of these exceeds the cutting stock area, our generator starts with the cutting stock area and uses expanding rectangles placed at random locations to achieve complete coverage of this area. As opposed to other methods of randomly cutting an area into smaller pieces, the presented method was chosen due to ease of implementation in an arbitrary programming language.

For easy use, the generator is provided a graphical interface where the input parameters can be set: the width and height of the cutting stock area, the initial seed used for the pseudo-random number generator, the minimum and maximum shape extensions (in other words the growth of the shapes) and a cutoff ratio for random shape location. The width and height determine the unit square size of the stock cutting area, where each generated shape size is a multiple of this unit area.

The generation process starts with a given size area represented as a $W \times \mathsf{H}$ (width, height) grid on which small shapes are placed and expanded in each step. These steps are repeated by the generator until these shapes cover the entire surface area.

Each step begins by randomly selecting an $\mathsf{X}$ and $\mathsf{Y}$ location on the grid. If the chosen coordinates are outside of any existing shape, in other words, it's a free cell, then a new $1 \times 1$ size shape is created at the location and marked for the next operation. If the coordinates are inside an existing shape then, that shape is marked.

Using the previously marked shape, if possible, this is expanded in a random direction: up, down, left or right. The number of expansion attempts of the shape is dictated by the input parameters.

It is trivial to observe that as the number of free unit squares decreases it's harder to hit any new unoccupied cells by generating random $\mathsf{X}, \mathsf{Y}$ locations, consequently the algorithm's progression is slowing down. The $\mathsf{FaCR}$ cutoff ratio represents the ratio between the free and total cells after which instead of choosing random locations, the generator selects one from the remaining free cells, marks it as a new shape and expands this until possible. This operation is repeated until all cells are occupied by a shape, thus completing the generation process and achieving a perfect matching problem. This measure is needed to ensure that the generation process ends in a timely manner. From experimental results a $0.1-0.25$ cutoff value is ideal. Figure 1 illustrates some of the steps and the end result.

**Algorithm 1:** Pseudo-code for problem generator

```
Data: Input parameters:
        W,H: width and height of the cutting stock area
        FaCR: Free area Cutoff Ratio
        minEs, maxEs: minimum and maximum number of shape
        expansion
Result: shape list
begin
    TotalArea ← W ∗ H;
    FreeArea ← 0;
    while (FreeArea/TotalArea) > FaCR do
        Generate random X, Y coordinates // 0 ⩽ X ⩽ W, 0 ⩽ Y ⩽ H
        if Grid[X, Y] is empty then
            Generate new 1 × 1 Shape[i] at Grid[X, Y] location;
            Mark Grid[X, Y] as part of Shape[i];
        Es ← Random() // minEs ⩽ Es ⩽ maxEs
         Expand Shape[i] Es number of times;
        Recalculate FreeArea;
    while FreeArea ¿ 0 do
        Get X, Y of next free unit square;
        Generate new 1 × 1 Shape[i] at Grid[X, Y] location;
        Es ← Random() // minEs ⩽ Es ⩽ maxEs
         Expand Shape[i] Es number of times;
        Recalculate FreeArea;
```

## 3 Upper deck placement genetic algorithm

Placement algorithms are used to place small objects, known as shapes  in our case rectangles  on a Large Object, known as cutting stock  in our case a rectangle too. The sides of the small pieces are parallel to the stock plate, the pieces do not overlap each other and do not exceed the dimension of the stock plate.

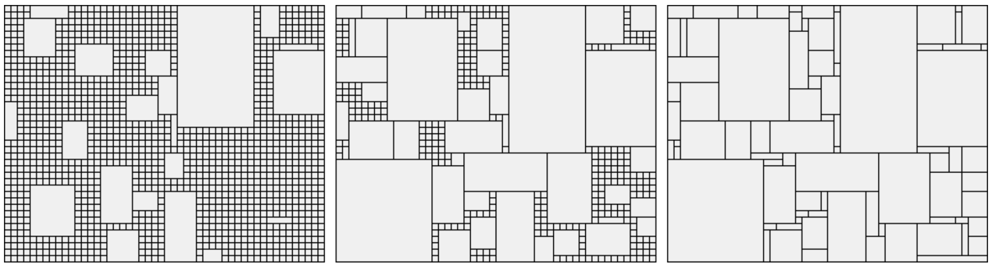


Figure 1: Shape generation process, early, mid and final stage

The genetic algorithm version of the upper deck algorithm was proposed at a previous conference [6], therefore a full version of this is not described here. Instead, we reflect on the differentiating parts from a conventional placement GA, namely the shape placement, angle generation part and the genetic operators of this. Before this paper, only the little objects order of placement was coded in permutation chromosomes. The rotation possibility was also addressed by Hopper and Turton in [3] but was not encoded in the gene, while the deck was not addressed at all.

Each input shapes is provided with a unique identification id. Even if two shapes are identical they are given separate ids and treated as distinct. This sacrifices some memory for computational speed gains since avoids the necessity of counting and checking the correct number of placed shapes against the input at each genetic operation.

## 3.1 Chromosome structure

Each gene in the chromosome represents a shape's placement properties. These include: unique identification `id`, a set of Boolean value indicating whether the shape is placed or not at the stored angle number and its rotation.

## 3.2 Heuristic

The strong point of the algorithm consists of the utilized heuristic. The algorithm maintains a list of the available angles where the next shape can be placed. This significantly lowers the number of possibilities needed to be checked for a possible shape placement, thus increasing the computational efficiency of the algorithm. The notation of the available angles is in the order of appearance and clockwise as shown in Figure 2. As new shapes are placed, depending on the placement, the number of angles can increase or decrease, but the notation is always maintained.

By definition [6], the upper angles are defined as the angle created by the intersection of either two of the cutting stock area walls, the edges of previously placed shapes or a combination of these. Therefore these are the concave angles. This definition has practical reasons behind it. As an example let's consider the filling a storage area that has a single entrance, where placing crates in concave angles would be unpractical.

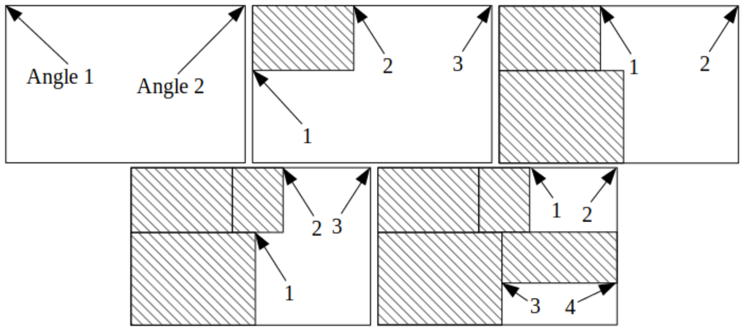


Figure 2: Angle notation on various stages of placement

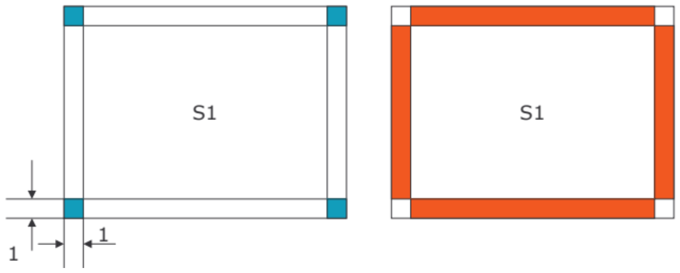


Figure 3: Collision detection: corners first, then along the edges

## 3.3 Collision detection

For computational speed improvement, as proposed prior [6], a raster grid is used to detect the possibility of placement. This sacrifices additional memory in favor of computational speed. The collision detection in this case, as illustrated in Figure 3, only involves checking the corners first and then checking against the borders in the raster grid. Another advantage is that the number of already placed shapes does not impact the detection speed since this depends only on the number of angles which has to be checked and on the length of the edges (width, height) of the shape being tested.

### 3.4 Genetic operators

Roulette Based Fitness (RBF) selection and Partially mixed-cyclic (PMX-CX) crossover are used. The fitness value is $F = 1 - R$, where $R$ is the waste percentage

$$R = \frac{\mathsf{Area}_{\mathsf{free}}}{\mathsf{Area}_{\mathsf{total}}},$$

$R \in [0, 1]$, therefore $F \in [0, 1]$. A higher value of $F$ indicates a better solution than a lower one. Mutation can either change the rotation of an angle or switch two genes. Since shape id is unique, their change it's not allowed.

## 4 Results and improvements

The first test of the algorithm was using an already presented problem [5, 8]. The problem is presented in more detail in Section 4.4. The initial tests were conducted using a population of 100 individuals, 1000 generations, 0.15 crossover and 0.05 mutation. Both crossover and mutation values express the fraction of affected population by the genetic operators in a [0..1] interval. The algorithm generated a best fitness value of 0.969 with an average best fitness value of 0.941 out of 100 runs. The fitness value was collected after each 100th generation. This result is illustrated in Figure 5a by the straight line.

However, using an input of 558 shapes listed in Appendix Input list, 120x110, 558 shapes with an cutting stock area of $120 \times 110$ from our generator, as indicated by the dotted line on the same figure, the algorithm convergence, in our opinion, was extremely low over the 1000 generations. Arguably the difference is due to the exact fit solution generated. In this case, each of the shapes not placed has a far bigger impact on the fitness value compared to the previous test where the number of shapes used has a far greater total area than the cutting stock. To improve on this limitation, instead of placing a shape at a random angle, we modified the algorithm to try all possible angles of placement available at a given step. The following sections describe the improvements obtained using this idea.

## 4.1 Crossover and mutation operation improvement

Since each gene stores the placed angle number for the shape, in case of crossover and mutation operations some of these will fall outside of the available angle numbers at a given step. While this is not necessarily an issue because with generation progression these individuals are eliminated, in practice, we found that this results in unreasonable slow convergence over the 1000 generations. To overcome this limitation all mutated chromosomes are reevaluated, the placement angles of the shapes are, if needed, recalculated and the genes updated. In such case, the shape is attempted to be placed again at the available angles. This in term, requires that the rest of the chromosome to be updated as well. The operation requires an additional computational effort, however, the result is a significant convergence improvement as seen in Figure 5b represented by the straight line compared to the result shown in Figure 5a with the dotted line.

We also applied the concept of elite individuals [15], the aim of which is to ensure that individuals carrying the best solutions propagate to the next generation. The result of this improvement is also incorporated in the result illustrated in Figure 5b with the straight line.

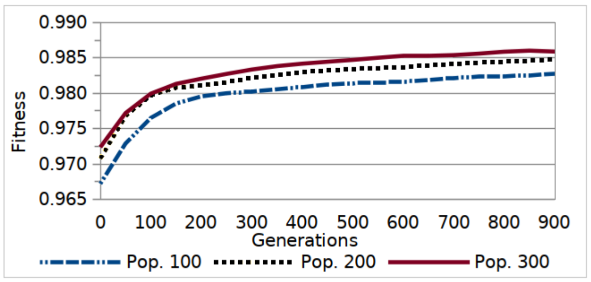
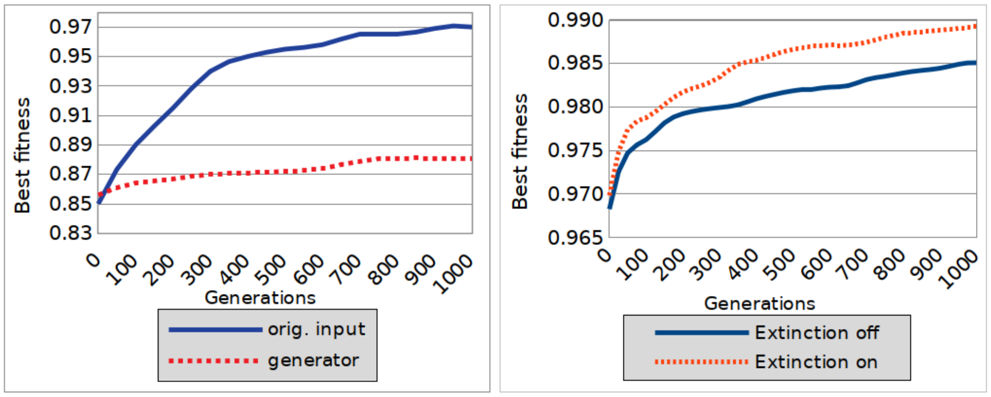


Figure 4: Population size influence on fitness

## 4.2 Initial population and maximum generation size

After we achieved a satisfying conversion, we investigated the quality of the starting population and the size of maximum generations. As there is an exact fit solution to the input used but also noticing the fitness convergence, we tried to improve the quality of the initial population by applying the above idea of testing all possible angles when trying to place a given shape. This resulted in the starting population's best fitness value improvement by an average value



(a) initial results (b) improved algorithm

Figure 5: Comparison between the original and the improved algorithm

of 0.1122 (11.22%). This result is observable in the difference of the starting fitness values between Figure 5a and Figure 5b.

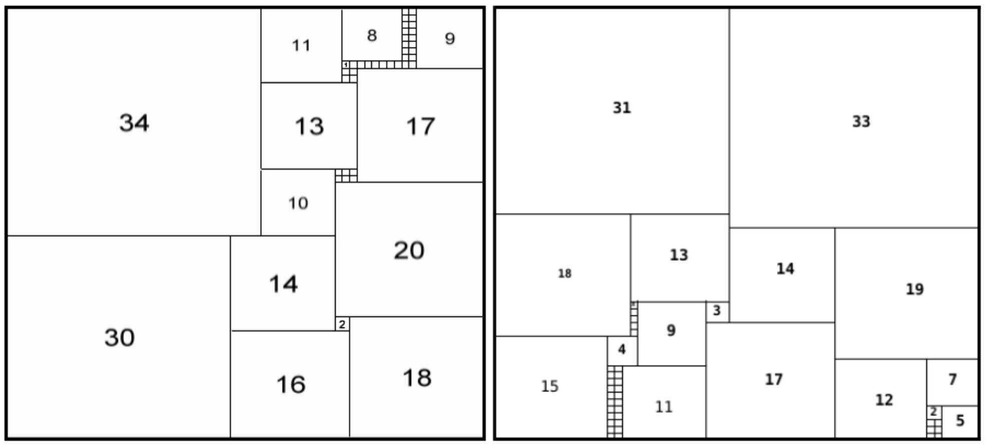
Testing the population size influence on the algorithm's result quality we considered that the average fitness value increase is beneficial up to a population of 200 − 250 for the current set of input. Illustrated in Figure 4, the result was obtained out of 10 runs using the 558 input shape list and the fitness value collected every 100th generation. As a result, we considered a population of 250 with a maximum of 1000 generations for the further tests.

## 4.3 Mass extinction

For even better convergence to the known solution, the notion of mass extinction [11] which shows some promise in regards to certain types of genetic algorithms [2] was also tested. Using preliminary tests, parameter values of 0.25 and 250 are used, meaning that a quarter of the population is eliminated each 250th generations. After the extinction process, the crossover probability is dynamically increased to double the value to allow for population regeneration. In this case, the chance increases for weaker fitness value individuals to produce offsets. The theoretical benefit of the extinction process is that the newly created individuals increase the population diversity, thus allowing the population to explore more of the search space or if being stuck in a local maxima this operation increases the chance of breaking out. The result obtained from a 100 consecutive runs, as illustrated in Figure 5b with the dotted line, shows a slight improvement of the average best fitness values by 0.00418 (0.418%) with a lightly faster convergence.

### 4.4 Revisiting the 64x64 particular covering problem

With the improved algorithm, we revisited the particular covering problem [5, 8] of placing most squares from a list of $1 \times 1$, $2 \times 2$, to $46 \times 46$ on a $64 \times 64$ grid, trying to get the most cell coverage. Here we managed to improve on the existing solution presented in the old paper where the best result obtained left 35 free cells, illustrated in figure Figure 6a, while the new solution, illustrated in figure Figure 6b, has only 32 free. The result was obtained from 100 runs and the best fitness was achieved twice, in runs 24 and 71.



(a) Previous result with 35 free space (b) Improved result with 32 free space

Figure 6: Improvement made to the particular placement problem

## 5 Conclusions and future work

In this paper, we presented an exact fit 2D problem generator which fills a gap in the problem generators area. Preliminary testing was done using the revisited upper deck placement genetic algorithm. Using the input from the generator a series of improvements made to the algorithm was investigated and presented. These improved both the quality of the starting population as well as the convergence of the GA. We also revisited the $64 \times 64$ particular covering problem where we managed to improve on the previous result. Further development plans for the presented generator include the option to generate guillotine stock problems as well as three-dimensional problems.

## Acknowledgements

This work was partially supported by the Collegium Talentum 2017 Programme of Hungary.

## References

[1] R. Alvarez-Valdez, F. Parreño, J.M. Tamarit, A Tabu Search Algorithm for two-dimensional, non-quillotine problems, *2nd ESICUP Meeting*, Southampton, England, 2005. ⇒74

[2] H. David Mathias, R. Vincent, An empirical study of crossover and mass extinction in a genetic algorithm for pathfinding in a continuous environment, *2016 IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada, 2016, pp. 4111–4118. ⇒81

[3] E. Hopper, B. C. H. Turton, A genetic algorithm for a 2D industrial packing problem, *Computers & Industrial Engineering* **37,** (1999), 375–378. ⇒77

[4] E. Hopper, B. C. H. Turton, An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, *European Journal of Operational Research* **128,** 1 (2001) 34–57. ⇒75

[5] L. Illyés, Genetic algorithms for a particular covering problem, *International Conf. on Economic Cybernetics*, Bucuresti, Romania, 2004. ⇒79, 82

[6] L. Illyés, Upper Angle Placement Algorithm with Genetic Approach for 2D Rectangle Knapsack Problem, *2nd ESICUP Meeting*, Southampton, England, 2005. ⇒74, 77, 78

[7] L. Illyés, Cs. Fábián, "Jigsaw" problem generator for 2D rectangle single large object for non-guillotine and guillotine cutting, *Proc. WSCSP2005, Workshop on Cutting Stock Problems*, Miercurea-Ciuc, Romania, 2005, pp. 83–89. ⇒74

[8] L. Illyés, L. Pál, Generalized particular covering problem with genetic algorithms, *AMO - Advanced Modeling and Optimization*, **7,** 1, 2005, 1–7. ⇒79, 82

[9] T. Inamichi et. al., Branch-and-bound algorithms for regular strip packing and perfect packing problems, *2nd ESICUP Meeting*, Southampton, England, 2005. ⇒74

[10] S. Jakobs, On genetic algorithms for packing polygons, *European Journal of Operational Research* **88,** (1996), 165–181. ⇒74

[11] B. Jaworski, Kuczkowski, R. Śmierzchalski, Extinction Event Concepts for the Evolutionary Algorithms, *Przeglad Elektrotechniczny*, **1,** 88 (2012), 252–255. ⇒81

[12] D. Liu, H. Teng, An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles, *European Journal of Operational Research* **112,** (1999), 413–420. ⇒75

[13] C. L. Mumford, P. Y. Wang, A genetic simulated annealig approach to aacking, *2nd ESICUP Meeting*, Southampton, England, 2005. ⇒74

[14] E. Silva, J. F. Oliveira, G. Wäsher, A problem generator for two-dimensional rectangular cutting and packing problems, *ESICUP.* (2007), https://paginas.fe.up.pt/~esicup/problem_generators. ⇒74

[15] M. Villalobos-Arias, C. Coello Coello, O. Hernndez-Lerma, Asymptotic convergence of some metaheuristics used for multiobjective optimization, *FOGA 2005: Foundations of Genetic Algorithms*, Aizu-Wakamatsu, Japan, 2005, pp. 95–111. ⇒80

[16] G. Wäscher, H. Hauner, H. Schumann, An improved typology of cutting and packing problems, *European Journal of Operational Research* **183,** 3 (2007), 1109–1130. ⇒74

[17] *EURO Special Interest Group on Cutting and Packing*, `https://paginas.fe.up.pt/~esicup/problem_generators` ⇒74

## Appendix

## Input list, 120x110, 558 shapes

The following table contains the list of input shapes used for testing the GA. The values in column "#" represents the id of the shape, columns "W." and "H." its width and height, while column "Nr." indicates the quantity of this.

| # | W. | H. | Nr. |
|---|---|---|---|
| 1 | 1 | 1 | 34 |
| 2 | 1 | 2 | 19 |
| 3 | 1 | 3 | 9 |
| 4 | 1 | 4 | 9 |
| 5 | 1 | 5 | 13 |
| 6 | 1 | 6 | 5 |
| 7 | 1 | 7 | 2 |
| 8 | 1 | 8 | 4 |
| 9 | 1 | 11 | 1 |
| 10 | 2 | 1 | 15 |
| 11 | 2 | 2 | 18 |
| 12 | 2 | 3 | 19 |
| 13 | 2 | 4 | 11 |
| 14 | 2 | 5 | 7 |
| 15 | 2 | 6 | 7 |
| 16 | 2 | 7 | 2 |
| 17 | 2 | 8 | 3 |
| 18 | 2 | 9 | 1 |
| 19 | 2 | 15 | 1 |
| 20 | 3 | 1 | 18 |
| 21 | 3 | 2 | 22 |
| 22 | 3 | 3 | 14 |
| 23 | 3 | 4 | 8 |
| 24 | 3 | 5 | 4 |
| 25 | 3 | 6 | 11 |
| 26 | 3 | 7 | 1 |
| 27 | 3 | 9 | 2 |
| 28 | 3 | 10 | 1 |
| 29 | 3 | 11 | 2 |
| 30 | 3 | 13 | 3 |
| 31 | 3 | 20 | 1 |
| 32 | 4 | 1 | 10 |
| 33 | 4 | 2 | 9 |
| 34 | 4 | 3 | 12 |
| 35 | 4 | 4 | 10 |
| 36 | 4 | 5 | 6 |
| 37 | 4 | 6 | 5 |
| 38 | 4 | 7 | 2 |
| 39 | 4 | 8 | 1 |
| 40 | 4 | 9 | 2 |
| 41 | 4 | 10 | 2 |
| 42 | 5 | 1 | 7 |
| 43 | 5 | 2 | 8 |
| 44 | 5 | 3 | 6 |
| 45 | 5 | 4 | 10 |
| 46 | 5 | 5 | 7 |
| 47 | 5 | 6 | 6 |
| 48 | 5 | 7 | 5 |
| 49 | 5 | 8 | 1 |
| 50 | 5 | 9 | 3 |
| 51 | 5 | 10 | 1 |
| 52 | 5 | 11 | 1 |
| 53 | 5 | 12 | 1 |
| 54 | 6 | 1 | 5 |
| 55 | 6 | 2 | 6 |
| 56 | 6 | 3 | 3 |
| 57 | 6 | 4 | 9 |
| 58 | 6 | 5 | 9 |
| 59 | 6 | 6 | 5 |
| 60 | 6 | 7 | 3 |
| 61 | 6 | 8 | 3 |
| 62 | 6 | 9 | 3 |
| 63 | 6 | 10 | 1 |
| 64 | 6 | 11 | 1 |
| 65 | 6 | 12 | 1 |
| 66 | 6 | 14 | 1 |
| 67 | 6 | 15 | 2 |
| 68 | 6 | 19 | 1 |
| 69 | 7 | 1 | 6 |
| 70 | 7 | 2 | 7 |
| 71 | 7 | 3 | 3 |
| 72 | 7 | 4 | 3 |
| 73 | 7 | 5 | 4 |
| 74 | 7 | 6 | 4 |
| 75 | 7 | 7 | 2 |
| 76 | 7 | 8 | 1 |
| 77 | 7 | 9 | 1 |
| 78 | 7 | 12 | 3 |
| 79 | 7 | 13 | 2 |
| 80 | 8 | 1 | 1 |
| 81 | 8 | 2 | 6 |
| 82 | 8 | 3 | 1 |
| 83 | 8 | 4 | 1 |
| 84 | 8 | 5 | 3 |
| 85 | 8 | 6 | 1 |
| 86 | 8 | 7 | 1 |
| 87 | 8 | 8 | 2 |
| 88 | 8 | 9 | 1 |
| 89 | 8 | 10 | 2 |
| 90 | 8 | 12 | 1 |
| 91 | 8 | 13 | 1 |
| 92 | 9 | 1 | 1 |
| 93 | 9 | 2 | 2 |
| 94 | 9 | 3 | 1 |
| 95 | 9 | 4 | 3 |
| 96 | 9 | 5 | 2 |
| 97 | 9 | 7 | 2 |
| 98 | 9 | 8 | 1 |
| 99 | 9 | 9 | 2 |
| 100 | 9 | 10 | 1 |
| 101 | 9 | 11 | 2 |
| 102 | 9 | 15 | 1 |
| 103 | 10 | 2 | 1 |
| 104 | 10 | 4 | 2 |
| 105 | 10 | 6 | 1 |
| 106 | 10 | 7 | 2 |
| 107 | 10 | 8 | 3 |
| 108 | 10 | 12 | 1 |
| 109 | 10 | 13 | 2 |
| 110 | 10 | 16 | 1 |
| 111 | 11 | 3 | 1 |
| 112 | 11 | 5 | 3 |
| 113 | 11 | 6 | 1 |
| 114 | 11 | 10 | 1 |
| 115 | 11 | 11 | 1 |
| 116 | 11 | 12 | 1 |
| 117 | 12 | 4 | 2 |
| 118 | 12 | 5 | 2 |
| 119 | 12 | 7 | 1 |
| 120 | 12 | 8 | 2 |
| 121 | 12 | 10 | 1 |
| 122 | 13 | 3 | 1 |
| 123 | 13 | 4 | 1 |
| 124 | 13 | 5 | 1 |
| 125 | 13 | 6 | 2 |
| 126 | 13 | 7 | 1 |
| 127 | 13 | 12 | 1 |
| 128 | 14 | 4 | 1 |
| 129 | 14 | 5 | 2 |
| 130 | 15 | 2 | 1 |
| 131 | 15 | 6 | 1 |
| 132 | 15 | 9 | 2 |
| 133 | 16 | 4 | 1 |
| 134 | 16 | 5 | 1 |
| 135 | 16 | 6 | 1 |
| 136 | 19 | 9 | 1 |
| 137 | 22 | 9 | 1 |

*Received: March 15, 2018 • Revised: July 13, 2018*