Hardness in Perfect Rectangle Packing Problems

Estimating the Hardness of a Problem Without Solving It

H.R. Oosterhuis (10196129)

Supervisor: Daan van den Berg

Afstudeerproject Thesis
Bachelor Kunstmatige Intelligentie
Informatics Institute, Faculty of Science, Universiteit van Amsterdam
Abstract

Rectangle Packing is considered a difficult NP-problem which has several application areas such as integrated circuit systems, and pallet packing. Generally, Rectangle Packing Problems consist of a set of rectangles for which a configuration must be found, the intention is to minimise the surface-area of the configurations bounding box. Existing research has predominantly investigated algorithms to solve these problems as quickly as possible. Conversely, this study focuses on predicting the hardness of Perfect Rectangle Packing Problems, as the difficulty has been observed to vary in degrees. This may enable application areas to avoid the hardest Rectangle Packing Problems. This study proposes the use of the ES-attribute of a rectangle set, which is based on the number of shared sides it has, for predicting the difficulty of rectangle packing problems in terms of solution density. To support this proposition an algorithm was designed to generate rectangle-sets uniformly, subsequently a sizable set of rectangle-sets was generated. The number of solutions was calculated for each rectangle set; any configuration with zero empty-space is considered a solution, and sets with no perfect configuration were discarded. Accordingly, the total number of solutions is considered as a measure of difficulty. Subsequently, the data suggests an exponential correlation between the ES-attribute and the average difficulty of a Perfect Rectangle Packing Problem. The strength of this correlation is evaluated in relation to the cardinality of rectangle-sets and the ratio of bounding boxes. The results suggest the correlation is able to estimate the average number of solutions for a great amount of rectangle-set, although, it is inaccurate for single instances. However, instances with a large number of solutions occur more often at large ES-values, whereas at low ES-values only instances with an extremely low number of solutions exist. Thus, for an easy Perfect Rectangle Packing Problem a large ES-value is required, however, a large ES-value does not guarantee an easy problem instance.
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Related Work</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Method</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Results</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>Conclusion and Discussion</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>Future work</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Appendices</td>
<td>11</td>
</tr>
<tr>
<td>A</td>
<td>Uniform Generating of Rectangle Sets</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>A.1 Computing the Number of Potential Sets</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>A.2 Generating a Random Set Uniformly</td>
<td>15</td>
</tr>
</tbody>
</table>
1 Introduction

It is generally accepted that efficiency is one of the most desired qualities in the modern world. This is especially apparent in the field of computing hardware, where attempts to enhance the processing capabilities of electronic hardware have been successful for more than half a century. As a result, recent computing devices are capable of solving multitudinous problems which had been deemed impossible to solve before these technological advances. Nevertheless, in spite of the significant increase in computational capabilities, a particular group of problems remains extremely difficult to solve. Primarily, this is due to their exponential complexity, hence, increases in computational power are hardly effective. Although most hard problem types have been researched, research has predominantly focused on NP-complete problems which require a significant time to solve, but whose solutions are verifiable in polynomial time. Moreover, the time necessary to solve an NP-problem grows exponentially with its size. Their relevance is largely due to the great number of areas to which they can be applied, such as planning, scheduling, and spatial configurations. However, despite the sizable body of research regarding NP-complete problems, only a few studies have attempted to investigate what attributes of NP-problems instances result in severe difficulty [3][2]. Astonishingly, these studies suggest that the difficulty of an NP-problem instance correlates with a defining parameter, furthermore, such a parameter was found for the k-colorability problem and the traveling salesman problem.

Similarly, this study will investigate the difficulty of Perfect Rectangle Packing Problem instances, which has been observed to vary in degree. Rectangle Packing is the problem of determining a configuration of a set of rectangles which minimises the surface area of the configurations bounding box. Depending on the instance, this determination process can take up an extensive period of time. Moreover, Perfect Rectangle Packing Problems (PRPP) are instances of Rectangle Packing Problems where the optimal configuration contains no empty space, thus, the surface of the bounding box is equal to the summed surface of the rectangles in the set. Due to this extra constraint, PRPPs are significantly harder than regular Rectangle Packing Problems. Besides being interesting from a computational point of view, Rectangle Packing has applications in logistics, where goods must be packed as efficiently as possible; the design of integrated circuit systems [8], where the required rectangular modules are placed in the smallest container possible; and indirect applications such as scheduling problems [11]. In practice some instances of PRPPs are observed to be easily solvable, whereas comparable instances require hours of computation to solve. Correspondingly, the ability to predict how difficult a PRPP is would be beneficial, as it could enable practical applications to be adjusted appropriately to reduce the occurrence of hard PRPP-instances. Furthermore, it could encourage directions of research in heuristic algorithms for solving the hardest instances.

Accordingly, this paper introduces a computable attribute $ES$ and examines whether the $ES$ of a PRPP is capable of predicting its difficulty. For this examination, the difficulty of a PRPP-instance is defined as the number of solutions it has. Thus, this study will investigate whether a correlation exists between the $ES$-values of solvable PRPP-instances and the number of their solutions.
For this study, an extensive number of solvable PRPPs were generated uniformly and examined statistically, as described in section 3. Correlations between attributes and number of solutions are displayed in section 4, furthermore, this section also evaluates the strength of correlation and whether it is affected by other attributes of a PRPP. Section 5 examines the results of the last section and discusses the implications of these findings. Finally, section 6 discusses what future steps seem most promising.

2 Related Work

Wäscher et al. [12] describe an extensive topology of packing and cutting problems, which improves a typology provided by Dyckhoff [1]. The problem this paper will focus on is a subclass of the Single Large Object Placement Problem (SLOPP) which the topology classifies as easier than problems regarding more dimensions or non-rectangular shapes. SLOPP is characterised by a single large object which functions as a container, and a set of weakly heterogeneous smaller objects which have to be placed within the container without overlapping each other. These problems are categorised as placement problems, as opposed to problems with single container and a strongly heterogeneous set of smaller objects which are categorised as Single Knapsack Problems (SKP). The main distinction between SLOPP and SKP is that the latter depends more on selecting which rectangles are to be placed, whereas the former depends mostly on how the rectangles have to be placed. Interestingly, Wäscher et al. do not mention Perfect Rectangle Packing Problems, moreover, there is relatively little existing research regarding this particular subset of the Rectangle Packing Problems. Despite the fact that it can be considered as more difficult than instances where empty space is allowed.

Alternatively, the majority of existing research regarding Rectangle Packing Problems proposes or compares efficient solving-algorithms [6]. Although algorithms for Perfect Rectangle Packing Problems have been proposed [4][5][7], there is considerably less research that assesses this type of packing problems. The efficiency of these algorithms is mainly due to the use of clever pruning; the most common optimisations include: placing rectangles in bottom-left corners only, attempting to place the rectangles in descending order of surface area, and addressing the corner with the smallest adjacent open space first. Consequently, their strategies rely mostly on being able to prune quickly. As a result, benchmarks of solvable problems can be solved quickly, however, various instances deviate significantly from the average performance. Unfortunately, no research has been done to investigate what makes these particular instances more difficult than others.

Although, difficulty in Rectangle Packing Problems has not been researched considerably, Cheese-man et al. [3] investigated the difficulty of other well-known NP-problems. Conclusions drawn from instances of NP-problems which can be solved easily and instances that are extremely difficult to solve suggest that every NP-problem has at least one parameter that defines the problem. Moreover, most hard problems exist around critical values of this parameter. This is illustrated in a short analysis of the k-colorability problem for graphs, which displays the relation between the average degree of its nodes and the probability of a k-colour solution. Remarkably, this parameter closely describes a transition
in solution probability from nearly one to nearly zero. In addition, the threshold for this particular problem has been extensively analysed by Hayes [2]. Lastly, Cheeseman et al. claim all NP-problems have such a parameter and transition. Accordingly, this is highly relevant to the Perfect Rectangle Packing Problem as it is an NP-complete problem. Additionally, Helmut Simonis and Barry O’Sullivan have shown how a general constraint programming implementation is able to outperform ad-hoc algorithms for solving Rectangle Packing Problems [9]. They further argue that rectangle packing is a domain where constraint programming significantly outperforms ad-hoc systems. Interestingly, this suggests that Rectangle Packing shares many similarities to NP-problems in general. Together these suggestions seem to imply Perfect Rectangle Packing Problems have at least one defining parameter which determines their difficulty. Conversely, this contrasts with the traditional view that the difficulty of an NP-problem instance can only be determined by solving that particular instance.

3 Method

For this study, a sizeable number of PRPP-instances were generated and solved completely, resulting in a large dataset of rectangle sets and the total number of solutions for each set. For fair analysis the rectangle sets had to be comparable, thus, the decision was made to only compare sets of the same cardinality, as the number of rectangles has a major influence on the total number of possible configurations. Furthermore, each set consists of unique rectangles to ensure each set has an equal number of arrangements. In addition, sets in which all the rectangles share a side were not permitted, because these problems are primarily 1-dimensional and rectangle packing is considered to be a 2-dimensional problem. Thus a set such as \( s = \{2 \times 3, 2 \times 5, 2 \times 12, 2 \times 18\} \) is not permitted, as it is bound to have at least \(|s|!\) solutions consisting of a single row of rectangles. Lastly, a generating procedure was designed and implemented to generate sets uniformly, a detailed description of this algorithm is given in appendix A. Because the generating procedure is only capable of generating sets for a given number of rectangles and their total summed surface, the procedure was run with the number of rectangles ranging from five to twelve, and the total surface ranging from 600 to 1500. Approximately fifty million rectangle sets were generated, each set having at least one configuration with no empty space. Subsequently, the total number of solutions for each set were computed, this was done efficiently by using optimisations based on symmetry [10]. Additionally, the number of solutions for each potential bounding box was stored separately. Accordingly, a potential bounding box is a rectangle whose surface is equal to the total surface of the rectangle-set, and as such could potentially serve as the bounding box of a solution. Unfortunately, attempts to add sets with thirteen or more rectangles proved infeasible, because determining the number of solutions required an excessive amount of time. The process of creating the dataset was run on the LISA cluster maintained by SURFsara\(^1\). Using a multi-threaded implementation in Java, the code was executed on a multitude of nodes, which each had a Intel Xeon

---

\(^1\)https://surfsara.nl/systems/lisa
Processor L5640 (2.26 GHz), the execution took approximately a two hundred hours and required around 4 GB to store.

After acquiring the dataset, the ES-attribut e was computed for each set. This attribute is based on a function \( f \) which calculates the frequency of a side with length \( x \) in a set \( s \):

\[
f_s(x) = |\{w, h \in s \mid w = x\}| + |\{w, h \in s \mid h = x\}|
\]  

(1)

The ES-attribute is the squared sum of each possible \( f_s(x) \):

\[
ES_s = \sum_{i=0}^{\infty} f_s(i)^2
\]  

(2)

As this attribute is fundamentally related to the number of Equal Sides in a set, the name ES was chosen to be the most appropriate. Furthermore, it can be calculated in linear time, thus, it potentially has a predictive quality. The \( ES_s \) was computed for each rectangle-set in the dataset and stored accordingly.

Lastly, the dataset was analysed for a correlation between the computed ES values and the number of solutions for each respective rectangle-set. The number of solutions was split in the number of solutions with narrow bounding-boxes and the number with wide bounding-boxes, where narrow bounding-boxes are all rectangles whose smallest side is less or equal to three, and wide bounding boxes are the remaining rectangles. This division was made because PRPPs with narrow containers seem to behave differently than wide containers, this difference will be further discussed in the section 5. For both the narrow and wide bounding-boxes and for each set cardinality separately the following correlations were investigated: the correlation between the ES-attribut e and the number of solutions for each set, and the correlation between the ES-attribute and the average number of solutions per ES. Both correlations were investigated by fitting a exponential curve: \( f(x) = ae^{bx} + c \), and calculating the \( R^2 \) coefficient of determination to evaluate the fit. Moreover, a separate fit was made and its \( R^2 \) was calculated for each bounding-box, to determine the effect of the bounding-box ratio. The code for processing was implemented in Python using the matplotlib\(^2\) library for plotting graphs and SciPy\(^3\) for fitting curves.

4 Results

Figure 1 displays the average number of solutions in wide bounding-boxes per ES value, and an exponential curve which was fitted on the averages. The correlation between the fit and the data seems to be stronger for sets with a greater cardinality, to display this effect separate plots for each set cardinality ranging from five to twelve are displayed in figure 1. The fits are all exponential curves: \( f(x) = ae^{bx} + c \), table 1 and 2 display the parameters for wide- and narrow-solutions respectively.

\(^2\)http://matplotlib.org
\(^3\)http://scipy.org/
Furthermore, to illustrate the difference between wide and narrow containers the same plots have been made for narrow bounding-boxes, figure 3 displays them for cardinality eight and twelve. For a fair comparison, all the averages on display in figure 1 and 3 are based on at least 200 rectangle-sets.

Furthermore, to illustrate the difference between narrow and wide bounding boxes the $R^2$ per container ratio is displayed in figure 4. Additionally, a second graph displaying the $R^2$ for ratios between 0 and 0.1 is displayed, as this part of the graph is the most relevant to the difference between narrow and wide bounding boxes.

Finally, to demonstrate how the actual number of solutions differ from the averages, figure 2 displays a scatter plot of all the rectangle-sets with cardinality eleven, displaying their ES-attribute set against the number of their solutions with wide bounding boxes. In addition, the standard deviation for the same sets per ES value is displayed as well.

<table>
<thead>
<tr>
<th>n</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>$R^2$</th>
<th>number of sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$5.29 \times 10^6$</td>
<td>$2.20 \times 10^{-7}$</td>
<td>$-5.30 \times 10^6$</td>
<td>0.75</td>
<td>6855516</td>
</tr>
<tr>
<td>6</td>
<td>$6.47 \times 10^6$</td>
<td>$5.68 \times 10^{-7}$</td>
<td>$-6.47 \times 10^6$</td>
<td>0.88</td>
<td>6903165</td>
</tr>
<tr>
<td>7</td>
<td>$1.37 \times 10^3$</td>
<td>$7.41 \times 10^{-3}$</td>
<td>$-1.52 \times 10^3$</td>
<td>0.91</td>
<td>6923350</td>
</tr>
<tr>
<td>8</td>
<td>$3.51 \times 10^3$</td>
<td>$3.00 \times 10^{-2}$</td>
<td>$-5.59 \times 10^2$</td>
<td>0.92</td>
<td>6940527</td>
</tr>
<tr>
<td>9</td>
<td>$2.74 \times 10^2$</td>
<td>$4.32 \times 10^{-2}$</td>
<td>$-5.63 \times 10^2$</td>
<td>0.88</td>
<td>6966461</td>
</tr>
<tr>
<td>10</td>
<td>$1.51 \times 10^2$</td>
<td>$5.68 \times 10^{-2}$</td>
<td>$5.81 \times 10^2$</td>
<td>0.88</td>
<td>7008315</td>
</tr>
<tr>
<td>11</td>
<td>$4.34 \times 10^1$</td>
<td>$7.22 \times 10^{-2}$</td>
<td>$1.08 \times 10^4$</td>
<td>0.98</td>
<td>1821367</td>
</tr>
<tr>
<td>12</td>
<td>$7.97 \times 10^1$</td>
<td>$6.92 \times 10^{-2}$</td>
<td>$4.06 \times 10^4$</td>
<td>0.99</td>
<td>207040</td>
</tr>
</tbody>
</table>

Table 1: Parameters of fits on average number of wide-solutions per ES-value for wide configurations, as displayed in figure 1. Each fit was an exponential curve: $f(x) = ae^{bx} + c$. The fits correlate more strongly for larger sets.

<table>
<thead>
<tr>
<th>n</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>$R^2$</th>
<th>number of sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$2.27 \times 10^4$</td>
<td>$1.97 \times 10^{-6}$</td>
<td>$-2.27 \times 10^5$</td>
<td>0.37</td>
<td>1220341</td>
</tr>
<tr>
<td>6</td>
<td>$6.31 \times 10^6$</td>
<td>$2.86 \times 10^{-7}$</td>
<td>$-6.31 \times 10^6$</td>
<td>0.40</td>
<td>1334715</td>
</tr>
<tr>
<td>7</td>
<td>$9.26 \times 10^3$</td>
<td>$6.80 \times 10^{-6}$</td>
<td>$-9.26 \times 10^5$</td>
<td>0.39</td>
<td>1412309</td>
</tr>
<tr>
<td>8</td>
<td>$3.21 \times 10^6$</td>
<td>$7.82 \times 10^{-6}$</td>
<td>$-3.21 \times 10^6$</td>
<td>0.66</td>
<td>1498632</td>
</tr>
<tr>
<td>9</td>
<td>$6.02 \times 10^3$</td>
<td>$1.07 \times 10^{-2}$</td>
<td>$-6.63 \times 10^3$</td>
<td>0.62</td>
<td>1621370</td>
</tr>
<tr>
<td>10</td>
<td>$-1.89 \times 10^8$</td>
<td>$-3.56 \times 10^{-6}$</td>
<td>$1.89 \times 10^8$</td>
<td>0.76</td>
<td>1801173</td>
</tr>
<tr>
<td>11</td>
<td>$1.24 \times 10^3$</td>
<td>$4.11 \times 10^{-2}$</td>
<td>$3.86 \times 10^4$</td>
<td>0.86</td>
<td>506457</td>
</tr>
<tr>
<td>12</td>
<td>$1.84 \times 10^3$</td>
<td>$4.62 \times 10^{-2}$</td>
<td>$2.08 \times 10^5$</td>
<td>0.81</td>
<td>61865</td>
</tr>
</tbody>
</table>

Table 2: Parameters of fits on average number of solutions per ES-value for narrow containers, two of which are displayed in figure 3. Each fit was an exponential curve: $f(x) = ae^{bx} + c$. Although, the fits correlate more strongly for larger sets, they correlate significantly less than the fits for narrow configurations from table 1.
Figure 1: Average number of solutions per ES-value for set-sizes ranging from 5 to 12. An exponential fit is made on each of the graphs, and as can be seen from the correlation coefficient $R^2$ these fits correlate more strongly for larger sets. Evidently, the average number of solution correlates with the ES-attribute, furthermore, for all set-sizes displayed a larger ES-value increases the average number of solutions.
Figure 2: Left: scatter plot of rectangle-sets of size 11. Right: standard deviation in number of solutions per ES-value. The large deviation for large ES-values can be explained by the occurrence of hard problems for each ES-value; the left graph shows that though the average number of solutions increases instances with near zero number of solutions exist everywhere. As a result, the deviation increases similar to the average number of solutions.

Figure 3: Average number of narrow-solutions per ES-value for set-sizes ranging for 8 and 12. These graphs are similar to the graphs for wide-solutions displayed in figure 1. Although the same correlations can be observed, they are significantly less strong for narrow solutions.

Figure 4: For each bounding-box a separate exponential fit on the average number of solutions with that bounding-box per ES-value was made, the correlation coefficient per ratio is displayed in these graphs. The majority of ratios have a high coefficient, expect for the ratios nearing zero. This suggests the number of solutions for bounding-boxes with coefficients near zero are differently related to the ES-attribute.
5 Conclusion and Discussion

Astonishingly, a correlation between the average number of solutions and $ES$ appears to exist. As can be seen in figure 1 the correlation for wide containers becomes gradually stronger as the set cardinality increases, culminating in a coefficient of 0.99 for cardinality 12. Similarly, this effect is also present for narrow containers; as is visible in figure 3 this correlation is stronger for a greater cardinality, albeit with a less significant coefficient than for the wide containers.

Furthermore, the decision to split narrow and wide containers is justified by the difference displayed in figure 4. From the graphs it can be inferred that the correlation between $ES$ and the number of solutions is strong, except for values nearing zero. This difference is further supported by the difference of figure 1 and figure 3, where for the wide containers and a cardinality of eight an exponential correlation is evident. Conversely, for the same cardinality but with the narrow containers, a correlation is less apparent.

Although, it is clear a correlation between the average number of solutions and the $ES$-attribute exists, the deviation in the number of solutions is considerable. The deviations displayed in figure 2 are significant for high $ES$ values, this can be explained by the scatter plot which shows that every $ES$ value has instances with a number of solutions close to zero. However, it appears that for low $ES$ the number of solutions is significantly small with little variation. Thus, it appears the $ES$-attribute correlates with the probability for a set with a great number of solutions. Therefore, despite the fact that the $ES$-attribute is unable to predict the number of solutions of a single rectangle-set accurately, it is capable of providing an expected average for a considerable amount of sets. Moreover, a low $ES$ is a clear indication of a rectangle set with a small number of solutions.

Since the $ES$ appears to correlate with the average number of solutions, it seems as if a high $ES$ is a requirement for a substantial number of solutions. Thus, it seems the $ES$ significantly affects the difficulty of a PRPP. However, there seem to be more factors affecting the difficulty as the $ES$ is unable to provide a lower boundary. Nonetheless, the effect of $ES$ is considerably great for containers with a large ratio, sets with a great number of rectangles, and low $ES$ values. As a result, the low $ES$ values can reliably predict hard instances of PRPPs.

Despite the fact that the lack of a lower bound seems unfortunate, it is important to consider the unlikelihood of such a correlation. Since determining whether a PRPP is solvable is an NP-problem, finding a way to predict a minimum number of solutions without solving the NP-problem seems infeasible. This is further noticeable in the scatter plot in figure 2 as each value for $ES$ has instances with a single solution. Accordingly, it is improbable that any method is able to predict the difficulty of a PRPP accurately, however, it would appear that the $ES$ is able to predict the probability of a hard PRPP. Thus, while unable to predict the difficulty of a PRPP, the results of this study suggest it is capable of predicting the chance that a PRPP is difficult to solve.

Consequently, in practice applications could be adapted accordingly to produce sets with a high $ES$, for instance, modules for integrated circuit systems could be designed to have more sides in common,
resulting in a greater $ES$. Similarly, logistic companies could pack their goods accordingly, for instance when selecting what goods to pack together, the $ES$ values of the resulting groups could be maximised.

6 Future work

Although, the results of this study suggest $ES$ can be used to calculate the probability of a hard PRPP, the investigation was limited to solvable PRPPs. Future research could evaluate this predictive ability for both solvable and unsolvable PRPPs. Moreover, this could be extended to research whether the solvability of a rectangle set can be predicted. Intuitively, a high probability for easy PRPPs should indicate similarly high probability for solvable PRPPs. However, the distribution of unsolvable PRPPs could contradict this intuition, furthermore, it can be a difficult task to verify as initial experiments showed the probability for a solvable set is around $10^{-4}$ for a cardinality of 12 and a surface of 1200.

Besides the $ES$-attribute several other attributes which can be calculated in linear time seem related to the difficulty of PRPPs. Initial experiments suggested the average ratio of the rectangles in the set, and the ratio of the container affect the difficulty of a PRPP. Interestingly, non-linear functions based on the knapsack-problem may be able to predict this difficulty as well, since Rectangle Packing is closely related to the knapsack problem. However, despite its potential predictive abilities, the computing-time required may negate its prediction quality.

Although, for this study the PRPPs in the dataset were selected as generally as possible, it is reasonable to assume that most practical areas do not encounter such a universal variety of rectangle-sets. For instance, a module with dimensions 900x1 will seldom occur in integrated circuit systems, whereas this study regarded its frequency similar to a module with dimensions 30x30, despite the fact that the latter is expected to be more common in real-world instances. Therefore, it may prove valuable to determine the distribution of rectangle-ratios in the most prevalent application areas of Rectangle Packing Problems, thus allowing insights into the specific subset of PRPPs relevant to real-world applications to be found.
References


Appendices

A Uniform Generating of Rectangle Sets

For this study the decision was made to only compare sets of unique rectangles and with an equal number of rectangles, unique meaning a rectangle may not appear more than once in a set. Additionally, all comparisons only included sets with the same total surface. Thus disabling these factors from influencing the results. Furthermore, since the total number of rectangle-sets for any significant size and surface is tremendous, a selection had to be made. For a fair analysis, this limited selection has to be representative of the entirety of possible sets.

In order to achieve such uniformity, a generating algorithm was designed for which each set that satisfies the constraints has an equal probability to be generated. This algorithm consists two parts: the initialisation part which maps the set of potential rectangle-sets, and the generating part which creates a rectangle-set in accordance with the previous mapping.

A.1 Computing the Number of Potential Sets

The mapping of the potential sets is created by calculating the number of sets with a given size that could potentially be configured to make a rectangle of a given surface. Let $\psi$ be the given surface and $n$ be the size of each set. Then let $R_{\psi}$ be the set of all rectangles that have a surface less or equal to $\psi$:

$$R_{\psi} = \{ w, h \in \mathbb{N} | w \leq h \land 0 < w \times h \leq \psi \}$$ (3)

The set of potential sets $S_{n,\psi}$ with size $n$ will then be:

$$S_{n,\psi} = \left\{ s \mid \forall w, h \in s[w, h \in R_{\psi}] \land \sum_{w, h \in s} w \times h = \psi \land |s| = n \right\}$$ (4)

Primarily, the unoptimised algorithm employs two procedures, POTSREC(size,max,surface) and RECTANGLESOFSURFACE(surface). Firstly, let the output of POTSREC(size,max,surface) be equal to $p(n, \mu, \psi)$, with $n$ as the set-size, $\mu$ as the upper boundary of rectangle surfaces, and the total surface $\psi$:

$$p(n, \mu, \psi) = |\{ s \in S_{n,\psi} \mid \forall (w, h) \in s[w \times h \leq \mu] \}|$$ (5)

Secondly, let the output of RECTANGLESOFSURFACE(surface) be $r(\mu)$, which is the number of rectangles in $R$ with an equal surface to $\mu$:

$$r(\mu) = |\{ (w, h) \in R_{\mu} \mid w \times h = \mu \}|$$ (6)
For any \( s \in S_{n,\psi} \), let \( \mu_s \) be equal to the surface of the largest rectangle, thus:

\[
\exists (w, h) \in s \left[w \times h = \mu_s \right] \land \forall (w, h) \in s \left[w \times h \leq \mu_s \right]
\]

(7)

By the definition of \( S_{n,\psi} \):

\[
s \in S_{n,\psi} \rightarrow \mu_s \leq \psi
\]

(8)

Thus, by the definition of \( p(n, \mu, \psi) \):

\[
p(n, \psi, \psi) = |S_{n,\psi}|
\]

(9)

Let \( S_{n,\psi,\mu} \) be:

\[
S_{n,\psi,\mu} = \{ s \in S_{n,\psi} \mid \mu_s = \mu \}
\]

(10)

Since \( \mu_s \) can be interpreted as an injective function \( s \rightarrow \mu_s \):

\[
(\mu_0 \neq \mu_1 \lor n_0 \neq n_1 \lor \psi_0 \neq \psi_1) \rightarrow S_{n_0,\psi_0,\mu_0} \cap S_{n_1,\psi_1,\mu_1} = \emptyset
\]

(11)

Consequently, it can be used to compute the cardinality of \( |S_{n,\psi}| \):

\[
p(n, \mu, \psi) = |\{ s \in S_{n,\psi} \mid \mu_s \leq \mu \}| = |\bigcup_{i=1}^{\mu} S_{n,\psi,i}| = \sum_{i=1}^{\mu} |S_{n,\psi,i}|
\]

(12)

Let \( M_\mu \) be:

\[
M_\mu = \{ m \mid \forall w, h \in m \left[w, h \in R_\mu \land w \times h = \mu \right] \land m \neq \emptyset \}
\]

(13)

Then for every \( s \in S_{n,\psi} \) there is a set \( m_s \):

\[
m_s = \{ w, h \in s \mid w \times h = \mu_s \}
\]

(14)

By the definition of \( m_s \) and \( M_\mu \):

\[
m_s \in M_{\mu_s}
\]

(15)

Since \( \mu_s \) is injective so is \( m_s : s \rightarrow m_s \). Furthermore, for every \( s \in S_{n,\psi} \) there is a set \( c_s, s \rightarrow c_s \), so that:

\[
c_s \cup m_s = s \land c_s \cap m_s = \emptyset
\]

(16)

Thus:

\[
c_s = s / m_s
\]

(17)
Therefore, \(|c_s| = n - |m_s|\), \(\psi_{c_s} = \psi_s - \psi_{m_s}\) and \(\mu_{c_s} < \mu_s\). Accordingly, the union between any \(m \in M_\mu\) and any set \(c \in \bigcup_{i=1}^{\mu - 1} S_{n-|m|,\psi_s-\psi_{m,i}}\) is an element of \(S_{n,\psi}\):

\[
m \in M_\mu \rightarrow c \in \bigcup_{i=1}^{\mu - 1} S_{n-|m|,\psi_s-\psi_{m,i}} \rightarrow (s = m \cup c \wedge |s| = n \wedge \psi_s = \psi \wedge \mu_s = \mu) \tag{18}
\]

Thus, by the definition of \(S_{n,\psi,\mu}\):

\[
m \in M_\mu \rightarrow c \in \bigcup_{i=1}^{\mu - 1} S_{n-|m|,\psi_s-\psi_{m,i}} \rightarrow (m \cup c \in S_{n,\psi,\mu}) \tag{19}
\]

Similarly, the following is true:

\[
s \in S_{n,\psi,\mu} \rightarrow m_s \in M_\mu \wedge c_s \cup m_s \in S_{n,\psi} \tag{20}
\]

Let \(M_{n,\mu}\) be:

\[
M_{n,\mu} = \{ m \in M_\mu \mid |m| = n \} \tag{21}
\]

Since, \(M_{n,\mu}\) is the set of all subsets of the set of all rectangles with surface \(\mu\) with cardinality \(n\), its cardinality can be calculated with the binomial coefficient. Moreover, using the procedure \textsc{RectanglesOfSurface}(\textit{surface}), the cardinality of this set can be computed:

\[
|M_{n,\mu}| = \left( \frac{|\{ w, h \in R_\mu \mid w \times h = \mu \}|}{n} \right) = \left( \frac{r(\mu)}{n} \right) \tag{22}
\]

Furthermore, according to equations 19 and 20 each set in \(S_{n,\psi,\mu}\) can be split up in an \(m_s\) and \(c_s\) part, thus:

\[
|S_{n,\psi,\mu}| = \left| \left\{ m \cup c \middle| m \in M_\mu \wedge c \in \bigcup_{i=1}^{\mu - 1} S_{n-|m|,\psi_s-\psi_{m,i}} \right\} \right| = \left| \bigcup_{j=1}^{n} \left\{ m \cup c \mid m \in M_{\mu,j} \wedge c \in \bigcup_{i=1}^{\mu - 1} S_{n-|m|,\psi_s-\psi_{m,i}} \right\} \right| = \left| \bigcup_{j=1}^{n} \left\{ m \cup c \mid m \in M_{\mu,j} \wedge c \in \bigcup_{i=1}^{\mu - 1} S_{n-|m|,\psi_s-\psi_{m,i}} \right\} \right| = \sum_{j=1}^{n} [M_{\mu,j} \times \bigcup_{i=1}^{\mu - 1} S_{n-|m|,\psi_s-\psi_{m,i}}] \tag{23}
\]
From combining this with equation 22, the following can be inferred:

\[ |S_{n,\psi,\mu}| = \sum_{j=1}^{n} \left( r(\mu) \right) \times |\bigcup_{i=1}^{\mu-1} S_{n-j,\psi-\mu \times j,i}| \]  

(24)

The disjunction of the sets \( S_{n-j,\psi-\mu \times j,i} \) as derived from 11 leads to:

\[ |S_{n,\psi,\mu}| = \sum_{j=1}^{n} \left( r(\mu) \right) \times \sum_{i=1}^{\mu-1} |S_{n-j,\psi-\mu \times j,i}| \]  

(25)

Hence, equation 12 can be extended to:

\[ p(n, \mu, \psi) = \sum_{l=1}^{\mu} \sum_{j=1}^{n} \left( r(l) \right) \times \sum_{i=1}^{l-1} |S_{n-j,\psi-l \times j,i}| \]  

(26)

Because this can be split up in three cases:

\[
p(n, \mu, \psi) = \begin{cases} 
0 & \text{if } \psi_i < 0 \lor \mu_i < 0 \lor n_i < 0 \\
1 & \text{if } \psi_i = 0 \land n_i = 0 \\
\sum_{l=1}^{\mu} \sum_{j=1}^{n} \left( r(l) \right) \times \sum_{i=1}^{l-1} |S_{n-j,\psi-l \times j,i}| & \text{else}
\end{cases}
\]  

(27)

It is possible to be rewritten recursively, using 12 again it can be split up in three cases:

\[
p(n, \mu, \psi) = \begin{cases} 
0 & \text{if } \psi_i < 0 \lor \mu_i < 0 \lor n_i < 0 \\
1 & \text{if } \psi_i = 0 \land n_i = 0 \\
\sum_{l=1}^{\mu} \sum_{j=1}^{n} \left( r(l) \right) \times p(n-j, l-1, \psi-l \times j) & \text{else}
\end{cases}
\]  

(28)

Finally, since:

\[
p(n, \mu - 1, \psi) = \sum_{l=1}^{\mu-1} \sum_{j=1}^{n} \left( r(l) \right) \times p(n-j, l-1, \psi-l \times j) \]  

(29)

The \( \sum_{l=1}^{\mu} \) can be removed by changing the limits of \( \sum_{j=1}^{n} \) to \( \sum_{j=0}^{n} \).

\[
p(n, \mu, \psi) = \begin{cases} 
0 & \text{if } \psi_i < 0 \lor \mu_i < 0 \lor n_i < 0 \\
1 & \text{if } \psi_i = 0 \land n_i = 0 \\
\sum_{j=0}^{n} \left( r(l) \right) \times p(n-j, l-1, \psi-l \times j) & \text{else}
\end{cases}
\]  

(30)
Consequently, the cardinality of any $S_{n,\psi}$ or $S_{n,\psi,\mu}$ can be computed with merely the procedures $\text{POTSREC}(\text{size}, \text{max}, \text{surface})$ and $\text{RECTANGLESOFSURFACE}(\text{surface})$. Moreover, the recursive nature of this definition allows for heavy optimisation, resulting in a short execution time. The usage of optimisations are encouraged, as the computing of $S_{n,\psi}$ is not a feasible in a sizable period of time, even for significantly small values for $n$ and $\psi$. For instance, for $n = 12$ and $\psi = 1200$ the number of sets is $|S_{12,1200}| = 34991904119803745785084$, and because the unoptimised algorithm has more function calls than $|S_{n,\psi}|$, the algorithm is not expected to finish within a lifetime. Thus, the use of the unoptimised algorithm is inadvisable, however, for readability it is displayed in algorithm 1.

Nonetheless, after applying several optimisations the computation of $|S_{12,1200}|$ can finish in a matter of seconds. Mainly, this speedup is gained by caching function calls of the $\text{POTSREC}(\text{size}, \text{max}, \text{surface})$ procedure. This dynamic-programming approach makes use of the large overlap potential sets have, and consequently achieves an exponential speedup. In addition, minor optimisations are performed by adding lookup tables for the binomial exponent, and the $\text{RECTANGLESOFSURFACE}(\text{surface})$ procedure, a great amount of runtime computation is replaced with a faster array operation. Lastly, by calculating the minimum surface for every $0 < i \leq n$ numerous branches can be pruned. For, if the minimum surface of a set of 11 unique rectangles is $\phi$, the largest rectangle in a set of 12 has a maximum surface of $\psi - \phi$. The algorithm with all these optimisations applied is displayed in 2.

A.2 Generating a Random Set Uniformly

Let $g_{n,\psi}$ be the result of a generating procedure where $g_{n,\psi} \in S_{n,\psi}$, then for uniform generating the probability for each set must be equal:

$$s \in S_{n,\psi} \rightarrow P(g_{n,\psi} = s) = \frac{1}{|S_{n,\psi}|} \quad (31)$$

For this study the approach was taken to implement the generating procedure in a recursive manner. Initially, this procedure is called with an empty set as its parameter, subsequently each intermediate step expands the set, culminating in an element of $S_{n,\psi}$. Firstly, let $\phi_s$ be the smallest rectangle-surface in set $s$:

$$\phi_s = w \times h \text{ where } w, h \in s \land \forall w_i, h_i \in s[w_i \times h_i \geq w \times h]$$

Secondly, let $g_{n,\psi}(s_i) = s_{i+1}$ be the event where generating procedure $g_{n,\psi}(s_i)$ generates rectangle-set $s_{i+1}$ from the rectangle-set $s_i$:

$$g_{n,\psi}(s_i) = s_{i+1} \rightarrow |s_i| = n \rightarrow (s_i = s_{i+1} \land s_i \in S_{n,\psi}) \quad (32)$$

$$g_{n,\psi}(s_i) = s_{i+1} \rightarrow |s_i| \leq n \rightarrow (s_i \subset s_{i+1} \land \phi_{s_{i+1}} < \phi_s \land \forall w, h \in s_{i+1} \setminus s_i[w \times h = \phi_{s_{i+1}}]) \quad (33)$$
As such, the rectangles which are in $s_{i+1}$ but not in $s_i$ have an equal surface, and have a smaller surface than any of the rectangles in $s_i$. Hence, generating a set using $g_{n,\psi}$ is done by generating rectangles in descending order of surface. Furthermore, let $C_{n,\phi,s_i}$ be all the sets in $S_{n,\psi}$ which can be generated using $g_{n,\psi}(s_i)$ repeatedly:

$$C_{n,\psi,s_i} = \{ s \in S_{n,\psi} \mid s \subseteq s_i \land \psi_{s_i \setminus s} < \phi_{s_i} \}$$  \hspace{1cm} (34)

Then, for uniformity the probability of $g_{n,\psi}(s_i) = s_{i+1}$ is:

$$P(g_{n,\psi}(s_i) = s_{i+1}) = \frac{\sum_{s \in C_{n,\psi,s_{i+1}}} P(g_{n,\psi} = s)}{\sum_{s \in C_{n,\psi,s_i}} P(g_{n,\psi} = s)}$$  \hspace{1cm} (35)

From which the following derivation can be made:

$$P(g_{n,\psi}(s_i) = s_{i+1}) = \frac{\sum_{s \in C_{n,\psi,s_{i+1}}} P(g_{n,\psi} = s)}{\sum_{s \in C_{n,\psi,s_i}} P(g_{n,\psi} = s)} = \frac{\sum_{s \in C_{n,\psi,s_{i+1}}} \frac{1}{|S_{n,\psi}|}}{\sum_{s \in C_{n,\psi,s_i}} \frac{1}{|S_{n,\psi}|}} = \frac{|C_{n,\psi,s_{i+1}}|}{|C_{n,\psi,s_i}|}$$  \hspace{1cm} (36)

As a result, the procedure `POTSETREC(size,max.surface)` is now applicable as:

$$|C_{n,\psi,s}| = \begin{cases} 
  p(n,\psi,\psi) & \text{if } s = \emptyset \\
  p(n - |s|,\phi_{s},\psi - \psi_{s}) & \text{else}
\end{cases}$$  \hspace{1cm} (37)

Therefore, the probability of $g_{n,\psi}(s_i) \rightarrow s_{i+1}$ is equal to:

$$P(g_{n,\psi}(s_i) = s_{i+1}) = \begin{cases} 
  \frac{p(n - |s_i+1|,\phi_{s_{i+1}},\psi - \psi_{s_{i+1}})}{p(n,\psi,\psi)} & \text{if } s = \emptyset \\
  \frac{p(n - |s_i|,\phi_{s},\psi - \psi_{s})}{p(n - |s|,\phi_{s},\psi - \psi_{s})} & \text{else}
\end{cases}$$  \hspace{1cm} (38)

Accordingly, the probability of each possible outcome of $g_{n,\psi}(s_i)$ can be computed. Hence, the computed distribution can be combined with a standard pseudo-random number generator resulting in a module which generates rectangles sets of $S_{n,\psi}$ uniformly.
Algorithm 1 Unoptimised algorithm for computing $|S_{n,\psi}|$ and $|S_{n,\psi,\mu}|$.

Require: $size > 0, surface > 0$
Ensure: Total number of sets consisting of $size$ unique rectangles with a given total $surface$

1: procedure POTENTIALSETS($size, surface$)
2: \hspace{1em} return POTSETSREC($size, surface, surface$)
3: end procedure

Require: $size >= 0$
Ensure: Total number of sets of $size$ consisting of unique rectangles and with a given total $surface$, and with each rectangle having a surface equal or less than $max$

4: procedure POTSETSREC($size, max, surface$)
5: \hspace{1em} if $size = 0$ and $surface = 0$ then
6: \hspace{2em} return 1
7: \hspace{1em} else if $surface <= 0$ then
8: \hspace{2em} return 0
9: \hspace{1em} else if $size \times max < surface$ then
10: \hspace{2em} return 0
11: \hspace{1em} end if
12: \hspace{1em} sum $\leftarrow 0$
13: \hspace{1em} options $\leftarrow$ RECTANGLESOFSURFACE($max$)
14: \hspace{1em} for $i \leftarrow 0, min(size, options)$ do
15: \hspace{2em} sum $\leftarrow sum + (options_i)^{opt} \times$ POTSETSREC($size - i, max - 1, surface - i \times max$)
16: \hspace{1em} end for
17: \hspace{1em} return sum
18: end procedure

Require: $surface > 0$
Ensure: Total number of rectangles with given $surface$

19: procedure RECTANGLESOFSURFACE($surface$)
20: \hspace{1em} max $\leftarrow \lfloor \sqrt{surface} \rfloor$
21: \hspace{1em} sum $\leftarrow 1$
22: \hspace{1em} for $i \leftarrow 2, max$ do
23: \hspace{2em} if $surface \mod i = 0$ then
24: \hspace{3em} sum $\leftarrow sum + 1$
25: \hspace{2em} end if
26: \hspace{1em} end for
27: \hspace{1em} return sum
28: end procedure
Algorithm 2 Optimised algorithm for computing $|S_{n,\Psi}|$ and $|S_{n,\Psi,\mu}|$.

Require: $size > 0$, $surface > 0$

Ensure: Total number of sets consisting of size unique rectangles with a given total surface

1: procedure POTENTIALSETS($size$, $surface$)
2:     $recSurf \leftarrow [\ ]$ \hspace{1cm} \triangleright Integer array of length $surface + 1$.
3:     for $i \leftarrow 0, surface$ do
4:         $recSurf[i] \leftarrow \text{RECTANGLESOFSURFACE}(i)$ \hspace{1cm} \triangleright As defined in 1.
5:     end for
6:     $maxRec \leftarrow \max(recSurf)$
7:     $binom \leftarrow [\ ][\ ]$ \hspace{1cm} \triangleright 2d integer array of length $size$ and $maxRec$.
8:     for $i \leftarrow 0, maxRec$ do
9:         for $j \leftarrow 0, size$ do
10:            $binom[i][j] \leftarrow \binom{i}{j}$
11:        end for
12:     end for
13:     \hspace{1cm} \triangleright Fill array with the minimum surfaces of sets with sizes from 1 to $size$.
14:     $minSurface \leftarrow [\ ]$ \hspace{1cm} \triangleright Integer array of length $size + 1$
15:     $surIndex \leftarrow 1$
16:     $total \leftarrow 0$
17:     $minIndex \leftarrow 0$
18:     while $recIndex \leq size \land minIndex < size$ do
19:         $tempCount \leftarrow recSurf[surIndex]$
20:         while $tempCount > 0 \land minIndex < size$ do
21:             $total \leftarrow total + surIndex$
22:             $minIndex \leftarrow minIndex + 1$
23:             $minSurface[minIndex] \leftarrow total$
24:             $tempCount \leftarrow tempCount - 1$
25:         end while
26:         $surIndex \leftarrow surIndex + 1$
27:     end while
28:     $cache \leftarrow \{\}$ \hspace{1cm} \triangleright Hashmap[String] $\rightarrow$ Integer
29:     return POTSETSREC($size$, $surface$, $surface$)
30: end procedure
Require: \( size \geq 0 \)

Ensure: Total number of sets of \( size \) consisting of unique rectangles and with a given total \( surface \), and with each rectangle having a surface equal or less than \( max \)

31: procedure \( \text{POTSREC}(size, max, surface) \)
32: \hspace{1em} if \( size = 0 \) and \( surface = 0 \) then
33: \hspace{2em} return 1
34: \hspace{1em} else if \( surface \leq 0 \) then
35: \hspace{2em} return 0
36: \hspace{1em} else if \( size \times max < surface \) then
37: \hspace{2em} return 0
38: \hspace{1em} else if \( \text{minSurface}[size] > surface \) then
39: \hspace{2em} return \( \text{cache}[size, max, surface] \)
40: \hspace{1em} end if
41: \hspace{1em} sum ← 0
42: \hspace{1em} options ← \( \text{recSurf}[max] \)
43: \hspace{1em} for \( i ← 0, \min(options, size) \) do
44: \hspace{2em} sum ← sum + \( \text{binom}[options][i] \times \text{POTSREC}(size - i, max - 1, surface - i \times max) \)
45: \hspace{1em} end for
46: \hspace{1em} cache[size, max, surface] ← sum
47: \hspace{1em} return sum
48: end procedure