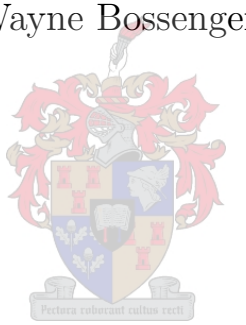
2D Irregular Strip Packing at Kohler Signs

Wayne Bossenger



Thesis presented in fulfilment of the requirements for the degree of
Master of Commerce
in the Faculty of Economic and Management Sciences at Stellenbosch University

Supervisor: Dr Isabelle Nieuwoudt

December 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 1, 2014

Abstract

Kohler Signs (PTY) Ltd is a sign production company located in Cape Town, South Africa. They manufacture and install signs for the City of Cape Town and private companies as well as manufacture advertisement signs to be placed on vehicles. Road signs consist of steel sheets that are cut and bent to the appropriate size and frame, and an image design, which is cut from reflective vinyl, are applied to the bent steel sheet. The image design consists of various letters, numbers and symbols which are categorised as *irregular items*. When these irregular items are combined in a distinctive way, with the use of different coloured vinyl, they convey a message to the road user which may be to yield for pedestrians crossing the street, or indicate to the road user the various highway exits that exist on the interchange ahead. These irregular items are placed upon reflective vinyl for cutting which results in vinyl offcuts that are wasted. The focus of this thesis is to minimise the waste incurred by placing these irregular items upon the vinyl in an optimal and timely manner for industry use. The vinyl printer, which cuts the irregular items out of the vinyl, consists of a fixed width and is only limited in height by the vinyl itself. Thus, this problem may be described as a Two Dimensional Irregular Strip Packing Problem. These irregular items have only a few possible heights for each type of irregular item packed, which allows these irregular items to be packed as a level packing problem. The items are packed within levels as though they are regular items with the assistance of a predefined rule-set. In this thesis various packing algorithms and image processing methodologies from the literature are researched and used to develop a new packing algorithm for this specific problem. The newly developed algorithm is put through various benchmarks to test its performance. Some of these benchmarks are procured from Kohler Signs themselves, whereas others are randomly generated under certain conditions. These benchmarks reveal that the newly developed algorithm performs better for both the minimisation of waste and the minimisation of algorithm running time than the tried and trusted techniques utilised in industry by Kohler Signs.

Opsomming

Kohler Signs (EDMS) Bpk is 'n padteken produksie maatskappy geleë in Kaapstad, Suid-Afrika. Hulle vervaardig en installeer tekens vir die Stad van Kaapstad en privaat maatskappye, sowel as advertensietekens wat op voertuie geplaas word. Padtekens bestaan uit staalplate wat gesny en gebuig word tot die toepaslike grootte en vorm. 'n Beeldontwerp, wat gesny is uit reflektiewe viniel, word vasgesit op die gebuigde staalplaat. Die beeldontwerp bestaan uit verskeie letters, getalle en simbole wat geklassifiseer word as *onreëlmatige items*. Wanneer hierdie onreëlmatige items gekombineer word op 'n eiesoortige manier, met die gebruik van verskillende kleure viniel, dra hulle 'n boodskap oor aan die padgebruiker, soos byvoorbeeld om toe te gee aan voetgangers by 'n voetoorgang of dit dui aan die padgebruiker die verskillende snelweguitgange wat bestaan op die wisselaar wat voorlê. Hierdie onreëlmatige items word op reflektiewe viniel geplaas en uitgesny wat lei tot die vermorsing van stukkies viniel. Die fokus van hierdie tesis is om die onreëlmatige items op 'n optimale en tydige wyse vir gebruik in industrie, op die viniel te plaas sodat die afval stukkies viniel geminimeer word. Die vinieldrukker, wat die onreëlmatige items sny uit die viniel, bestaan uit 'n vaste wydte en is slegs beperk in hoogte deur die viniel self. Dus kan hierdie probleem beskryf word as 'n Twee-Dimensionele Onreëlmatige Strookverpakkingsprobleem. Hierdie onreëlmatige items het slegs 'n paar moontlike hoogtes vir elke tipe van onreëlmatige item wat verpak word, wat dit moontlik maak om hierdie onreëlmatige items te verpak as 'n strook verpakkingsprobleem. Die items word met behulp van 'n gedefinieerde stel reëls binne vlakke verpak asof hulle reëlmatige items is. In hierdie tesis is verskeie verpakkingsalgoritmes en beeldverwerkingsmetodes van die literatuur nagevors en gebruik om 'n nuwe verpakkingsalgoritme vir hierdie spesifieke probleem te ontwikkel. Die nuut ontwikkelde algoritme se prestasie is deur middel van verskeie normbepalingsvoorbeelde getoets. Sommige van hierdie normbepalingsvoorbeelde is verkry van Kohler Signs self, terwyl ander lukraak gegenereer is onder sekere voorwaardes. Hierdie normbepalingsvoorbeelde toon dat die nuut ontwikkelde algoritme beter vaar as die beproefde tegnieke gebruik in industrie deur Kohler Signs vir beide die minimering van vermorsde viniel sowel as die minimering van die algoritme se uitvoertyd.

Acknowledgements

The author wishes to acknowledge a number of people and institutions for their various contributions towards the completion of this work.

- the Department of Logistics, for the use of their study space and facilities;
- Kohler Signs for allowing me to study their business, with special mention to Russell Kohler, my contact to whom I sent many back and forth emails and chats;
- Dr Isabelle Nieuwoudt, my supervisor, for her guidance, patience and friendship throughout this pursuit of knowledge;
- To the fellowship of the GOReLAB, with special mention to Jacques du Toit and Antoinette Erasmus, for the many hours of assistance, brainstorming, Neelsie runs and tea times shared;
- Last but not least to my family and friends, for their unyielding support and belief in me throughout my years of study.

Table of Contents

Glossary

1D [2D, 3D respectively] packing problems One [Two, Three respectively] dimension is needed to compare items to be packed and placed within objects.

Adjacency region The common boundary between an items perimeter and the edges of the already placed items, the object or bounding rectangle.

Beaded vinyl Sheets of vinyl that consists of small hexagonal shapes tightly packed together.

Bin A packing object with all dimensions bounded.

Bin Packing Items are packed into many objects called bins where all dimensions are fixed.

Block Shape Items that consist of parallel vertical lines on either side.

Bottom set Set of items that have free space at the bottom and not at the top.

Bounding rectangle A rectangle that envelops a set of packed items as tight as possible.

Bubble sort A sorting algorithm that steps through each item in the sorting list in order, and swaps adjacent items if they are in the wrong order.

Candidate initial locations All possible unique placement locations within a packing space for an item in a packing algorithm that does not overlap.

Cleaning threshold value The parameter used for converting a dirty pixel to either black or white, dependant on whether the dirty pixel value is lower (black) or higher (white) than this parameter value respectively.

Collective waste The sum of the space wasted between already packed items, per level.

Combination signs Road signs that combine two or more groups of signs to make one sign.

Command signs Regulatory signs convey the rules of the road that the road user is allowed to follow.

Comprehensive signs Regulatory signs which illustrate the exact nature of the road.

Contour The curve along the outer edge of an image.

Control signs Regulatory signs that instruct the road user how to use the road.

Convex only polygon If a straight line segment is drawn between any two vertices in the polygon, the line segment will never touch outside the polygon.

Convex concave polygon Polygons that consist of both convex and concave angled polygons so that if a straight line segment is drawn between two vertices in the polygon, the line segment *may* exist outside the polygon.

Cyclic shape item The interior angles of the item are equal.

De-restriction signs Regulatory signs that indicate to the road user that the rule shown is not allowed.

Diagrammatic signs Guidance signs that inform the road user of the changes to the road further down and other changes in the traffic conditions.

Direction of movement signs Warning signs that illustrates to the road user how the upcoming road acts.

Direction signs Guidance signs that give information to the road user that will help them find their way.

Dirty pixels Extra pixels in an image that do not form part of the image itself and as such should be removed.

End space waste The space at the end of a level of packed items except the last level, that is too small for the next item to be packed.

Equilateral items Items with which all the opposite sides have equal side length.

Free-form item An item formed when vertices are joined together by curved lines.

Free space The last level's end space waste when the list of items to be packed at this point in time is zero. This space might be available for the next list of items to be packed.

Freeway direction signs Direction signs found on freeways.

Guidance signs Road signs that convey information to the road user that will guide them.

Guillotineable packing The packing of the items always results in such a way that any straight cut does not intersect any item.

Hazard marker signs Warning signs that warn the road user of the impending danger ahead.

Height dimension The vertical dimension of an item.

Height of a strip object The unbounded vertical dimension of the container object.

Heuristics Strategies for solving optimization problems approximately by constructing "good", but not necessarily optimal solutions at a reasonable computational cost.

Information signs Road signs that indicate to the road user relevant information regarding the use of the road.

Interpolation Technique whereby new data points are constructed by estimating the path of known data points.

Irregular item The shape of the item is not equilateral nor cyclic and may have varying side length and angle degree.

Item Object that must be packed.

Letter point The dot part of a letter like the letter "i" that is not joined to the main letter section.

Letter stroke The main letter shape without the letter point.

Level A horizontal rectangular band parallel to the floor of a strip with length equal to the width of the strip.

Location signs Guidance sign that conveys to the road user a location on the road.

Minimum enclosing rectangle A rectangle that is enclosed with items with its area minimised.

Minimum residual horizontal space The smallest free space available, for the item to be packed into, from all levels.

No Fit Polygon Arrangement that two arbitrary polygons may take such that the polygons touch but do not overlap.

Non-guillotineable packing Any straight cut through such a packing may result in the intersection of one or more items, while at least one straight cut will result in the intersection of an item or items.

Object A containment area within which items are packed.

Off-line problems Problems where the dimensions of the entire set of items to be packed are known in advance.

On-line problems Only the dimensions of the item that is next in the set to be packed is known, and not any other item in the set after that.

Orbiting polygon Polygon that has multiple possible initial locations for placement by rotating the polygon.

Overhead signs A road sign erected above the road user, whereby the road ahead is illustrated and captioned with their names.

Packing Problems Optimization problems in which a good placement of multiple items in larger containing areas which are considered as objects, is sought.

Polygon item Formed by many vertices that are joined together by edges.

Polygon thinning A technique used to remove redundant points from the edge of an image.

Prismatic vinyl Vinyl that consists of many lines that go horizontally across the roll which helps with the reflective process and can be seen when observed at close range.

Prohibition signs Regulatory signs that convey the rules that prohibit certain actions allowed on the road.

Rectangular module The enclosed items together with the rectangular shape enclosing them which will then be packed as a single item.

Regular item The shape of the item is equilateral and cyclic.

Regulatory signs Road signs that regulate the rules of the road to the road user.

Reservation signs Regulatory signs that reserve the road for specific use.

Road layout signs Warning signs that illustrate how the road ahead changes.

Rotated item Item that has been shifted clockwise on its own axis to a certain degree.

Route markers Guidance signs that indicate to the road user the locations that are coming up on the route.

Rule-set or Rule-set Technique Items of similar shape are grouped together and packed.

Rule-set value A weight value given to each shape in the rule-set when compared to other shapes placed adjacent to it on either side.

Selective restrictive signs Regulatory signs restricting the road to specific times of the day.

Street name signs Signs that convey to the road user the name of the road.

Shape group Items that share similar characteristics are grouped together.

Strip A packing object with one bounded dimension (and all others unbounded).

Strip Packing Items are packed into an object called a strip and this object has one dimension fixed and the other dimension(s) unbounded.

Symbolic signs Warning signs that convey to the road user that the symbol shown in the sign is coming up.

Text height The text height indicates the maximum height that any item from the particular font may be.

Top set Set of items that have free space at the top and not at the bottom.

Temporary signs Road signs that are installed on roads for temporary use. They have a yellow background instead of the default colour of the sign they represent.

Variable message signs Guidance signs that variate the message they show.

Vinyl Sheet substance containing a normal colour layer and a reflective layer used in the production of road signs.

Warning signs Road signs that convey to the road user of the upcoming hazard that they might not be aware of.

Waste The non utilised pieces of material or space of a packing algorithm after items have been cut from the material.

Width dimension The horizontal dimension of an item.

Yield signs Cautions the road user to pause at the intersection.

List of Acronyms

1D: One-Dimensional

2D: Two-Dimensional

3D: Three-Dimensional

BFDH: Best Fit Decreasing Height

BL: Bottom Left

CA: Constructive Approach

CAA: Constructive Approach (minimum area)

CAD: Constructive Approach (maximum adjacency)

CW: Collective Waste (per level)

ESW: End Space Waste

FFDH: First Fit Decreasing Height

FS: Free Space

NFDH: Next Fit Decreasing Height

NFP: No Fit Polygon

KS: Kohler Signs

PP: Packing Problem

List of Figures

List of Tables

CHAPTER 1

Introduction

Contents

In the road sign production industry there are many different types of signs constructed, from traffic signs to signs constructed for private companies to large billboards. All these signs share one common property in that they all convey a message to the viewer via the medium of text and images. Using the *Stop* sign, given in Figure 1.1(a), as an example, this traffic sign instructs the drivers of vehicles to stop before proceeding further down the road.

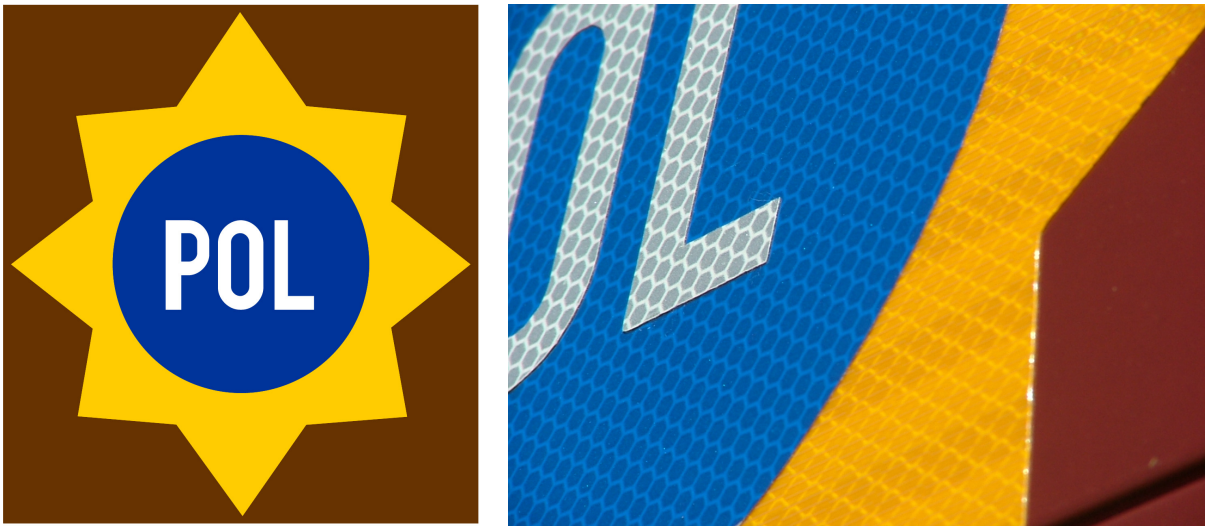
(a) *Example of a Stop sign* (b) *Example of a Warning sign* (c) *Example of a Yield sign*

FIGURE 1.1: *Three examples of different road signs.*

Signs consist of different sizes, shapes, and colours and they contain different texts and images. For example, the stop sign is shaped like a regular octagon, whereas a *Warning* sign is shaped like an equilateral triangle and a *Yield* sign is shaped like an equilateral triangle that has been rotated on its axis by 180 degrees. These three signs can be seen in Figure 1.1. A stop sign

has a red background and a white regular octagon shaped border with white text that displays the word "STOP" in the centre of the sign, whereas a warning sign consists of an image on a white background and a yield sign consists of an empty image on a white background. Also, *Street Name signs* only convey the name of the road to the road user with text while a so-called *Overhead sign* consists of the illustration of the road ahead as well as the names of the roads depicted in the illustration. These overhead signs are placed over highways while for example the stop sign is attached to a wooden or metal pole of appropriate height for the road. These are just a few examples of the many different traffic signs that do exist.

There are many processes in manufacturing a single sign. The design of the sign is constructed by cutting different colour pieces of reflective vinyl into specific shapes and placing these vinyl cut-outs onto a steel sheet. Sign designs consist of items which consist of letters, numbers and symbols. The sign on the steel sheet is then attached to one or many wooden or steel poles depending on the specifications for the sign that is made. In Figure 1.2, a closer look is taken at the Police symbol that has been placed on top of a Tourism sign. Each layer of vinyl can be seen very closely.



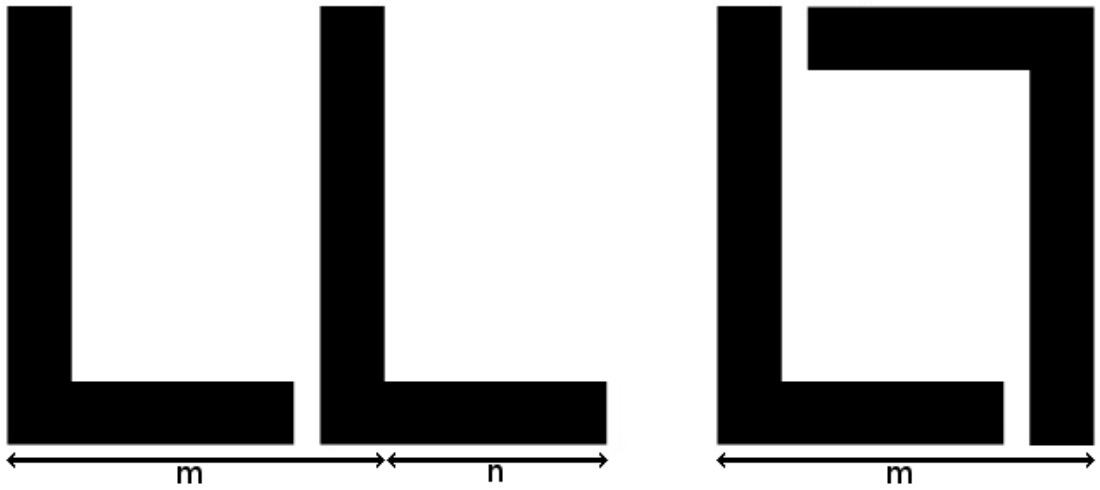
(a) *A Police symbol* (b) *A close look at a Police symbol on a road sign*

FIGURE 1.2: *A Police symbol in (a) and a close look at a Police symbol on a Tourism board where the different layers of the vinyl are visible in (b).*

1.1 Some packing terminology

The items that must be cut out of the vinyl must be placed onto and cut out of the vinyl in some manner. The problem in deciding where to place these items onto the vinyl, is considered as a *Two-Dimensional* (2D) *Packing Problem* (PP). Each item that is to be packed in the 2DPP has a *width* and a *height* dimension. The items are packed adjacent to each other on the roll of vinyl. Once the items are cut out of the vinyl there is excess vinyl left over called *waste*. The aim of the packing problem therefore is that the items are placed on the vinyl in such a manner that this waste should be minimised so as to have more vinyl on the roll left over for other signs. Therefore, in order to minimise this waste, one wants to arrange the different items that need to be cut-out of the vinyl, with as little space between the different items as possible. For example, in Figure 1.3 two different packing strategies are used to pack two letter L's together. The packing in Figure 1.3(a) uses $m + n$ unit lengths of the roll of vinyl, while the packing in

Figure 1.3(b) only uses m unit lengths. The first packing uses a normal packing strategy where the two L's are placed next to each other with a small gap between the letters to allow them to be safely cut out. The second packing is a tight packing of the two L's, where the second L has been rotated by 180 degrees. This second tight packing, causes less waste than the first normal packing.



(a) *Normal packing* (b) *Tight packing*

FIGURE 1.3: *Comparison of two different packings of two letter L's.*

General packing problems are described by certain criteria. These problems are regarded as either *off-line* or *on-line* problems, the items to be packed are described as being *regular* or *irregular* and these items may or may not be *rotated* to some degree. Off-line problems are described as problems where the dimensions of the entire set of items to be packed are known in advance, that is, the width and height of each item in the set that is to be packed is known before the packing of the items has begun. On-line problems are defined as only knowing the dimensions of the item that is *next* in the set to be packed, and not any other item in the set after that. Therefore, only off-line problems are considered in this thesis, since the items that are dealt with in the scenario at hand, are already known before packing. All the information that is needed for the specific sign is known when an order for a sign is received. Regarding the shape of the items to be packed, regular items are *equilateral* and *cyclic*, that is they have equal side length and the interior angles are equal in degree. Irregular items do not conform to these rules and thus have varying side length and angle degree. The items to be packed on the vinyl for traffic signs is clearly irregular. Furthermore, the items to be packed may be rotated by some degree. If an item is rotated by 30 degrees it means that the item rotates on its own axis in a clockwise manner 30 degrees. An item is either rotated or not in order to improve the overall packing, however rotations add another level of complexity to the packing problem.

1.2 Road traffic signs

There are many groups of road traffic signs. These types of road traffic signs are classed as *regulatory signs*, *warning signs*, *information signs*, *guidance signs*, *combination signs* and lastly *temporary signs*. Figure 1.4 shows these different sign groups and their subgroups.

Regulatory signs regulate the rules of the road and consist of *control signs*, *command signs*, *prohibition signs*, *reservation signs*, *comprehensive signs*, *selective restrictive signs* and *de-restriction*

FIGURE 1.4: *List of road traffic signs.*

signs. *Control signs* instruct the road user how to use the road like stopping at an intersection, as seen in Figure 1.1(a), or not allowing entry into the road from the road users location. *Command signs* convey the rules of the road that are allowed to the road user, like allowing the use of bicycles on the road or allowing traffic to turn right at the intersection. *Prohibition signs* convey the rules that prohibit certain actions allowed on the road, like prohibiting the road user in exceeding the speed limit of 100km/h or prohibiting pedestrians in crossing the road. *Reservation signs* reserve the road for specific use, like reserving a road lane for buses only. *Comprehensive signs* illustrate the exact nature of the road like indicating the road is now a highway. *Selective restrictive signs* restrict the road to specific times of the day like only allowing buses to ride in the bus lane between 06:00 and 09:00. *De-restriction signs* indicate to the road user that the rule shown is not allowed like not allowing the use of bright lights on the road. In the background of Figure 1.5 is a regulatory sign that indicates to the road user to yield at the traffic circle ahead, followed by a regulatory sign that indicates to the road user to yield if there are pedestrians crossing the road [18].

Warning signs convey to the road user of the upcoming hazard that the road user might not be aware of and consists of *Road layout signs*, *Direction of movement signs*, *Symbolic signs* and *Hazard marker signs*. *Road layout signs* illustrate how the road ahead changes like when one lane in the road branches off into two lanes. *Direction of movement signs* illustrates how the upcoming road acts, for example an upcoming bend in the road. *Symbolic signs* convey to the road user that the symbol shown in the sign is coming up in the road like the pedestrian crossing warning sign in the foreground of Figure 1.5. *Hazard marker signs* warn the road user of the impending danger ahead like the marker used to indicate that at the bend of the road is a dangerous cliff [9].

Information signs indicate to the road user relevant information regarding the use of the road like the upcoming on-ramp to a highway or upcoming cul-de-sac [3].

Guidance signs convey to the road user information that will guide the road user and consist of *location signs*, *route markers*, *direction signs*, *freeway direction signs*, *overhead freeway direction signs*, *crossroad freeway direction signs*, *toll direction signs*, *local direction signs*, *tourism direction signs*, *diagrammatic signs* and *variable message signs*. *Location signs* convey to the road

FIGURE 1.5: *A warning sign in the foreground and a few regulatory signs in the background.*

user of a location on the road, like the name of the street or the presence of a toll road. *Route markers* indicate to the road user the locations that are coming up on the route like an upcoming highway to the left. *Direction signs* give information to the road user that will help them find their way like the direction sign that indicates to the road user in which direction to travel to Cape Town. *Freeway direction signs* are direction signs found on the freeways, *overhead freeway direction signs* are direction signs found overhead on the freeways and *crossroad freeway direction signs* are direction signs that are found at the intersections where a public road intersects a freeway. Figure 1.6 shows an overhead freeway direction sign over the N1 freeway indicating that in 1.8km there is an intersection with the M5 whereby the left offramp takes the road user north on the M5 and the right offramp takes the road user south on the M5. *Toll direction signs* are direction signs found on toll roads and *local direction signs* are direction signs found within a city that give local directions to the road user like where the nearest shopping mall is. *Tourism direction signs* are direction signs that indicate to the road user which road to take to get to a tourist attraction like a bed and breakfast or national park. *Diagrammatic signs* inform the road user of the changes to the road further down and other changes in the traffic conditions, for example the road changing from allowing three lane traffic on the road to allowing two lane traffic on the road or the sign that informs large vehicle users to shift into a lower gear to drive safely down the incline of the road. *Variable message signs* are signs that variate the message they show like a roller blind sign that alternates the speed of which the traffic may travel at [9].

Combination signs are signs that combine two or more groups of signs to make one sign. An example of this type of sign is the sign that prohibits two lane traffic combined with the sign that gives information that the rule above is only required for the next two kilometres. Another example of a combination sign can be seen in Figure 1.1(b) where the sign indicates to the

FIGURE 1.6: *A guidance overhead freeway direction sign.*

road user of the upcoming speed bump in the road and the speed limit for the road which is 40km/h. Figure 1.1(c) is also a combination sign which indicates to the road user to yield ahead for pedestrians [3].

Finally there are *Temporary signs* for each of the groups of signs discussed above and they have a yellow background instead of the default colour of the sign. These temporary signs follow the same rules as their permanent counterparts except that the temporary signs are only temporary on the section of the road for which they apply. These signs are typically used at road works [18]. Figure 1.7(a) shows a temporary combination sign that indicates to the road user that in 400 metres the road will change from three lane traffic to two lane traffic temporarily and Figure 1.7(b) indicates to the road user that the temporary road rules no longer apply from this point further as the road works are at an end and thus traffic may resume at the correct speed limit for the road of 120km/h.

1.3 Sign production company

The work done in this thesis concentrates on a specific sign production company namely, *Kohler Signs (PTY) Ltd.* Kohler Signs (KS) is located in Cape Town, South Africa. They manufacture and install signs for the City of Cape Town, occasionally supply signs to Namibia and other parts of South Africa, manufacture and install commercial signs for private companies as well as manufacture advertisement signs to be placed on vehicles. They also sell road safety products to contractors, although this division of their company is not considered in this thesis [14].

There are many processes that Kohler Signs follow to manufacture any given traffic sign. Firstly the materials that are needed must be procured. These materials consist of steel sheets, various rolls of vinyl and either metal or wooden poles. The steel is then cut to the appropriate size and bent to match the design. The design of items that must be cut from vinyl is constructed and approved for cutting. The vinyl items are then cut out of the vinyl roll and applied to the steel sheet. The signs are then stored in the factory where they wait to be installed on site, or to be delivered to the client. The signs that must be installed by Kohler Signs are taken to the site along with the poles needed and are installed on site [14].

The process to construct the design of items that is to be cut out from the vinyl is done by the graphic designer of Kohler Signs using a brute force technique. The design of items is first approved by management before it is sent to the vinyl cutter to be cut from the vinyl. The items are placed next to each other as tightly as possible while allowing a small space to exist

(a) *A temporary combination sign indicating a lane change*

(b) *A temporary combination sign indicating the end of road works*

FIGURE 1.7: *Temporary signs.*

between two items to ensure that the vinyl cutter does not cut through an item accidentally [14].

The properties of a roll of vinyl consists of a normal colour layer and a reflective layer which reflects light back to the viewer. There are two types of reflective vinyl used by Kohler signs, namely *prismatic vinyl* and *beaded vinyl*, which may be seen in Figure 1.8. Prismatic vinyl consists of many lines that go horizontally across the sheet which helps with the reflective process and can be seen when observed at close range. During the packing process of letters, numbers and pictures on the vinyl, the items are only rotated by 180 degrees if the rotation provides an improved packing, since any other degree rotation will cause the lines on the reflective layer of the vinyl to not line up, which Kohler Signs do not want. The second type of vinyl is beaded vinyl which consists of small hexagonal shapes tightly packed together. This type of vinyl is not restricted by the rotations and thus may be rotated by any degree and still retain the same reflective properties and look. In this thesis only prismatic vinyl is considered.

1.4 Problem and objectives

In this thesis the problem is to minimise the waste generated through the placement of irregular items consisting of numbers and letters on a roll of prismatic vinyl. This waste consists of all the discarded pieces of vinyl that occur between the various irregular items that have been placed on the vinyl. Since all numbers and letters on a road sign have a width and height dimension and are known before production starts, only 2D off-line packing problems are considered in this

(a) *Rolls of Prismatic Vinyl* (b) *Sheets of Beaded Vinyl*

FIGURE 1.8: *Different types of vinyl.*

thesis. Finally, as discussed in §1.3, only 180 degree rotations will be allowed. Therefore, during the process of this research in order to be able to minimise the waste the following objectives are pursued:

1. To *review* heuristics for 2D packing problems found in the literature.
2. To *determine* an efficient way to represent the items (letters and numbers) that need to be packed.
3. To *develop* an algorithm to solve the 2D irregular packing problem in this case study.
4. To *generate* test cases for benchmark testing.
5. To *evaluate* the performance of the algorithm on generated benchmark cases as well as on real cases from Kohler Signs.

1.5 Thesis organization

Some background of the problem at hand has been described in this chapter. This includes the manufacturing process of a sign as well as the different types of road traffic signs. The emphasis of this thesis is on the process of cutting items from vinyl and therefore a few basic ideas of packing problems were also given. These will be elaborated upon a bit more in §2.1.

In §2.2 the various heuristics found in the literature regarding packing problems are described. This includes packing algorithms for regular items for 1D, 2D and 3D items as well packing heuristics for irregular items. However, for the purposes of this thesis, signs are manufactured by cutting items from a roll of prismatic vinyl and placed upon steel sheets. Therefore, more emphasis is placed on 2D packing heuristics. The first problem objective of reviewing 2D packing heuristics is thus covered in this section.

The method with which items are represented is described in Chapter 3. In this chapter, image processing techniques that are utilised to reduce the items to a few points from a large matrix of values, are discussed. Parameter values that are required for these processes are also determined in this chapter. Thus, Chapter 3 addresses Objective 2 in the list above and everything explained in this chapter was contributed to the author.

Chapter 4 describes the algorithm that has been developed to pack 2D off-line irregular items into a strip. This rule-set algorithm groups items into shapes that are compared to each other to help with the packing. Some of these items that need to be packed can be advantageous to rotate, to minimise wasted vinyl. The third objective of this thesis namely, to develop an algorithm to solve this 2D irregular packing problem case study, is thus presented in this methodology chapter.

In order to gauge if the developed algorithm is an improvement on the industry practices of Kohler Signs as well as being a good packing algorithm for this specific type of 2D irregular packing problem in general, benchmark cases must be collected and generated to test the developed algorithm. Chapter 5 opens with the properties that these benchmark cases consist of, as well as how the random cases are generated, addressing problem Objective 4. The resulting packings for these cases are also given in Chapter 5 and thus in this Chapter the fifth problem objective is also addressed.

The thesis is concluded in Chapter 6 with a brief summary of what has been done, followed by recommendations to KS that have been derived from the comparison of results in the previous chapter. The second part of Chapter 6 is devoted to the challenges that have been seen and ideas for future work.

CHAPTER 2

Literature review

Contents

In this chapter the focus is on packing problems. First, the properties of different packing problems as briefly mentioned in Chapter 1 are discussed in more detail in §2.1. Once the general type of packing problem needed for this research project is established, §2.2 is devoted to heuristics that are used to solve this packing, namely the off-line 2D strip packing problem. These heuristics include regular packing algorithms described in §2.2.1, as well as irregular packing algorithms as discussed in §2.2.2.

2.1 Packing problems

Packing problems are optimization problems in which a *good* placement of multiple *items* in larger containing areas which are considered as *objects*, is sought. A good placement of multiple items is one in which the items are packed as tight as possible without overlapping such that the unused areas in the objects are minimised. This type of problem can be seen in many fields of industry and business, for example, at Kohler Signs as discussed in Chapter 1. Other real life examples include industrial applications like cutting items from sheet metal, textile and leather respectively [13], cutting tubes for radiators heat units as well as container loading problems [8]. Each of these industries look to minimise the wasted material or wasted space between the packed items as described in §1.1.

In terms of dimensions, packing problems are divided into, *one-dimensional* (1D), *two-dimensional* (2D) and *three-dimensional* (3D) problems. In the case of 1D, only one dimension, a length dimension for example, is needed to compare the different items and to place them in the objects. Similarly, in the 2D [3D, respectively] case, two [three, respectively] dimensions need to be considered when the items are packed into 2D [3D, respectively] objects. The objective in a 1D packing problem is to pack the items in a number of 1D objects such that the number of objects is minimised resulting in the waste being minimised. An example of a one-dimensional packing problem is the cutting of tubes for radiator heat units. Tubes must be cut from stock

into smaller predefined lengths ensuring that the tube cut off's are minimised as illustrated in Figure 2.1. In this case 100m tubes must be cut into tubes of lengths 55m, 40m, 25m, 25m, 20m, 15m, 15m. For the cutting in Figure 2.1(b), the total waste is 5m and only two objects were used, whereas the cutting in Figure 2.1(c) used three objects with a total waste of 105m. Many items should be packed for a 1D problem to be utilised effectively, otherwise the items may be placed within one 1D object in any order and the total length would remain the same.

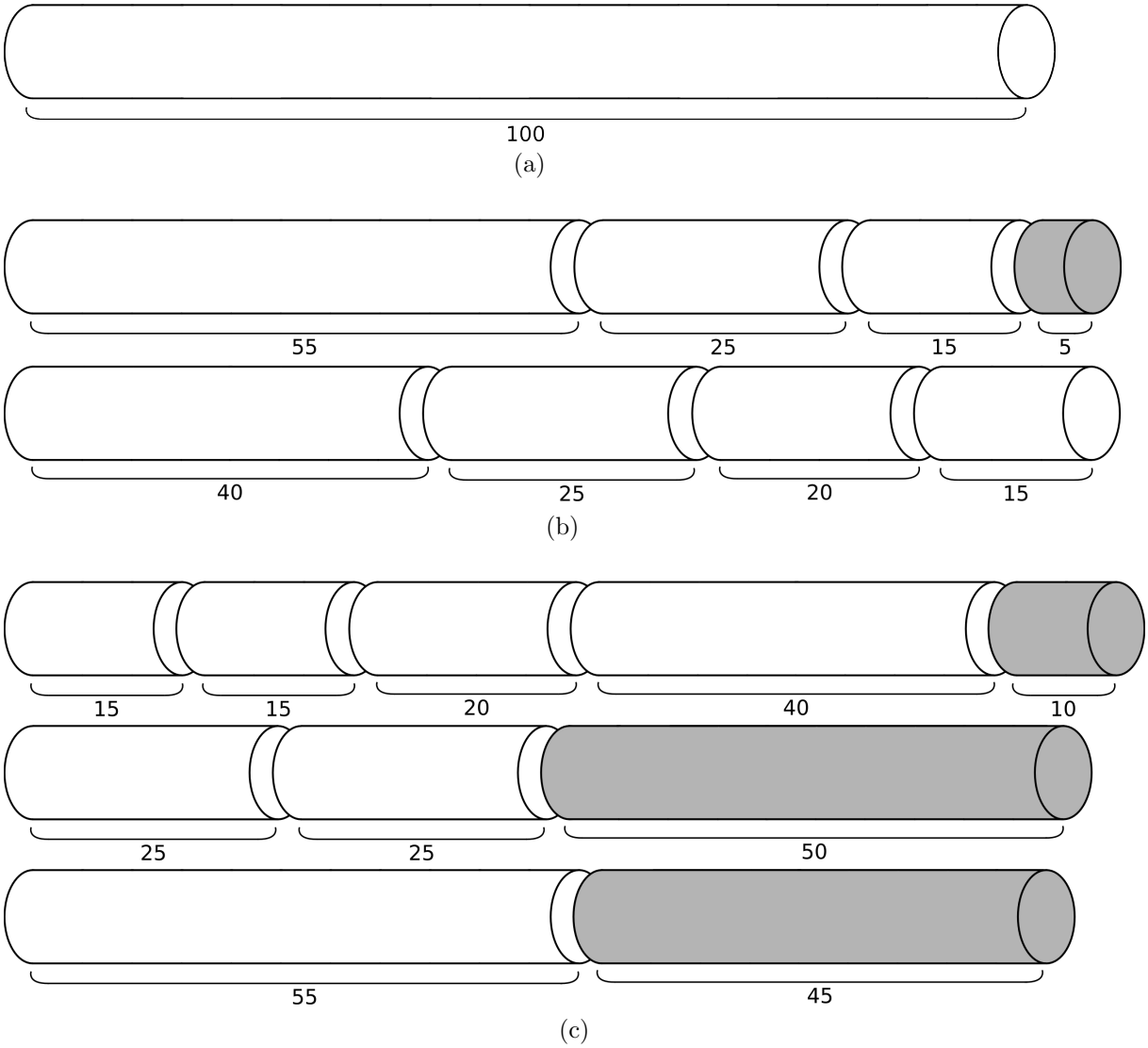
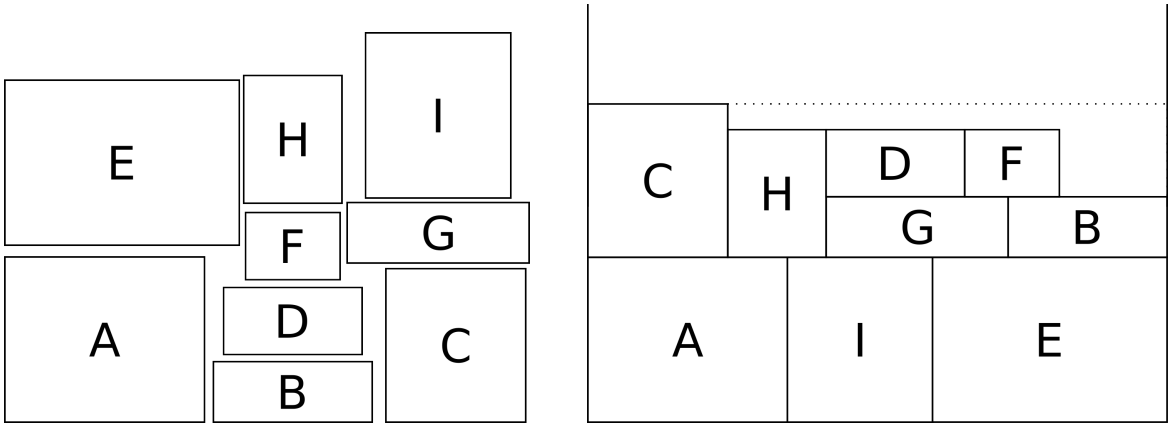


FIGURE 2.1: *Example of a 1D packing where 100m stock tubes shown in (a) are cut into tubes of lengths 55m, 40m, 25m, 25m, 20m, 15m and 15m such that the total waste is minimised. For the packing in (b) the total waste, indicated in dark tubes, is 5m. Suppose now that the 15m, 15m, 20m and 40m tubes are cut first, followed by the two 25m tubes, then the 55m tube needs to be cut from a third 100m tube resulting in the packing in (c) with a total waste of 105m.*

When one considers the objects of a 2D or 3D packing, there are two types of packing techniques used by industries called *strip packing* and *bin packing*. In 2D [3D, respectively] strip packing problems, items are packed into one object only and this object has one dimension fixed and the other dimension [the other two dimensions, respectively] is unbounded. Items are placed within the strip in such manner as to minimize the length of the unbounded dimension(s). This type of packing is used in industries where items are cut out of rolls of material, where the

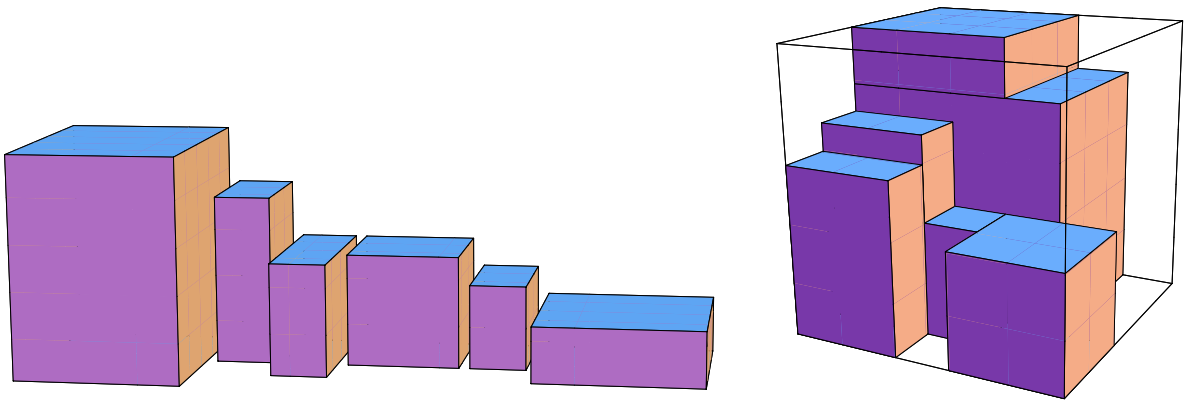
unbounded dimension is the length of the material, and this length of material from the roll must be minimized [16]. The examples of the sheet metal, textile and leather industries provided at the beginning of the section are also two-dimensional strip packing problems. Sheet metal must be cut from coiled material with a fixed width. Metal designs must be cut from this coiled material in such a manner as to minimise the metal offcuts. A simple version of the sheet metal example can be seen below where the sheets of metal, as seen in Figure 2.2(a) are cut out from a long sheet material, which may be seen in Figure 2.2(b).



(a) (b)

FIGURE 2.2: *Example of a 2D strip packing where the items to be packed are listed in (a) and the packed items are given in (b).*

The sheets of metal listed in Figure 2.2(a) may also be cut from large sheets of material with both dimensions fixed, in which case it is referred to as bin packing. In general, bin packing problems are similar to strip packing except all the dimensions are bounded. Furthermore, in bin packing problems, the items must be placed within many objects. The objective for bin packing problems is to minimize the number of bins (*i.e.* objects) used for packing. The bins used for packing may vary in dimensions or they may consist of many bins that have the same dimensions. Examples of regular 3D bin packing problems include container and pallet loading. An example of a 3D bin packing where all the items fit into one object (bin), is illustrated in Figure 2.3.



(a) (b)

FIGURE 2.3: *Example of a 3D packing where the items to be packed are listed in (a) and the items packed within a single bin is shown in (b).*

Apart from the fact that packing problems may be regarded as off-line or on-line depending on whether the items to be packed are known before the packing starts, or not, as discussed in §1.1, the items in 2D and 3D packing problems can also be described as being *regular* or *irregular*. Regular items are those that can be determined by a few parameters like rectangles and circles. Irregular items are more complex and require more parameters in order to be described correctly. Examples of regular and irregular items may be seen in Figure 2.4. Packing problems, for both

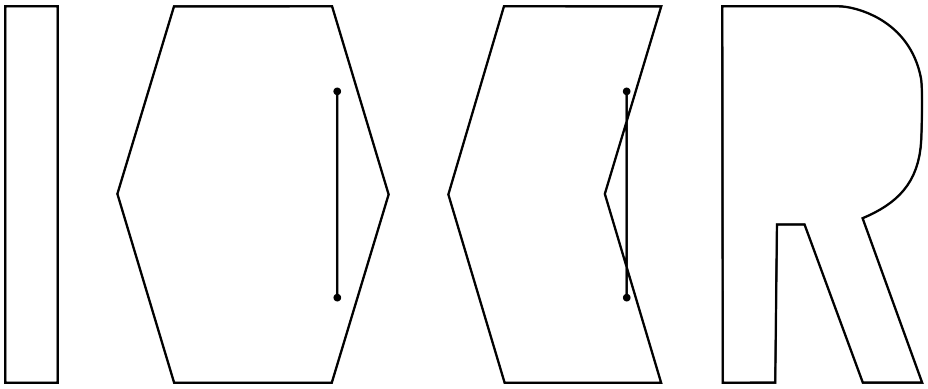


FIGURE 2.4: *Examples of regular, (a) and (b), and irregular, (c) and (d), 2D items respectively. Furthermore, the item in (b) is a convex only polygon and the one in (c) is a convex concave polygon, while the item in (d) is free-form.*

2D and 3D, can further be separated into those that are *guillotineable* and those that are *non-guillotineable*. Guillotineable problems refer to problems where the packing of the items always result in a straight cut that does not intersect any item. In non-guillotineable problems any straight cut through the packing might result in the intersection of one or more items. An example of the differences between guillotineable and non-guillotineable packings can be seen in Figure 2.5. In Figure 2.5(a) one can make any orthogonal cut along the bounds of any item and it will not intersect another item. However, in Figure 2.5(b), an orthogonal cut cannot be made without intersecting an item, as illustrated by the dotted lines.

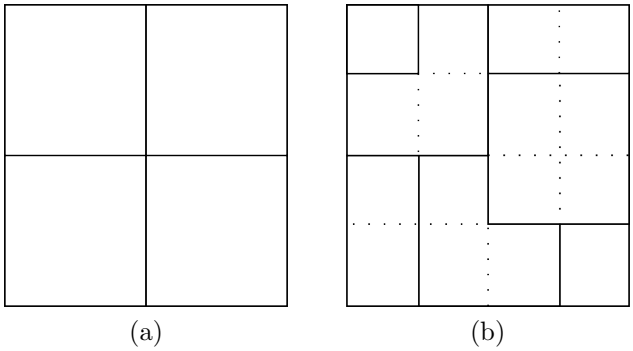


FIGURE 2.5: *Examples of (a) guillotineable and (b) non-guillotineable packings.*

Irregular packing problems, for both 2D and 3D, are described as being either *polygon* or *free-form*. Polygon items are comprised of many vertices that are joined together by edges and can be further described as being either *convex only* or *convex concave*. Convex only items are polygons that only make convex shapes, *i.e.* if a straight line segment is drawn between two vertices in the polygon, the line segment will never touch outside the polygon as illustrated in Figure 2.4(b). Convex concave items are polygons that consist of both convex and concave angled polygons,

that is, if a straight line segment is drawn between two vertices in the polygon, it *may* be able to exist outside the polygon as illustrated in Figure 2.4(c). Free-form items consist of vertices that are joined together by curved lines. The curved lines give the free-form nature of the items and can be considered as infinitely many edges joined together [13]. The "R" in Figure 2.4(d) is an example of a free-form item.

Packing techniques refer to the manner in which the items are packed within the objects. Industries use different packing techniques to find an optimal packing strategy specific for the job at hand. The sheet metal industry as well as the leather industry deal with regular and irregular nesting, or cutting stock, problems where 2D items are cut from sheet metal or leather stock. The cutting of tubes for radiator heat units are also considered as a cutting stock problem, in this case for 1D items. The textile industry utilises strip packing techniques to solve the packing problem. The container loading problem packs regular objects into one or many containers.

In this thesis only 2D off-line strip packing problems are considered since the items to be packed, namely letters, numbers and figures, are 2D items that need to be packed onto the 2D vinyl roll. Kohler signs use strip packing techniques when cutting out the letters and numbers from the prismatic vinyl rolls. Clearly, the letters and numbers are irregular items. However, the focus will be on both regular and irregular types of packing as well as polygon and free-form packing problems since some of the regular packing techniques are incorporated into the algorithms.

2.2 Packing heuristics

Heuristics are strategies for solving optimization problems approximately by constructing "good", but not necessarily optimal, solutions at a reasonable computational cost [16]. Heuristic methods are used for the strip packing problem instead of exact methods as this problem is NP-hard and the real world applications of this problem typically leads to large scale problems. The trade-off is made knowing that the heuristic delivers a near-optimal solution in a reasonable time, as opposed to an exact algorithm that might not obtain a solution at all, or obtain a solution in an unacceptable amount of time.

2.2.1 Regular packing algorithms

Regular packing algorithms consist of strip and bin packing algorithms and the items to be packed are regular items. These items are typically rectangular in nature. Strip packing algorithms pack the items in the order given onto a *level* within a strip, where a level is a horizontal rectangular band parallel to the floor of a strip with length equal to the width of the strip. These algorithms are used primarily when off-line packing problems are used. The bottom of the strip is considered the first level of the strip. The height of the strip is infinite and the height of each level is determined by the item with the largest height that has been placed on that level. Bin packing algorithms work in a similar manner to strip packing algorithms, except that the bins within which items are placed have a fixed width and height dimension.

One-Dimensional algorithms

One-dimensional packing algorithms pack items into an object, in this case a bin, that share a dimension, for example a width, with the object in which they are packed. Only one dimension, for example the height of the items, is thus significant. These 1D types of packing problems

are simple to solve as the solution space is quite small and can thus be solved optimally at reasonable computational costs [13].

The *next fit decreasing height* (NFDH) *algorithm* [5], for one-dimensional packing problems, sorts the items to be packed by decreasing height before any item is packed. Items that need to be packed using the algorithm, are placed within a packing list and an example of such a list can be seen in Figure 2.6(a). The first item is packed within the strip at the bottom of the strip. When an item is packed it is removed from the packing list. The next item is packed on top of the previous item if it can fit in the space remaining. Items are packed into the first strip in this manner until the next item cannot fit within the remaining space of this strip. A new strip is then constructed and this item is placed at the bottom of this new strip. The packing procedure repeats in this manner until all the items have been packed into strips. Figure 2.6(b) illustrates how the NFDH algorithm functions.

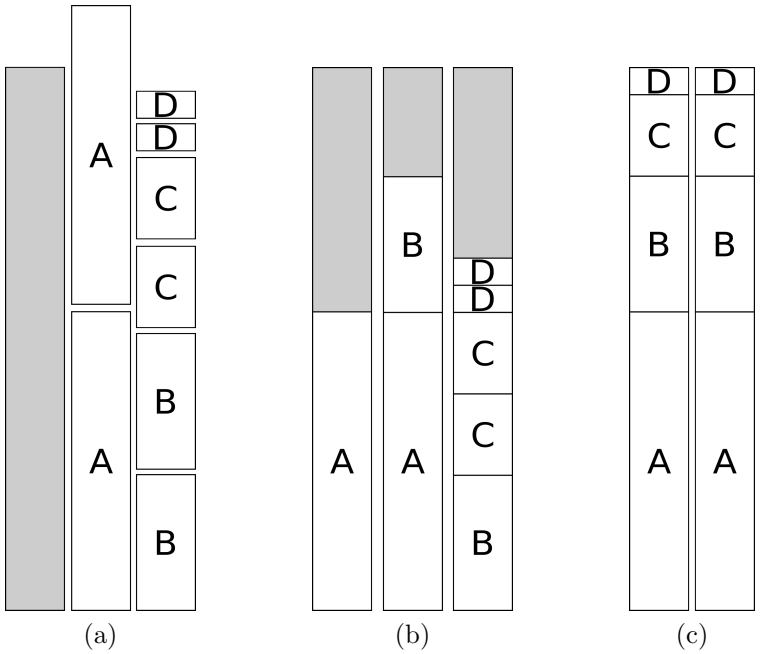


FIGURE 2.6: *(a) Example of a 1D bin and a list of one-dimensional items, packed within 3 (2, respectively) bins using (b) NFDH algorithm ((c) FFDH algorithm, respectively).*

Similar to the NFDH algorithm, the *first fit decreasing height* (FFDH) *algorithm* [13] also sorts the items to be packed by decreasing height before any item is packed. Again, items that must be packed using the packing algorithm are placed into a packing list. The first item that needs to be packed is placed on the bottom of the initial strip and removed from the packing list. The next item in the list is placed on top of the first item if it fits within the remaining space in the strip. If the item cannot fit within the remaining space in the first strip, then it is placed at the bottom of a newly constructed strip. In order to pack the next item, each stripped is looked at, and the first strip that has enough free space, is the strip the item is packed into. Items are thus packed into the first strip that has enough free space. An example of how this packing algorithm works can be seen in Figure 2.6(c).

Two-Dimensional algorithms

Two-dimensional bin packing algorithms work in a similar manner to the 1D packing algorithms in a sense that the number of bins utilised is minimised. However, now there are two variables

to consider for packing, that is a width and height variable. The dimensions for each bin may vary from bin to bin, but they are known before packing. Only strip packing algorithms are relevant for this project and thus only the various 2D strip packing algorithms are described here and not any 2D bin packing algorithms.

The NFDH algorithm for 2D problems sorts the items to be packed by decreasing height before any item is packed. Similarly to the 1D algorithm described above, items that must be packed using the algorithm, are placed into a packing list. An example can be seen in Figure 2.7. The

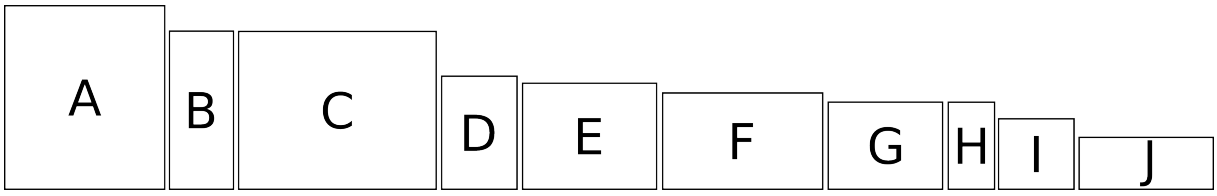
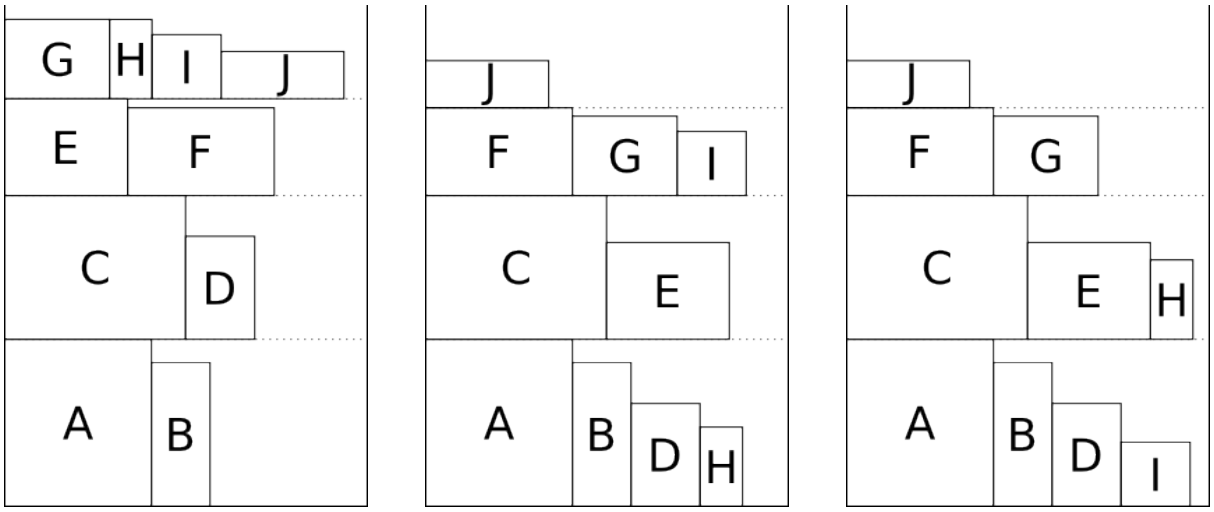


FIGURE 2.7: *An example of a list of 2D items to be packed.*

first item is packed within the strip in the far left corner of the strip. A level is constructed within the strip by utilising the height of the first item that has been packed as the height of the first level. The item that was just packed is then removed from the list. The next item within the packing list is placed next to the previous item if it can fit in the space remaining within the strip. If the item cannot fit within the space available adjacent to the previously packed item, it is placed on top of the already constructed level in the far left corner and a new level, with the same height as the newly packed item, is constructed above the previous level. Items are thus packed in this manner until the packing list is empty. Figure 2.8(a) illustrates how this packing algorithm works for the items listed in Figure 2.7.



(a) (b) (c)

FIGURE 2.8: *Illustration of the two-dimensional (a) NFDH, (b) FFDH and (c) BFDH algorithms using the list of items in Figure 2.7.*

Similar to the NFDH algorithm, the FFDH algorithm for 2D problems also sorts the items to be packed by decreasing height before any item is packed. In the same manner as the NFDH algorithm, the height of the first level is determined from the height of the first item packed. Again, the first item is packed within the strip in the far left corner of the strip and when an item is packed into the strip it is removed from the list. Occasionally, a new level is constructed above the last level using the height of the newly packed item. The construction of a new level

happens when an item cannot fit into the available space of any of the previous levels. Therefore, the next item is packed into the first space available that it can fit into, *i.e.*, if the item can fit in the space adjacent to an already packed item, it is placed there. If this item cannot fit into the remaining space on any existing level, then it is packed onto the left corner just above the existing levels, and a new level is constructed. An example of how this algorithm works can be seen in Figure 2.8(b).

The final algorithm discussed here is the *best fit decreasing height* (BFDH) *algorithm* [13]. Again, the items to be packed are sorted by decreasing height before any packing commences, just like the aforementioned packing algorithms above. The first item is packed within the strip in the bottom left corner of the strip and it is removed from the packing list. In the same manner as the NFDH and FFDH algorithms, the height of the first level is determined from the height of this first packed item. This algorithm works in a similar manner to the FFDH algorithm except that items are not packed in the first space that they fit, but are rather packed into the space with the *minimum residual horizontal space*. This means that each item is placed, in packing order, within the level of the strip which has the smallest space available for it to be packed into. Each item is thus placed on the level that leaves the smallest horizontal space and an example of such a packing can be seen in Figure 2.8(c). Comparing Figure 2.8(b) and (c), both algorithms place the items in the same locations from item "A" to "G", however there are notable differences for the remaining items. Item "H" is placed, according to the FFDH algorithm, on the first level that it can fit within. Thus item "H" is placed on the same level as items "A", "B" and "D". The final item to be placed, item "I", is placed upon the first level within which it may fit, which is not the first level consisting of items "A", "B" and "D", nor the level that consists of items "C" and "E" as the remaining widths of these levels are too small for the item. Item "I" is instead placed on the final level consisting of items "F" and "G" as this is the first level that has remaining width larger than the width of the item "I". Figure 2.8(c) illustrates that the BFDH algorithm places items "I" and "H" differently to the FFDH algorithm. Item "H" is placed upon the level with the best space available which is the level that occupies the minimum residual horizontal space. In this case, the level with the smallest space remaining that is large enough for item "H" is the second level, which consists of items "C" and "E". Item "I" is placed in a similar manner upon the first level adjacent to item "D".

Three-Dimensional algorithms

Three-dimensional packing algorithms work in a similar manner as the 1D and 2D algorithms discussed above, however they have a third variable, a depth variable that must be considered for packing. Items are packed into bins (where the dimensions are known beforehand) or strips where the depth dimension is unbounded. Packing algorithms of this nature adapt well known 2D algorithm techniques, like the NFDH and BFDH algorithms discussed above, to the 3D packing case. The 3D packing problem is reduced to packing items into 2D subbins called *slices* which are in turn packed into one or many 3D bins [5].

2.2.2 Irregular packing algorithms

Irregular packing algorithms consist of shapes that are irregular in nature and thus increases the level of complexity when packing these items. The more the number of non-linear sides of an irregular item is, the more complex packing it becomes. Therefore, there is a relationship between the level of complexity of the items to be packed and the complexity of the packing process. Simpler irregular items can be approximated to rectangular or polygonal shapes in

order to simplify the packing process.

Bottom Left heuristic

The *bottom left* (BL) *heuristic* for packing irregular items tries to move the items into the bottom most left open space. Irregular items begin to be placed within the packing space in the top-right-hand corner of the packing area. Items are lowered and shifted to the left as far as possible within the packing area in an iterative process always ensuring that no item overlaps the bounds of the packing area or other items already packed. This process of shifting the item lower and to the left repeats until the item cannot be placed any lower or to the left as possible. The BL heuristic performance relies heavily on the order of the initial packing list of items. This technique is simple and fast in implementation which is an advantage that the algorithm has over other techniques [17].

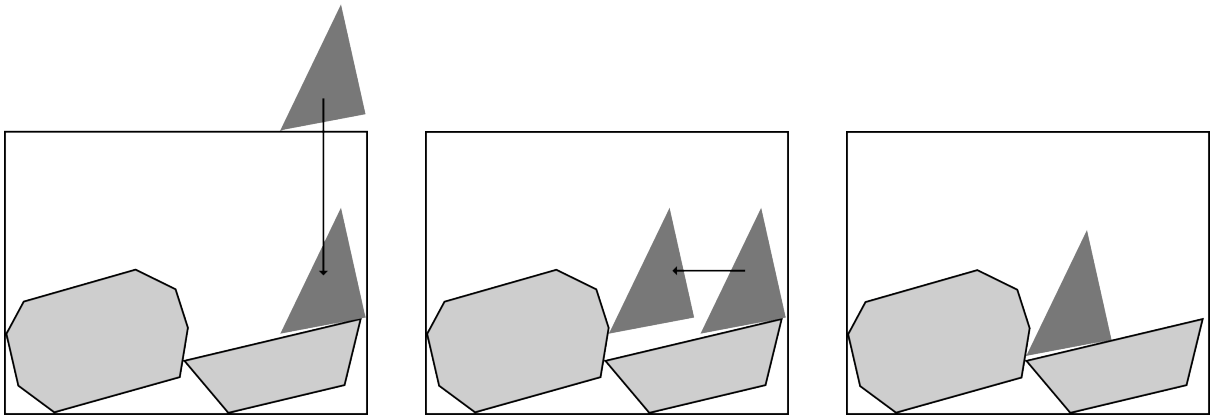


FIGURE 2.9: *Illustration of the bottom left heuristic for irregular items.*

Figure 2.9 illustrates the process of placing an item, which in this example is a triangular item, into the 2D bin utilising the bottom left heuristic for irregular items. An irregular octagon and trapezium items have already been placed within the bin. The triangle item has an initial position of just outside the bin's top-right corner. It is then lowered into the bin as low as possible without intersecting either the bottom of the bin or any item already placed, in this case the trapezium item. The triangle item is then moved as far left as possible without intersecting the left hand side of the bin or any item that has already been placed which in this example would be the irregular octagon item already in the bin. These two moving procedures repeat in this order until the item cannot move any lower or any further to the left without intersecting the bin or any items. The triangle item is thus placed adjacent to the irregular octagon item and on top of the trapezium item.

Constructive Approach

The *constructive approach* (CA) *heuristic* places the first item into the bottom left of the packing space. The next item has five possible locations where it may be placed within the packing space. These five possible packing locations are $(\bar{x}, 0), (0, \bar{y}), (\underline{x}, \bar{y}), (\bar{x}, \underline{y})$ and $(\bar{x}, \bar{y})$, where $\bar{x}, \underline{x}, \bar{y}, \underline{y}$ are the maximum and minimum x and y coordinates of the last item already packed, and can be seen in Figure 2.10 where two items are already packed.

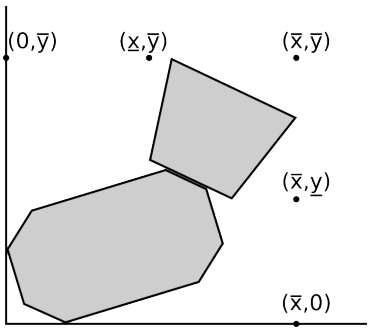


FIGURE 2.10: *The five possible locations for the next item to be packed using the CA heuristic after two items were already packed with the trapezium the last one being packed.*

Some of these locations may coincide and thus only unique locations are recorded. Out of these possible locations, those that may overlap with already packed items are ignored and the rest are considered as *candidate initial locations* for the next item to be packed. The next item is packed into the packing space from one of the five possible candidate locations utilising the BL heuristic described above, that is the item is shifted as far down and far left as possible in an iterative manner stopping before it may intersect the bounds of the packing space or other items that have already been placed in the packing space, as seen in Figure 2.11. This process is done for each of the five possible candidate locations, and the best packing procedure, where the "best" procedure is defined by the deepest (left and bottom) packing, is chosen, except in special cases when a hole is formed.

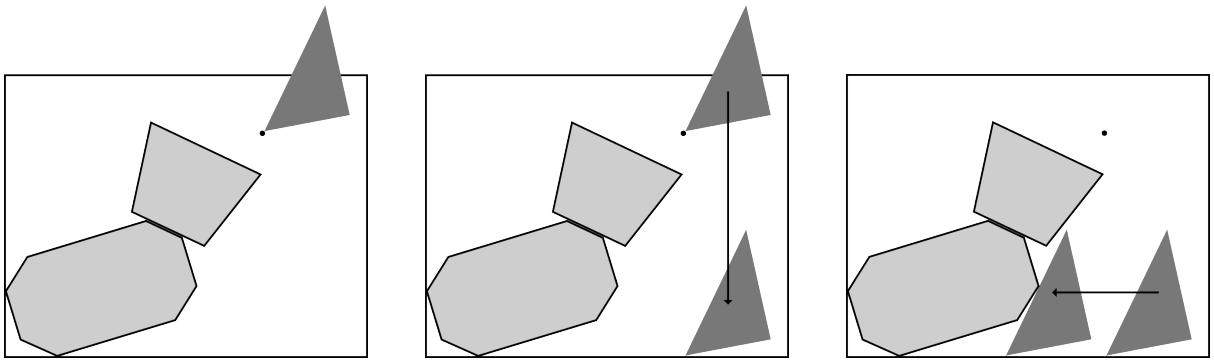


FIGURE 2.11: *The placement of an item utilising the CA heuristic when the next item is packed from the candidate initial location* $(\bar{x}, \bar{y})$.

The candidate initial locations are considered as departure points for the next item to be packed via the CA heuristic. The item slides down and left from these locations, such that certain gaps between already packed items may be reached. This constructive approach utilises simple geometric tools, avoiding more sophisticated computations such as minimal convex enclosures[1] [12].

A couple of variations for the CA heuristic exists [17], including the *constructive approach (minimum area)* (CAA) and *constructive approach (maximum adjacency)* (CAD) respectively. For the first modification, namely the CAA, the variation consists of selecting the best position from the list based on which one yields the (minimum) bottom left *bounding rectangle*, which

[1] Given two simple polygons, the algorithm finds the relative position of the one with respect to the other such that the convex area of the space that encloses the two polygons, is minimised.

is a rectangle that envelops a set of items, containing all the packed items within the minimum area. Figure 2.12 shows the bottom left bounding rectangle around the items already placed within the packing area. The area of the maximum horizontal coordinate and maximum vertical coordinate of all the items already placed plus the new item to be placed in the proposed position, is calculated.

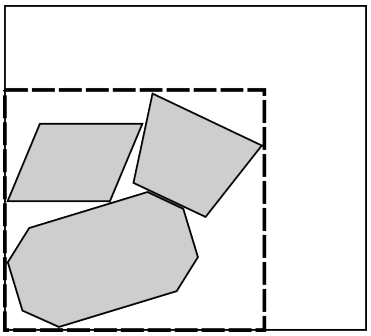


FIGURE 2.12: *Minimum bounding rectangle located in the bottom left corner containing all the items already packed, that is used in the CAA heuristic.*

Figure 2.13 shows two possible bounding rectangle candidates for a single item to be placed at two separate locations. This criterion is based on the idea of selecting a point with which all items are deepest (bottom and left), not only the latest item.

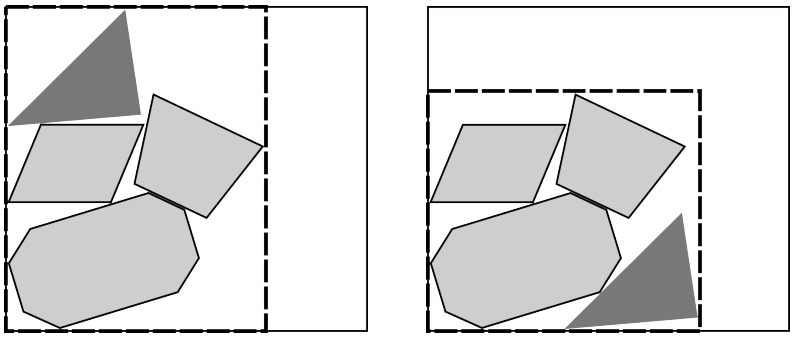


FIGURE 2.13: *Candidate bounding rectangles for the addition of a triangular item in the CAA heuristic.*

In the second modification on the CA, *i.e.* the CAD, when the first item is to be placed, only the corners of the object are considered. For the placement of the remaining items, the candidate initial locations are the same as those in the CA heuristic. Each candidate initial location is evaluated twice. First, the item to be packed starts in the selected candidate initial location and its *adjacency region*, which is the common boundary between its perimeter and the already placed items, its bounding rectangle or the object edges, is computed. Then, the item shifts down and left and the adjacency region is determined again. The location with the largest adjacency region is selected as the location of the newly placed item [17]. This packing is illustrated in Figure 2.14 with the same item being packed as above, just slightly rotated to illustrate this packing procedure to greater effect. Figure 2.14(a) shows an adjacency region that encompasses 3 points only, as indicated by the blue vertices, due to the common boundary of the newly placed triangular item with the already placed items, as well as the common boundary of this triangular item and the bounding rectangle. Figure 2.14(b) consists of a greater adjacency region than above, due to the large vertical line that the newly placed triangular item shares with the bounding rectangle in addition to the point where the already placed item is touching

the newly placed triangular item. The packing with the greater adjacency region, as shown in Figure 2.14(b), is thus chosen during the execution of the CAD heuristic.

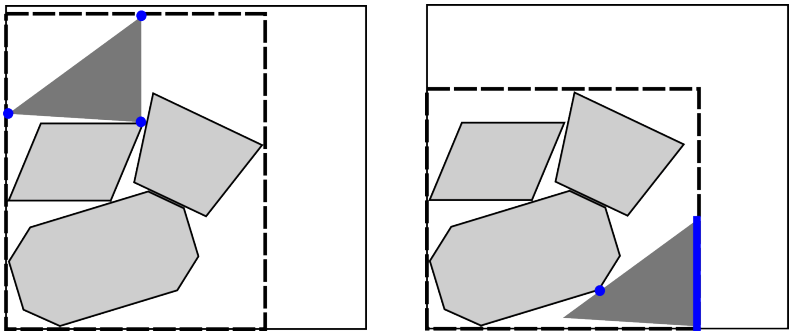


FIGURE 2.14: *Candidate bounding rectangles for the addition of a triangular item in the CAD heuristic with the adjacency regions highlighted.*

Packing of rectangular modules

Another method for packing irregular items is to enclose these items in rectangular shapes and to pack these newly formed shapes instead, because simple algorithms already exist for these rectangular packing problems. The enclosed items together with the rectangular shape enclosing them, are referred to as a *rectangular module* and all these rectangular modules are thus now considered the items to be packed. Many researchers have focused on this type of packing problem and Hopper [13] has done thorough research documenting the various techniques. An algorithm that computes the *minimal rectangular module* for an arbitrary closed curve has been developed by Freeman and Shapira [10] in 1975. Martin and Stephenson [15] proposed in 1988 a number of algorithms which pack arbitrary polygons and curved items into rectangles. In 1970 Haims and Freeman [11] applied the *minimum enclosing rectangle* concept similar to the bounding rectangle to a cluster of irregular items. Up to eight irregular shapes are clustered and circumscribed with the minimum enclosing rectangle after which the resultant rectangular modules are packed into the object in an optimal manner utilising dynamic programming. Adamowicz and Albano [1] introduced a technique in 1976 that allows the efficient clustering of two polygons by calculating the *No Fit Polygon* (NFP), which is all the arrangements that two arbitrary polygons may take such that the polygons touch but do not overlap, which in turn was initially presented by Art [4] in 1966 and describes all possible locations that an *orbiting polygon* (a polygon that has multiple possible initial locations for placement) can take with respect to a fixed already placed polygon so that the polygons do not intersect. Figure 2.15 shows the no fit polygon encasing a rectangular polygon and an irregular polygon where the polygons touch but do not overlap. This technique is used to find the minimum enclosing polygon of two polygons. This NFP method is thus applied to cluster two or more polygons that are then packed via the minimum enclosing rectangle approach. In 1977 Albano [2] developed upon these ideas an interactive packing system that allowed the operator to manipulate the proposed layout. The rectangular modules are grouped first and then placed within the rectangular object using dynamic programming.

Dagli and Tatoglu [7] solved an irregular packing problem involving multiple objects in 1987. Irregular items are enclosed by rectangles and allocated to objects applying mathematical programming techniques after which they are, based on this initial allocation, packed by a heuristic procedure. After an item is selected according to a set of priority rules it is moved towards

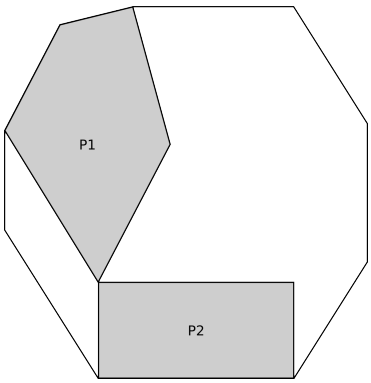


FIGURE 2.15: *An example of a No fit polygon for two polygons.*

the cluster of previously placed items. The algorithm searches for the configuration which has the smallest enclosing rectangle by repetitively matching the sides of the selected item and the cluster, and rotating the selected item. This procedure achieves good layouts, however it is computationally inefficient for irregular figures with a large number of sides due to the many rotations it may have to undergo.

2.3 Packing techniques summarised

In this chapter many packing techniques, problems and heuristics from the literature are presented and discussed. There are many properties that govern a packing problem. These range from the dimension with which they are packed, that is, 1D, 2D or 3D, respectively. The packing problems are further separated by whether or not the items to be packed are regular or irregular, polygons or free-form. The packings can either be off-line or on-line, guillotineable or not. These items can be placed on level shelves within strips or bins, or placed within rectangular modules. Table 2.1 gives a collated summary of the different types of 2D packing techniques found in the literature.

Technique	Characteristics of Packing	Author(s)
NFDH,	Regular, off-line, strips, bins	Hopper E [13]
FFDH &	Regular, 3D, off-line, guillotineable,	Bossenger W [5]
BFDH	rectangular, bins	
BL heuristic	Irregular, polygon, off-line, bins	Terashima-Marin H *et al* [17]
CA	Irregular, polygon, off-line, bins	Hifia M & MHallah R [12]
CAA & CAD	Irregular, polygon, off-line, bins	Terashima-Marin H *et al* [17]
	Irregular, polygon, free-form, off-line	Freeman H & Shapira R [10]
	Irregular, polygon, free-form, off-line	Martin RR & Stephenson PC [15]
Packing	Irregular, off-line	Haims MJ & Freeman H [11]
rectangular	Irregular, polygon, off-line	Adamowicz M & Albano A [1]
modules	Irregular, polygon, off-line	Art, Jr, RC [4]
	Irregular, off-line	Albano A [2]
	Irregular, off-line	Dagli CH & Tatoglu MY [7]

Table 2.1: *The techniques and characteristics of 2D packing heuristics.*

However, the 2D items that are packed in this project consist of letters that are in general

terms, irregular shapes but with rectangular features and can thus be packed as a regular packing problem to some degree. In particular, the height of each item is one of only a few possibilities, and as such rectangular type packing techniques can be utilised with regards to the height dimension. Thus, a strip packing method that packs items that are irregular in the width dimension only is sought.

CHAPTER 3

Display Method

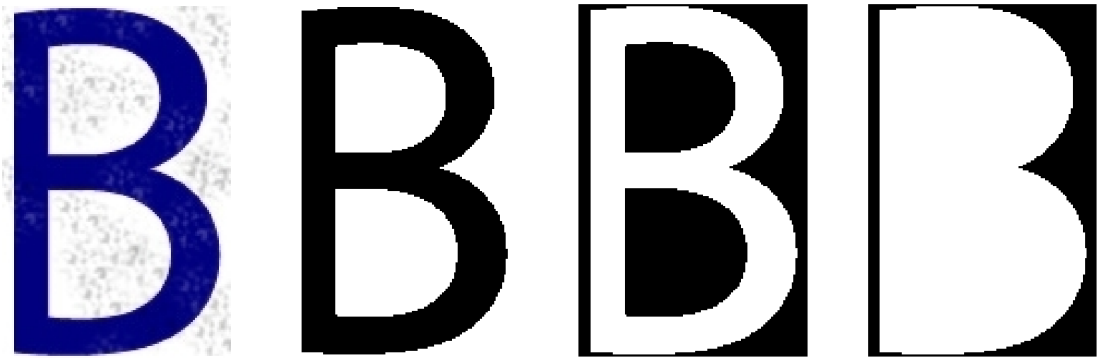
Contents

Before any packing can be done, the data structure of the sign items that need to be packed in this research project must be developed. Originally, the items are represented in the well known form of a matrix. The size of the data structure (matrix) is then reduced by first converting the item to a binary item as explained in §3.1, followed in §3.2 by determining the contour of the item instead of representing each pixel in the image. Finally, the number of pixels in the contour of the items is further reduced via polygon thinning. This will be discussed in §3.3. The last part of the chapter, §3.4, is devoted to the determination of the parameters for the data structure developing procedure.

3.1 Image processing

The items that must be packed onto the vinyl consist of letters, numbers and shapes. Each item must be processed so that it may be in the correct format to be packed by the packing program. The outline of the processing procedure is to first clean the item as well as converting this image with less disturbances into a binary representation of the item. Next, the colours are inverted if required, and finally any holes within the item are covered. Each of these procedures are discussed in greater detail together with the various outcomes associated with each procedure as illustrated in Figure 3.1.

Each item is represented by a matrix of values ranging between 0 and 255 where 0 represents black, 255 represents white and the values in between represent the colour spectrum. Some items might not be in the best condition for the packing program as they may have *dirty pixels* which are parts of the image that should not be there, as seen in Figure 3.1(a). Dirty pixels reside within the item and in the background of the image as well. It is important to clean the image so that these dirty pixels are not carried through to the packing algorithm causing



(a) (b) (c) (d)

FIGURE 3.1: *The letter 'B' during the various image processing stages. The original image including dirty pixels is in (a), a cleaner binary (black and white) representation of the image in (b), the inverted image in (c) and finally the letter without any holes in (d).*

incorrect items to be packed. This cleaning process is done as part of the binary conversion of the image.

The packing algorithm only needs to know where the item is within the complete image, *i.e.* where the outside boundaries of an item to be packed are. No other details, such as the colour of the item, are required. Therefore only two matrix values for an item are necessary in order to pack it correctly, namely a 0 and a 1, for black and white respectively. In order to get such a binary matrix, the item is normalized from a multiple colour image (a matrix ranging from 0 to 255) into a black and white image (a matrix consisting of 0 and 1's only). The normalization process compares each pixel in the image to a *cleaning threshold value* and if the value of the pixel is lower than this threshold value then that pixel is converted to black (at value 0), otherwise it is converted to white (value 1). The resulting black and white image for the "B" in Figure 3.1(a) is given in Figure 3.1(b). The threshold value is also used to determine which pixels in the image are dirty pixels that must be removed in order to clean the image. If the threshold value is too low or too high, dirty pixels that should be removed will stay behind, as seen in Figure 3.2(a) for a too low value and Figure 3.2(c) with a too high value. The cleaned up image of the item using a good threshold value may be seen in Figure 3.2(b), which is the same as the one in Figure 3.1(b). The process with which a good cleaning threshold value is obtained will be discussed in §3.4.1.

The final image that is required for the packing algorithm must be a white item with a black background as shown for the letter "B" in Figure 3.1(c). For this reason, if the item that is to be processed is some dark colour, say black, on a light background, say white, then the image must be inverted from a black item to a white item. To get the inverted image is a simple process of just swapping the 0's and 1's in the binary matrix. It is important to note that although this process works with images that vary with colour, the original images used in this thesis are black items with white backgrounds. Figure 3.1(a).

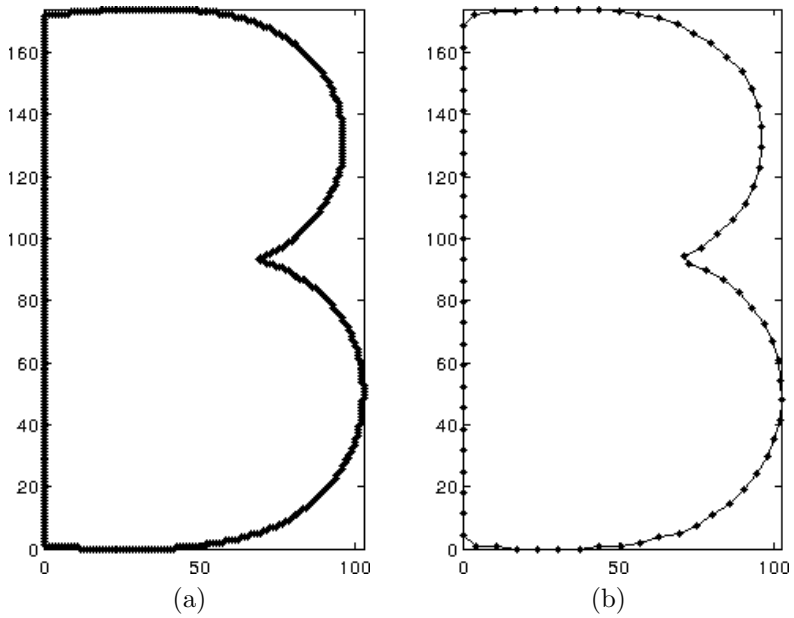
Some items, like the letter "B" as seen in Figure 3.1(a), or "Q", have holes within them that are irrelevant to the packing process as no item may be placed within these holes. Thus, for simplicity reasons these holes are removed from the item entirely. After this procedure, an item consists of a large matrix of either zeros or ones without holes within the item for which the image representation can be seen in Figure 3.1(d). Although these matrices are already reduced in size from the original matrices, working with these matrices can still be a cumbersome experience if there are many items to be packed resulting in many large matrices. Computationally, this can be a problem and thus it would be advantageous to simplify these matrices even further.

(a) (b) (c)

FIGURE 3.2: *The letter "B" after the image has been cleaned with threshold values of 0.1 in (a), 0.5 in (b) and 0.9 in (c). With the very low threshold value of 0.1 in (a) some dirty pixels become part of the image itself, while a threshold value that is too high as in (c) for example, the background of the image still contains dirty pixels that will make it difficult to isolate the image for the packing algorithm.*

3.2 Contour and Interpolation

The first method to simplify the final matrix representation of an item is to only consider the outer points of the image. If you consider the letter "I", for example, it is a rather simple letter. Essentially only a few outer points are required to construct this letter, however, this letter consists of a rather large matrix for each pixel found in the letter. This can be rectified by taking the outside (boundary) line of the image, which is a curve along the outer edge of the image, and thus, instead of having to work with all the pixels inside the item, only the outer points are used to represent the item [20]. Doing so will result in many points that may be joined together to form the *contour* of the item. The resultant contour of the letter "B" can be seen in Figure 3.3(a). At this point the image has been reduced from a large matrix to a set of points, representing the contour.

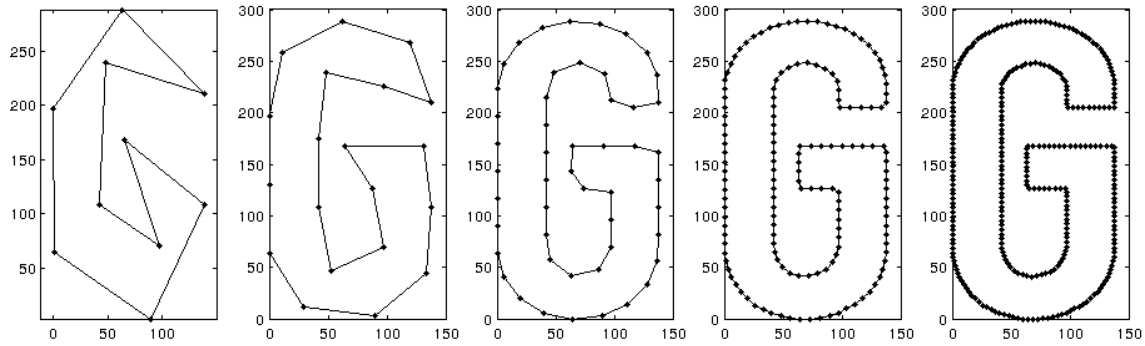


(a) (b)

FIGURE 3.3: *(a) The original contour of the letter "B" and (b) an interpolation to a smaller amount of data points of the same letter "B".*

The number of points used may be simplified even further using *interpolation*. Interpolation is a technique whereby new data points are constructed by estimating the path of known data points [22]. In this manner, interpolation may be used to expand on or reduce the number of already existing data points [6]. In this thesis, the program produces many data points from the original image and thus in this case, these data points are simplified to only a few points using interpolation. Figure 3.3(b) shows a new image of the letter "B" with a few interpolated points instead of the many points in the original contour seen in Figure 3.3(a).

The interpolated image still retains the same shape as before as long as the number of points to be interpolated is sufficient. In Figure 3.4 five different interpolations of the same image with varying number of points are shown. The first image looks quite distorted than what it should be as there are too few points to be interpolated. The last image resembles the letter correctly but has quite a number of points that are redundant. A balance must be found to best interpolate the image for the packing problem. Figure 3.4(c), where 50 data points have been interpolated for the letter "G", seems to be the best result for this example as the interpolated letter retains its original shape and utilises a small number of data points.



(a) *10 points* (b) *20 points* (c) *50 points* (d) *150 points* (e) *300 points*

FIGURE 3.4: *Some interpolations of the letter "G" with different number of points.*

3.3 Polygon thinning

Once the items have been interpolated they can be reduced to even fewer points by *polygon thinning*. Polygon thinning, or edge thinning, is a technique used to remove unwanted or redundant points within the edge of an image, from the contour representation of the image [21]. The straight line edges of an item are detected and any point in the convex combination of the two end points of the edge, is removed, thereby reducing and simplifying the number of points required to represent an item. For this research project, one may look at the path the points lie on and identify any significant deviations these points may follow. Paths that consist of points that do not deviate much, are considered edges. The polygon thinning technique identifies which points fall upon longer edges and remove those, except the two end points.

The letter "L", for example, consists of three horizontal edges and three vertical edges for a combined total of six edges. The edge detection for the letter "L" is straightforward and polygon thinning removes the points that lie within each of these edges, thus leaving behind the start and end points. Another example is the letter "J", this letter consists of two edges that are vertical, one edge that is horizontal and one edge that is diagonal for a total of four longer edges. The letter "J" also consists of a curved section. This curved section consists of many

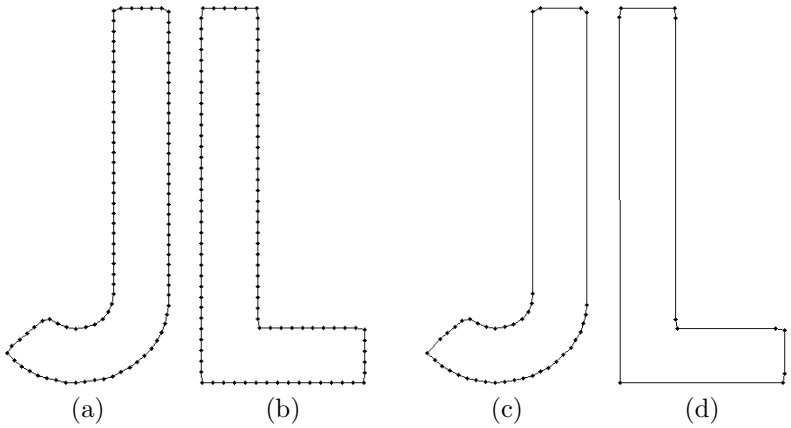


FIGURE 3.5: *The items "J" and "L" in (a) and (b), together with their polygon thinned versions in (c) and (d) respectively.*

straight line edges to form a semi-circle. The four longer edges of the letter "J" are identified and thinned using the polygon thinning technique and the curved section is ignored as it consists of a path with points that deviate greatly between each other. The small diagonal edge of the letter "J", in this instance, is only thinned to a small piece upon the edge. This is due to the fact that only a few points along the diagonal edge are used in the interpolation, resulting in an edge that is not entirely straight for all points upon the edge. A higher number of interpolated points over such a small edge will fix this issue, as seen with the short edges of the letter "L". Figure 3.5 illustrates the difference in the number of points for the letter "L" and "J", and their polygon thinned counterparts. These are significant improvements for the reduction of the number of points required to represent each letter.

3.4 Determination of parameters

In this section the value to be implemented for the three parameters that need to be used in the various image processing stages for this thesis, are obtained via sensitivity analysis. The image processing parameter values that need to be determined are the cleaning threshold value (§3.4.1), the number of points to obtain the contour of an item and the number of points to interpolate for the same item (§3.4.2).

3.4.1 Cleaning Threshold value

After an image item is read from file and converted into a colour matrix of values that range from 0 to 255, the first step is to convert the image matrix into a binary matrix. As explained in §3.1, during this process any objects formed from inaccurate pixels that should not exist within the image, are also removed. In industry these inaccurate pixels may be due to, for example, the letter "B" being scanned in from a dirty piece of paper, or the letter "B" may be sent to the graphic designer from a client but in poor resolution or quality, among other potential reasons. This would lead to the image consisting of dirty pixels that need to be removed via the cleaning threshold value. Therefore, a cleaning threshold value must be chosen with care. Choosing a cleaning threshold value that is too low will result in good pixels from the item to become deleted, whereas choosing a cleaning threshold value that is too high will keep too many dirty pixels within the image. In both of these cleaning threshold value cases, they give an incorrect

representation of the item at hand.

Figure 3.6 illustrates the letter "B" being cleaned with different cleaning threshold values over a full range of [0,1]. The first image, Figure 3.6(a), has a cleaning threshold value of 0 and shows an empty letter space. Every pixel has been given the value 1 (white) and have thus been removed from this image. Figure 3.6(b) has a cleaning threshold value of 0.05 and only shows a few pixels that have not been removed and still no evidence of the letter "B" exists. From Figure 3.6(c) towards Figure 3.6(f) the letter "B" is visible, however, some pixels that should exist are still missing. Thus, threshold values from 0.1 to 0.25 show obvious signs of pixels that have not been carried over, due to the fact that the cleaning threshold value is still too low.

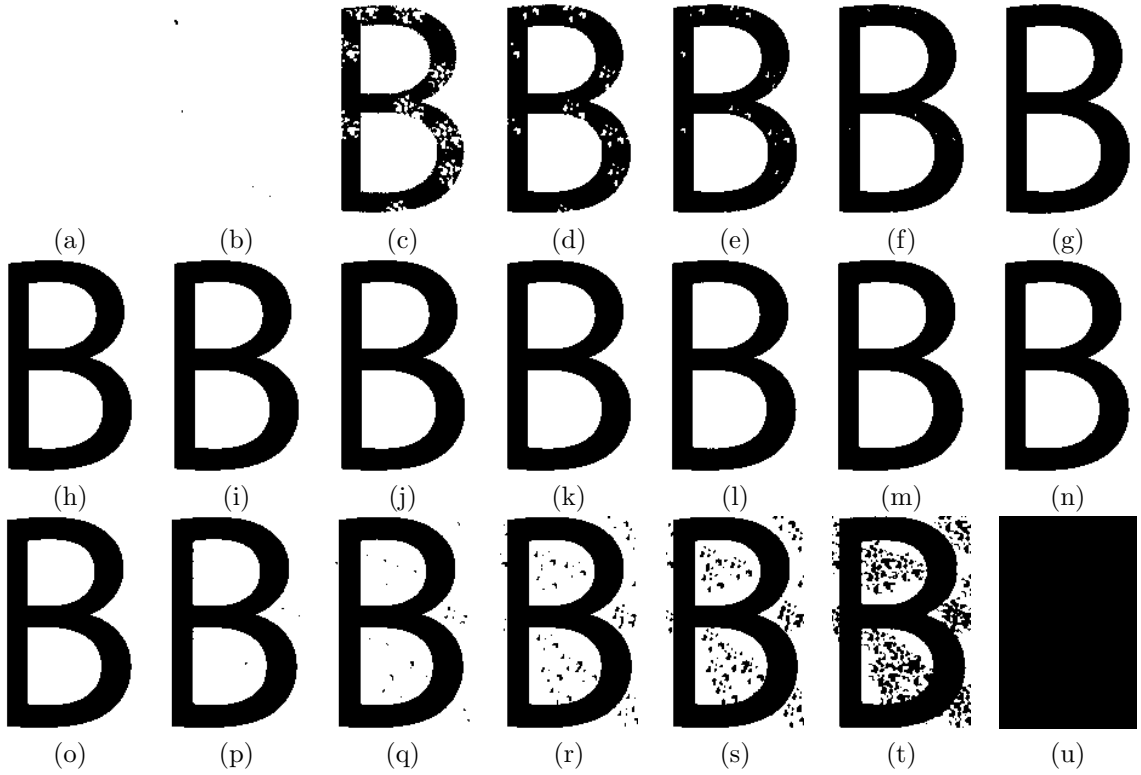
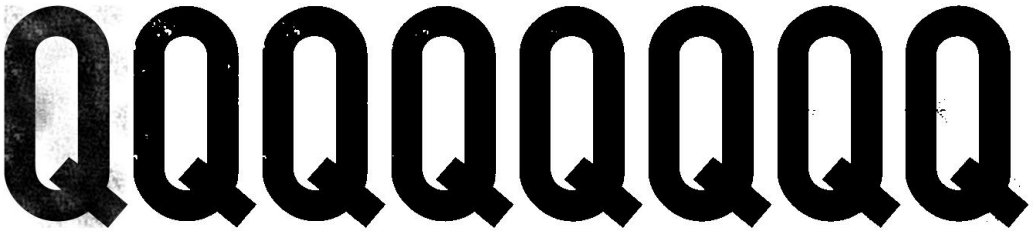


FIGURE 3.6: *Item "B" cleaned with different cleaning threshold values starting at 0 and ending at 1 in increments of 0.05.*

Looking from the high end of the cleaning threshold values, Figure 3.6(u) shows that having a cleaning threshold value of 1 takes every pixel, dirty or otherwise, and converts it to black. Figures 3.6(p) to (t) show obvious errors in not cleaning the items enough due to a cleaning threshold value that is too high. Cleaning threshold values that differ between 0.3 and 0.7 seem to be good values to use, since the images are cleaned, as seen in Figures 3.6(g) to (o).

Figure 3.7 shows another dirty item cleaned via different threshold values. The dirty item, in this example the upper case letter "Q", is shown in Figure 3.7(a) and it is cleaned with cleaning threshold values of 0.35 in (b) to 0.65 in (h), with increasing threshold value increments of 0.05. There exists decreasing errors in the cleaned item when the cleaning threshold value is increased from 0.35 in (b) to 0.45 in (d). Similarly, from the larger end of cleaning threshold values, the errors decrease as the cleaning threshold values decrease from 0.65 in (h) to 0.55 in (f), with (f) not containing any errors. The image in Figure 3.7(e) also does not contain any errors after cleaning.

The next example utilised in the search for a cleaning threshold value to use in this project,



(a) (b) (c) (d) (e) (f) (g) (h)

FIGURE 3.7: *(a) Dirty item "Q" cleaned with different cleaning threshold values starting at (b) 0.35 and ending at (h) 0.65 in increments of 0.05.*

(a) (b) (c) (d) (e) (f) (g) (h)

FIGURE 3.8: *(a) Dirty item "7" cleaned with different cleaning threshold values starting at (b) 0.35 and ending at (h) 0.65 in increments of 0.05.*

is a dirty number "7", as seen in Figure 3.8. This example has errors in each iteration, but upon closer inspection, give a good indication which cleaning threshold value to use. Cleaning threshold values that range from (b) 0.35 to (d) 0.45 consist of items that are missing pixels, specifically the top layer of pixels of the number "7". These missing pixels start to reappear in (d) onwards until they fully exist in (h). Conversely, error pixels that do exist in (h) start to disappear with decreasing cleaning threshold values until they are completely gone in (b). Thus, there is an overlap of incorrect pixels that do not exist in the image, and those that do, throughout the different cleaning threshold values and not one instance where the best value cleans the item completely. This is due to the image being too dirty to be cleaned correctly. The 'best' representation of the number "7" comes from cleaning threshold values of (d) 0.45 and (e) 0.5.

(a) (b) (c) (d) (e) (f) (g) (h)

FIGURE 3.9: *(a) Dirty item "e" cleaned with different cleaning threshold values starting at (b) 0.35 and ending at (h) 0.65 in increments of 0.05.*

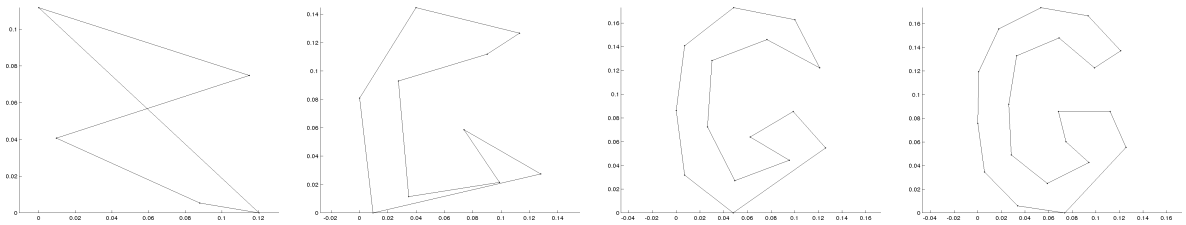
The last example, a dirty lower case letter "e" is cleaned, as seen in Figure 3.9. Many pixels are missing within the centre of this item when the cleaning threshold value is (b) 0.35, with pixels returning after each 0.05 increment increase of cleaning threshold values until the pixels have completely returned for a cleaning threshold value of (f) 0.55. Figure 3.9(h), which has a

cleaning threshold value of 0.65, shows pixels that should not exist in the image. These error pixels disappear as the cleaning threshold value is decreased by 0.05 per increment, until they are done, as seen in the image with a cleaning threshold value of (e) 0.5. As a last example, the item "e" with a cleaning threshold value of 0.5 as seen in Figure 3.9(e) still shows some errors, but is the best representation for the lower case letter "e".

Overall a good parameter for the cleaning threshold value would be a value ranging between 0.45 and 0.55. Thus 0.5 is chosen for the cleaning threshold value as it lies in the middle of this range of values and gives consistently close to accurate results.

3.4.2 Number of Interpolated points

After the item is cleaned it is reduced, through the various processes discussed earlier in the chapter. It was emphasised in §3.2 that the number of data points used during interpolation is important since choosing too few will disfigure the item, and choosing too many data points does not improve on the already obtained form while adding unnecessary complexities and computation time to the algorithm which are not ideal. However, as explained in §3.4.1, the number of points on the diagonal edge of the letter "J" is not reduced much during polygon thinning due to the fact that the small number of points during interpolation result in an edge that is not entirely straight. It is also stated that a higher number of points during interpolation will fix this problem. However, this will also increase the number of points along the curved section which is not reduced during polygon thinning. Therefore, choosing the number of points for interpolation must be done by considering the interpolation and polygon thinning processes simultaneously. Figures 3.10(a) to (d) shows how poorly chosen number of points to interpolate

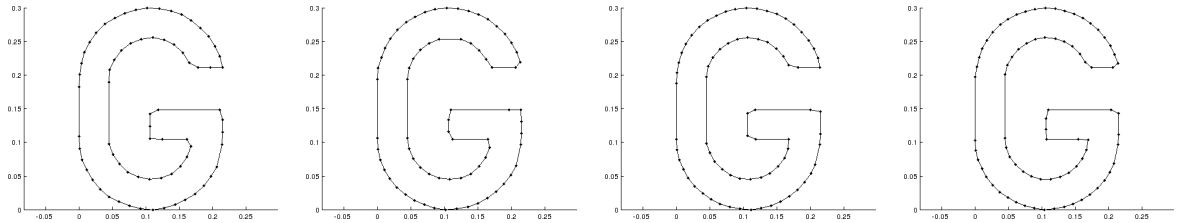


(a) (b) (c) (d)

FIGURE 3.10: *Item "G" interpolated with different number of points starting at (a) 5 and ending at (d) 20 in increments of 5.*

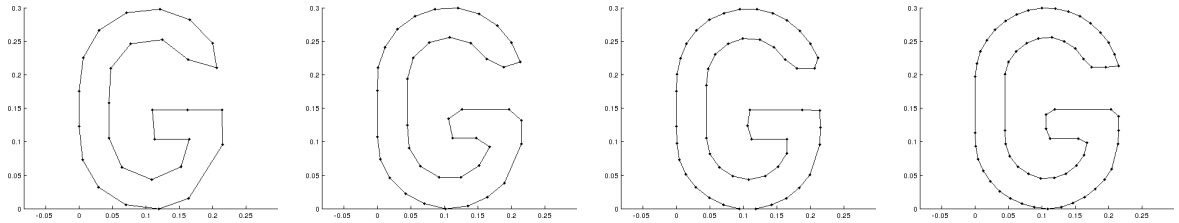
can reduce the item to nonsense. The first and second image does not even resemble the letter "G", after which the items begin to take on the appearance of the letter "G" to some degree, but are still misshapen. It is also clear that in all these cases no reduction of data points was obtained during polygon thinning.

On the other end of the spectrum, Figures 3.11(a) to (d) illustrate that too many points used for interpolation does not change the item much from the original image, if at all. Any number of data points to interpolate from 85 to 100 resemble the same image, due to a combination of points being removed from polygon thinning as well as redundant points being interpolated. Interpolating 85 or more data points is thus irrelevant. Figure 3.12 consists of items that fall between unrecognisable images of the letter "G" as illustrated in Figure 3.10, and items that resemble the letter "G" to a fault due to redundant data points, shown in Figure 3.11. Little change is made between the images that is interpolated using between 30 and 60 data points. By the time 75 data points have been used for interpolation, the image seems to be rather stable and appearing very much like the letter "G". Thus, 75 data points to interpolate seems to give



(a) (b) (c) (d)

FIGURE 3.11: *Item "G" interpolated with different number of points starting at (a) 85 and ending at (d) 100 in increments of 5. The resulting number of points after polygon thinning is (a) 72, (b) 73, (c) 76 and (d) 80, respectively.*



(a) (b) (c) (d)

FIGURE 3.12: *Item "G" interpolated with (a) 30, (b) 45, (c) 60 and (d) 75 data points respectively. The resulting number of points after polygon thinning is (a) 29, (b) 41, (c) 53 and (d) 64 respectively.*

a good representation of the letter "G" without compromising on the accuracy of the image as a whole or using too many data points.

The same comparisons were done for an arbitrarily chosen set of items which differ in size and construction. The results for these comparisons are summarised in Table 3.1 and illustrated in Figures 3.13 - 3.17. In all cases the number of points used for interpolation are (a) 60, (b) 70, (c) 80, (d) 90, (e) 100, (f) 110 and (g) 120 data points. The basic shape of the first item, namely the number "2" is retained throughout all these respective examples, becoming sharper as the number of data points increase until these points over saturate the "2" such that too many points are used for its diagonal spine. There is a need for many data points to represent the item "2" when 60 data points become 51 due to polygon thinning, but after that the number of original data points needed only increases slightly. The number "2" consists of a couple of curved pieces that utilise many data points. Too many original data points causes the straight lines to become slightly crooked, as seen in Figure 3.13(e) onwards.

	number of data points for interpolation.						
item	60	70	80	90	100	110	120
2	51	57	62	73	78	83	93
a	46	54	60	66	70	75	82
m	39	45	47	51	55	58	62
R	39	43	48	53	56	61	63
W	59	68	79	74	99	86	106
G	53	61	68	73	80	89	92

Table 3.1: *The final number of data points after polygon thinning for a few items.*

The letter "a" as illustrated in Figure 3.14, utilises fewer data points per data point increment than the number "2" even though there are more curved pieces in the letter "a" than in the

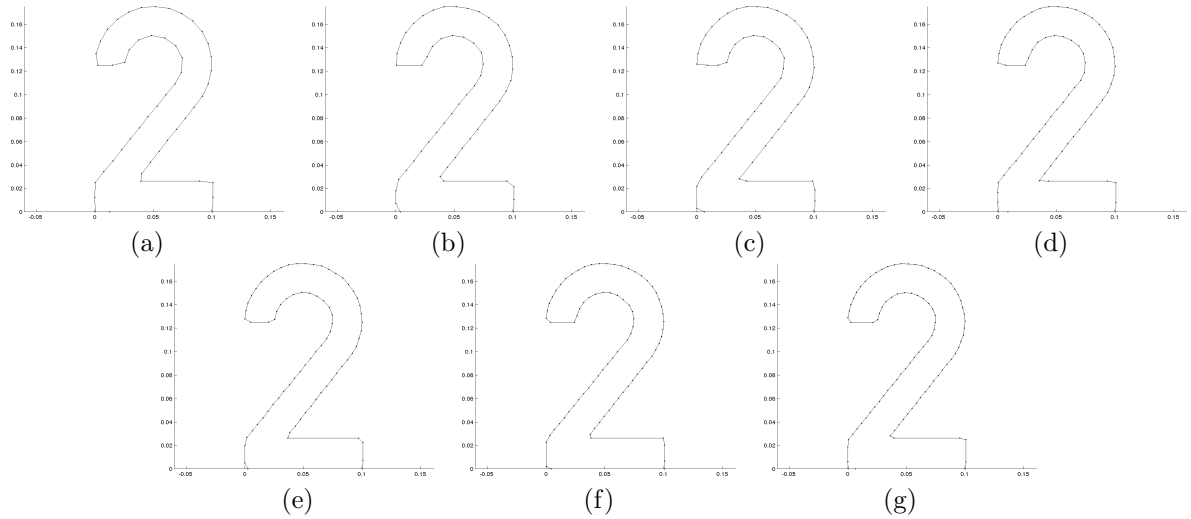


FIGURE 3.13: *Item "2" interpolated with (a) 60, (b) 70, (c) 80, (d) 90, (e) 100, (f) 110 and (g) 120 data points respectively. The resulting number of points after polygon thinning is (a) 51, (b) 57, (c) 62, (d) 73, (e) 78, (f) 83 and (g) 93 respectively.*

number "2". The shape of the item "a" displayed in the example, are represented more from 100 data points onwards.

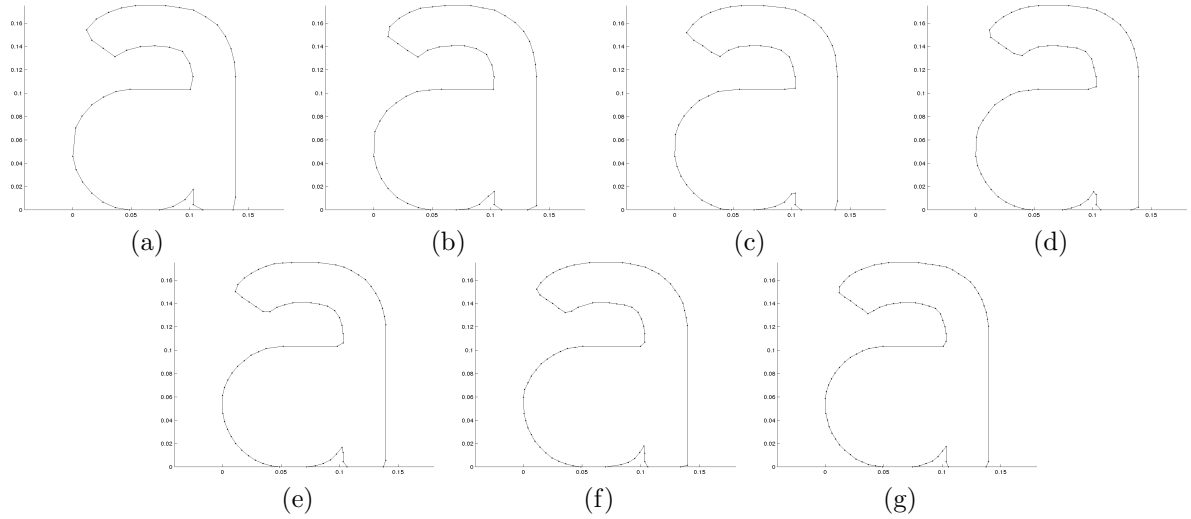


FIGURE 3.14: *Item "a" interpolated with (a) 60, (b) 70, (c) 80, (d) 90, (e) 100, (f) 110 and (g) 120 data points respectively. The resulting number of points after polygon thinning is (a) 46, (b) 54, (c) 60, (d) 66, (e) 70, (f) 75 and (g) 82 respectively.*

The first two polygon thinned results of the letter "m" in Figure 3.15, are poor reflections of the letter "m" as they consist of examples where the tips of the letter "m" are chipped. The item "m" resembles its original letter where the data points equal 100 or more. This illustrates that even though items like the letter "m" consist of many horizontal and vertical lines which help with polygon thinning, the curved lines cause difficulties in a good representation for the item with too few data points.

The shape of the item "R", shown in Figure 3.16, resembles the upper case "R" from 60 data points but becomes a better reflection of the letter from 90 data points upwards. These data points, from 90 to 120, consist of diagonal lines that become crooked as the data points are

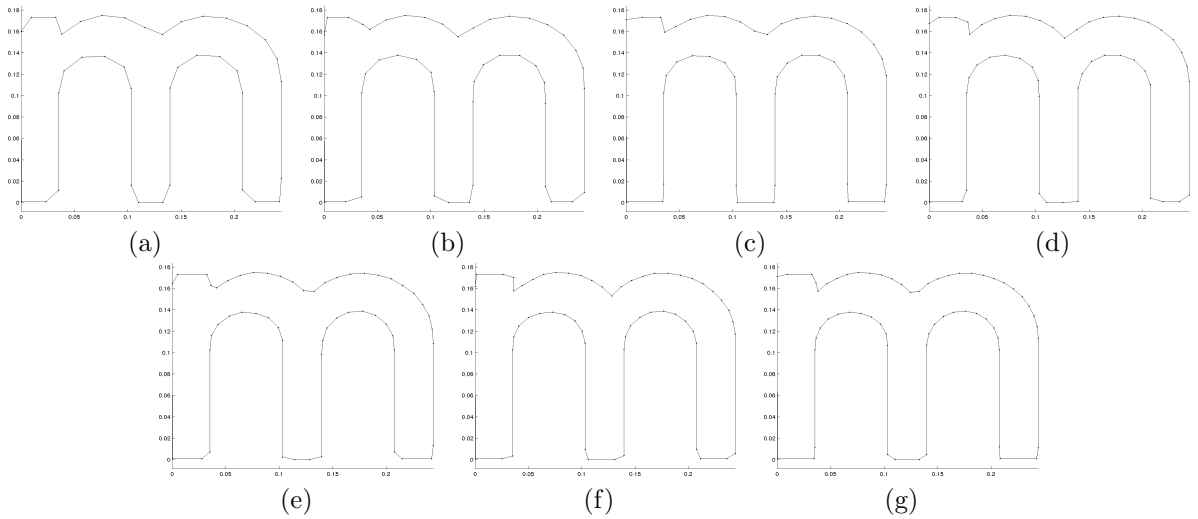


FIGURE 3.15: *Item "m" interpolated with (a) 60, (b) 70, (c) 80, (d) 90, (e) 100, (f) 110 and (g) 120 data points respectively. The resulting number of points after polygon thinning is (a) 39, (b) 45, (c) 47, (d) 51, (e) 55, (f) 58 and (g) 62 respectively.*

increased.

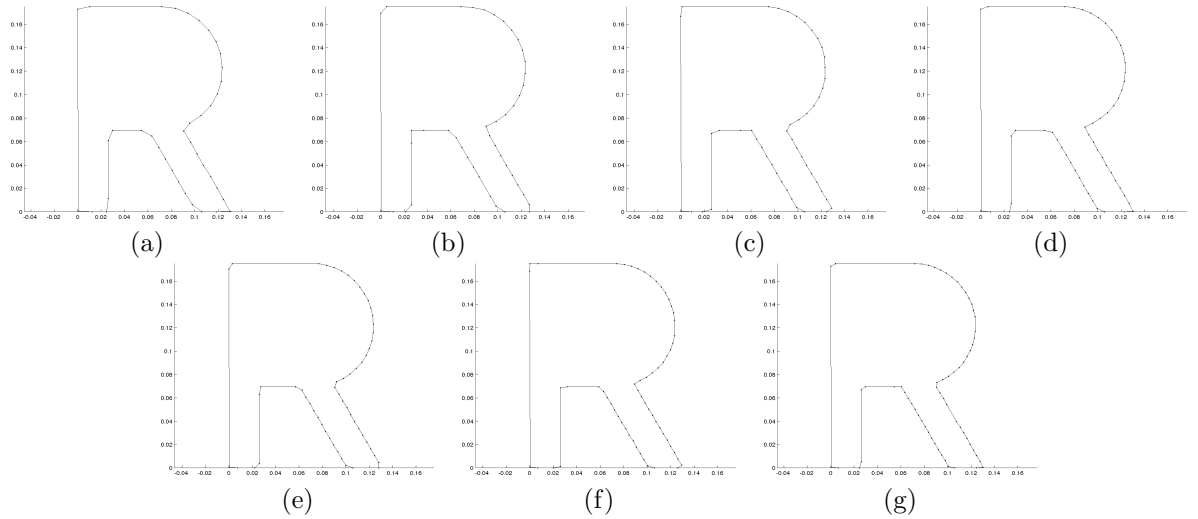


FIGURE 3.16: *Item "R" interpolated with (a) 60, (b) 70, (c) 80, (d) 90, (e) 100, (f) 110 and (g) 120 data points respectively. The resulting number of points after polygon thinning is (a) 39, (b) 43, (c) 48, (d) 53, (e) 56, (f) 61 and (g) 63 respectively.*

For the final item, the letter "W" illustrated in Figure 3.17, not many data points are polygon thinned between the 60 and 80 data point marks, thus indicating that there are too few data points representing this shape. The tips of the item "W" are erratic from 60 data points till 90 data points and are not a fair reflection of the letter "W". Thus this item represents the letter "W" the best from 100 data points onwards.

The resulting (final) number of data points after polygon thinning for each of the items above, are compared in Table 3.1. The second column of Table 3.1 is the final data points after polygon thinning for the items in the case where 60 data points were used for interpolation. Studying the first row, one may notice that item "2" utilises a significant proportion of available data points after polygon thinning, but does not hit the upper bound of data points supplied. Items

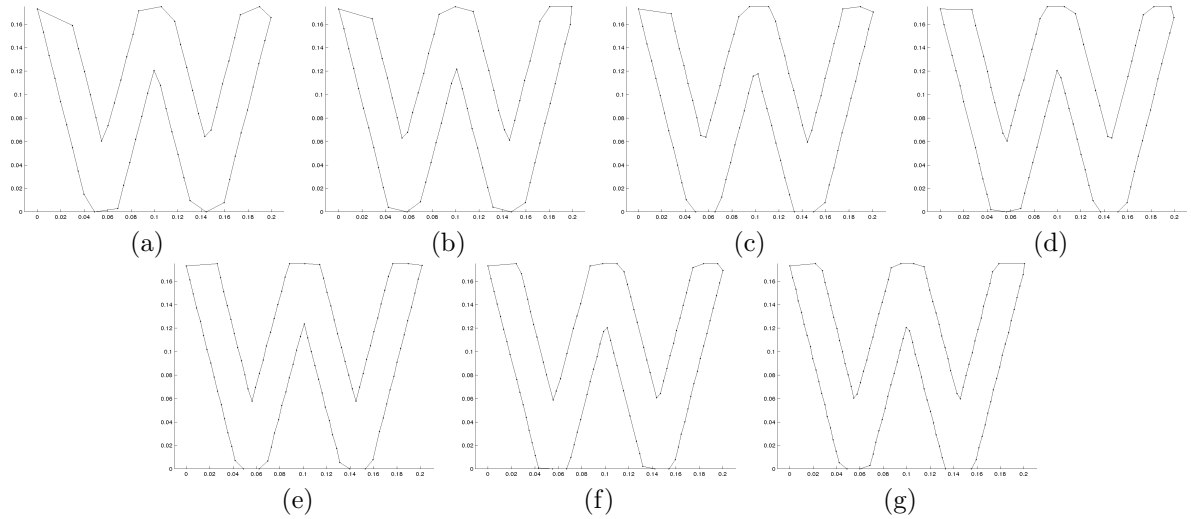


FIGURE 3.17: *Item "W" interpolated with (a) 60, (b) 70, (c) 80, (d) 90, (e) 100, (f) 110 and (g) 120 data points respectively. The resulting number of points after polygon thinning is (a) 59, (b) 68, (c) 79, (d) 74, (e) 99, (f) 86 and (g) 106 respectively.*

"a", "m" and "R" use fewer data points than what the interpolation provided, making good use of the polygon thinning technique. Finally item "W" uses close to the interpolated data points for each packing, most noticeably for data points 60, 70, 80 and 100 where the number of data points after polygon thinning are 59, 68, 79 and 99 points respectively, which is only 1 or 2 data points away from the interpolated number of data points in each case. This near full utilisation of data points occurs when the item is interpolated in such a manner that it creates an edge that appears as a collection of small zig-zag edges with small deviations. Adding more data points for interpolation for this item "W" would only cause more zig-zag edges to occur.

From this table as well as the corresponding figures for each item, it seems as if 100 data points for interpolation is a good choice for the packings. This is a good balance between having enough data points to give a fair, smooth, representation of the item, and not having too many data points that are either redundant or causes the item to consist of many small zig-zag edges which are not ideal.

In this chapter the procedures for preparing an item to be packed are discussed. Items are read into the program and processed for cleaning. Holes that exist within items are removed where applicable. The contour of each item is then identified and the number of these vertices are reduced even further via the process of interpolation. Finally, polygon thinning is used in reducing the number of vertices of each item even further.

CHAPTER 4

Methodology

Contents

In this chapter the methodology of the rule-set packing algorithm that is developed for this research, is explained in greater detail. At this point each item has been identified and converted into its graphical data representation via the display method discussed in Chapter 3. The next step is to pack the items utilising some aspects of the packing techniques discussed in §2.2. To assist in packing the items in this research project, the items are placed within certain groups depending on certain criteria. The chapter starts with this description in §4.1. The items are then packed into the strip utilising the rule-set technique algorithm described in §4.2. Finally, some items may be rotated by 180 degrees, and the process that determines which of these items can be rotated and how they are rotated, is discussed in §4.3.

4.1 Item groupings

In order to improve on the performance time of the strip packing method developed in this research, a *Rule-set* is defined which groups similar shaped items together. To illustrate the various similar shape groupings, consider for example the "M", "H" and "I" items. Each item consist of parallel vertical lines on either side, but each item varies in the space between. Where one item has a large gap, like the "H", another does not, like the "I". Due to the strip packing techniques that is used here, the variations of this nature would be negligible for the packing as each item is placed on a solid level. Thus, items like the "M", "H" and "I" share similar characteristics which groups them into the same shape category of items, called *shape group*, in this case the *block shape*, which can be seen in Figure 4.1(a). All the items in this block shape group is also shown in Figure 4.1.

Another set of items that can be grouped together is the items "V" and "W", where these items have vertical lines that start a fair distance apart and then converge to the bottom. On the other hand, the item "A" consists of two vertical lines that diverge to the bottom. The rest of the various item shape groups can be described in a similar manner as above and can be seen in Figure 4.2, where the "V" and "W" mentioned above are i n t he sh ape group (f), while "A" is in shape group (a). Table 4.1 lists which upper-case letter items fall within which shape group

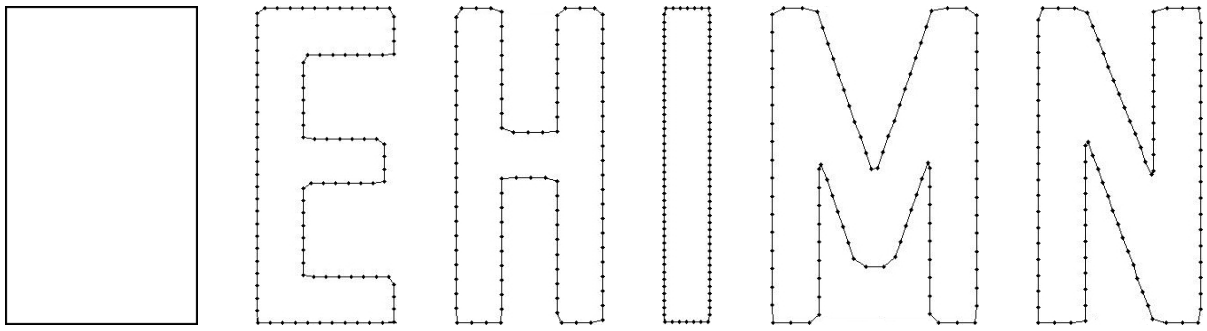


FIGURE 4.1: *The shape that resembles a block in (a) and the items that join the block shape group in (b) - (f), where all these items have the characteristic of parallel vertical lines on either side.*

found in Figure 4.2. The complete table of shape groups for the upper-case letters, lower-case letters and numbers can be found in AppendixA.

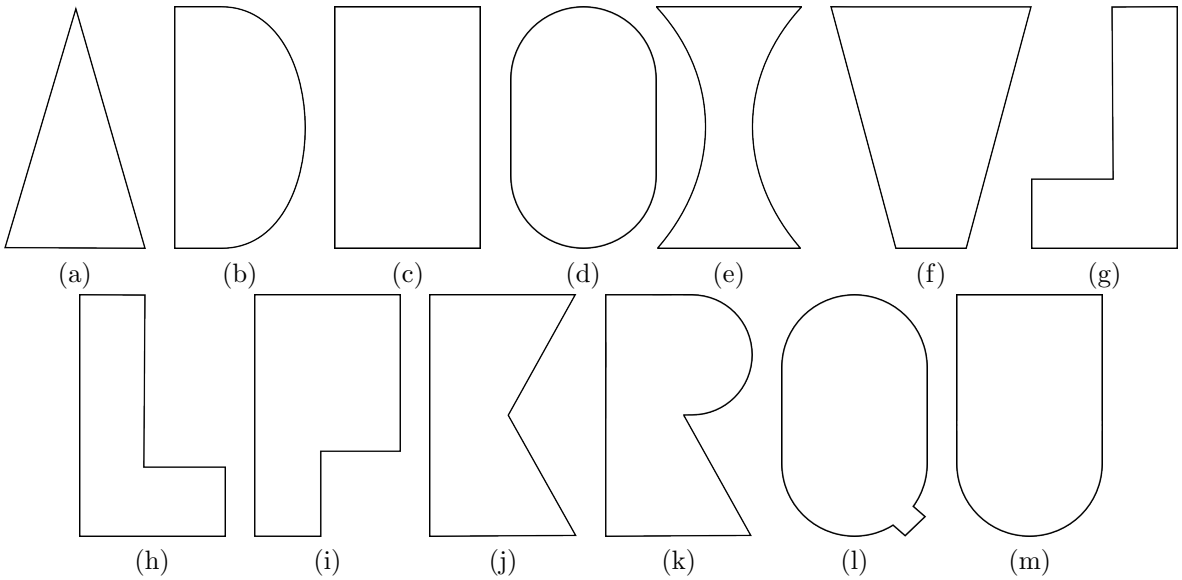


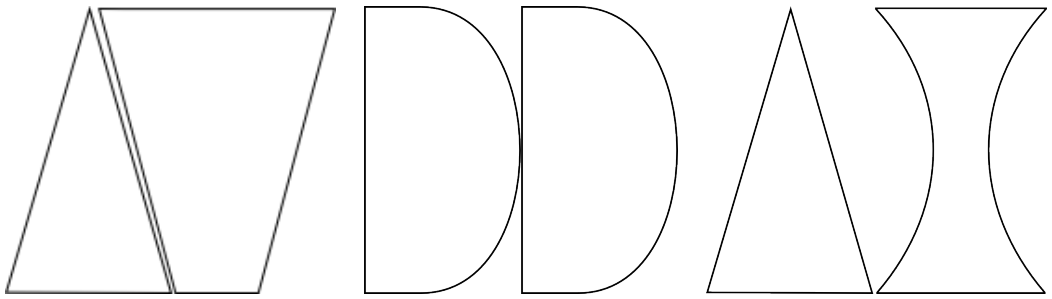
FIGURE 4.2: *The various shape groups (for upper-case letters) utilised in the rule-set technique.*

Shape Group	a	b	c	d	e	f	g	h	i	j	k	l	m
Item(s): Upper-case letter	A	B D	E H I M N	C G O S	X Z	T V W Y	J	L	F P	K	R	Q	U

Table 4.1: *Shape groups from Figure 4.2 with all items contained in each group.*

The rule-set technique allocates a value, called the *rule-set value*, for each type of item in a packing depending on what other shape group is placed adjacent to it. For example, items that fall in the shape groups shown in Figure 4.2(c) work well when placed on either side of each other, as well as being placed on the left of an item from one of the shape groups found in

Figure 4.2(b), (h), (i), (j) and (k). These items placed in this manner receive a positive rule-set value of 2 as shown in Table A.2 found in Appendix A. Another example of a good fit (when placed adjacently to each other) is an item from the shape group shown in Figure 4.2(a) and an item from the shape group in Figure 4.2(f). These shape group combinations, and many others, illustrate a very good use of the available space on offer for each of these shape groups. An "A" and a "V", for example, when placed adjacent to each other utilises space extremely well so that there is almost no space wasted, and this type of placement for items is the most ideal, thus receiving a rule-set value of 2 for both items, as seen in Figure 4.3(a).



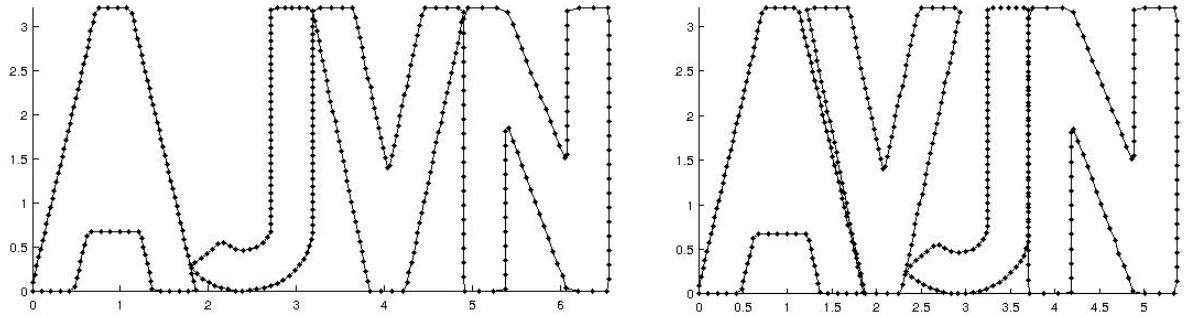
(a) (b) (c)

FIGURE 4.3: *The items (a) "A", (b)"D" and (c) "A" have a rule-set value of* 2, −1 *and* −2 *when placed on the left hand side of the items "V", "D" and "X" respectively.*

The items found in the shape group in Figure 4.2(b) works very well when placed to the left of the items found in the shape group in Figure 4.2(e), for example item "D" on the left of item "X", where there is almost no waste in space between them. Thus, in this instance the item "D" will receive a rule-set value of 2 when an "X" is placed on its right hand side. Items from other shape groups placed to the right of an item in the shape group associated with item "D" do not save as much space as well as "X" does, but are not completely wasteful either. Items that, when placed on the right of an item from the shape group associated with the item "D" do not save space as well as the examples given above, but do save some space, fall into the shape groups found in Figure 4.2(b), (c), (f), (h), (i), (j) and (k). One example of two items placed adjacent saving a favourable amount of space would be "D" and "T", since the "T" makes good use of the free space above the "D", but there exists a rather large gap at the bottom of the "D" and "T" combination to warrant some space wasted. The item "D" will receive a rule-set value of 1 as some space is saved, but there still exists some waste between the "D" and "T". Another example would be a "D" placed adjacent to another "D", where the second "D" makes use of all the space it can, but there still exist some space waste above and below the first "D" in the right-hand corners. This example of two "D"s placed adjacent to each other grant the "D" on the left hand side a value of −1 since this packing wastes quite a bit of space by the first "D" and may be seen in Figure 4.3(b).

Finally there are items from shape groups that when placed adjacent to each other waste a considerable amount of space and are therefore an unfavourable packing. An example of such a placement would be the items found in the shape group shown in Figure 4.2(a) placed adjacent to those found in Figure 4.2(e), and an example would be the items "A" and "X" where space is wasted above the "A" and in the centre of both the "A" and "X". This type of packing would grant a rule-set value of −2 for both items since a rather large portion of space is wasted between the items when either item is the left hand side item, as shown in Figure 4.3(c). It would be better to place almost any item from any other shape group adjacent to the "A" than to stay with the "X". These types of comparisons have been done for each shape group type and the final rule-set values may be found in Table A.2 in Appendix A.

In Figure 4.4 an example of two placements for a given set of items, namely "A", "J","N" and "V" are shown. In the first scenario, Figure 4.4(a), the items are placed in no particular



(a) (b)

FIGURE 4.4: *Examples of (a) a bad placement and (b) a good placement of the same 4 items, namely "A", "J", "N" and "V".*

order. When this placement is compared to the placement found in the second scenario, namely Figure 4.4(b), one can see the significant improvements in space allocation making the latter a good placement whereas the former is a bad placement. The "A" is placed first followed by the "V" which is placed adjacently on the right of it, thus saving a great deal of space between them due to the almost parallel lines for the end of "A" and the start of "V". The "J" is then placed adjacent to the "V" on its right, making use of the space below the diagonal line provided by the "V". The last item placed is the "N" as it wastes very little space between the "J" and "N".

4.2 Rule-set algorithm

The rule-set packing technique given in pseudo code in Algorithm 4.1, is a heuristic method. The detail of the algorithm as well as the working of Algorithm 4.1 will be explained with the aid of an example. Suppose the set of items to be packed is {A, F, I, J, J, L, M, P, W, Y, Y, Z}. The initial solution is arbitrarily chosen and is illustrated in Figure 4.5(a). The width left over on the first level, called the *end space waste* (ESW), for the initial solution is 0.73 units and the end space waste on the second level, is 1.74 units. The space "wasted" at the end on the final level of a packing is called the *free space* (FS) as this free space allows more items to be packed for future problems, thus, the FS is 1.74 units. There is some *wasted space* between the letters, for example, the space around the letters "W", "A", "L" and "J" on the first level of the initial packing. These wasted spaces which are added together, are called the *collective waste* (CW). These spaces are tough to categorise as the potential space they may be able to save is dependant on the available items for packing. In the initial packing of the example in Figure 4.5(a), the space at the bottom between the "M" and "W" wastes about 0.5 units of width, the space on either side of the "A" wastes collectively 1 unit at the top and the large space between the "L" and "J" wastes 2 units. Thus the collective waste on level 1 is 3.5 units. The next level considered is level 2, which is the highest level of the example packing. There is quite a bit of space between the items that is potentially wasted and from left to right on level 2 this wasted space is 1.2 units in the space that separates the items "F" and Y" at the bottom, 0.2 units between the "Y" and "P" items at the bottom, 1.15 units between the "P" and "Y" items at the bottom and finally 0.1 units separating the top of items "Y" and "J". Thus, the collective waste on level 2 is 2.65 units. For the highest level, level 2 in this case, only the wasted space between items are given since the free space at the end of the strip on level 2 *may* be used for more items to be packed.

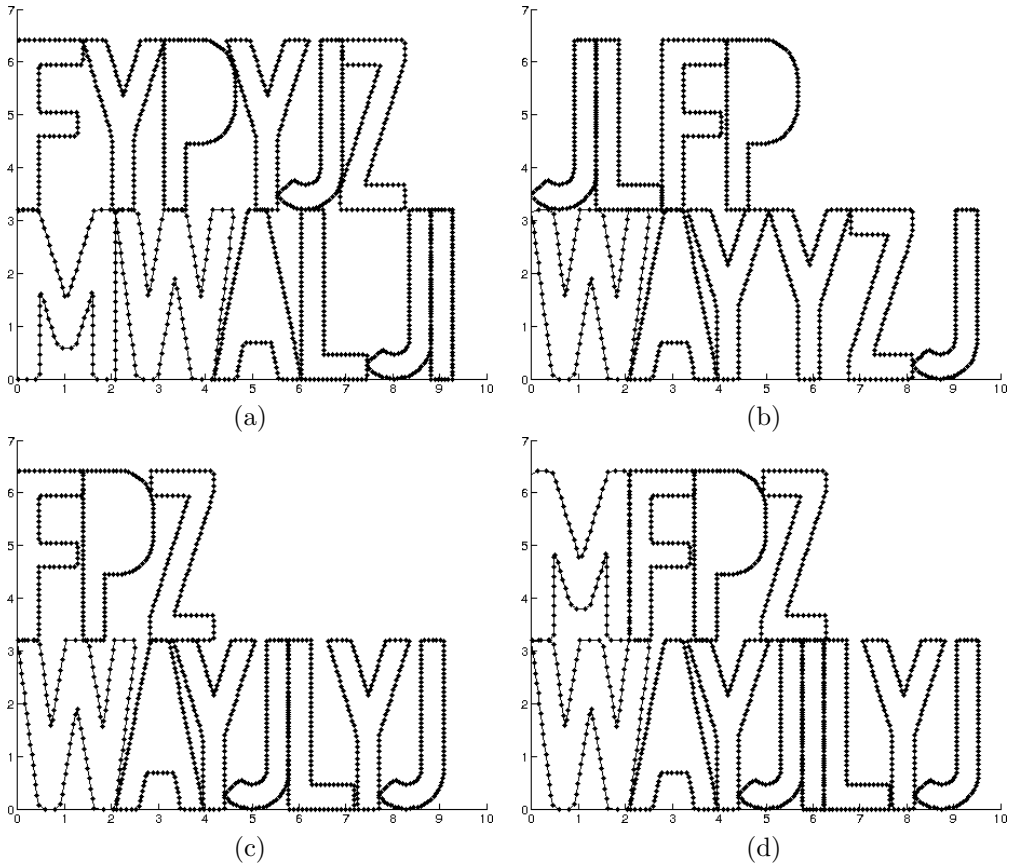


FIGURE 4.5: *Example of a rule-set packing with (a) the initial solution, (b) the packing without block items after the rule-set sorting iteration, and NFDH algorithm have already been executed (c) the packing after the bubble sort (Step 5 of Algorithm 4.1) and (d) the final packing after block shape insertion.*

The rule-set technique is applied to the initial solution shown in Figure 4.5(a). First, the items within the packing list are sorted according to the rule-set values as described in §4.1 in ascending order of favourable values (Step 1 in Algorithm 4.1) after which the block items "M" and "I" are removed from the packing list and placed into a separate block packing list momentarily (Step 2 in Algorithm 4.1). Items "W", "A", and the two "Y"s and "Z" are the most unfavourable to pack as these items waste sufficient space on either side of them, followed by the two "J"s as they are unfavourable on the left, ending with the "L", "F" and "P" which are unfavourable on the right. Figure 4.5(b) illustrates the packing procedure so far, with end space wasted on first level being 0.50 and the free space on the final level 4.33 units, however noting that the two block shape items are yet to be packed. On the left of item "W" 0.4 units of space is wasted, only 0.1 units of space is wasted at the top between items "W" and "A", 1.2 units are wasted between the two "Y" items at the bottom, 0.6 on the right of the second "Y" item at the bottom and finally 1 unit on the left of the "J" at the top. Thus the collective waste for the first level is 3.3 units. For the second level there is quite a bit of space wasted. There is 1 unit wasted on the left of the "J" at the top, 1 unit at the top on the right of the "L" item, and 1 unit each at the bottom of the "F" and "P" respectively, bringing the collective waste on this top level to be 4 units.

The items are then sorted by favourable item pairings utilising the *bubble sort* sorting algorithm which has been described in Appendix A (Step 5 in Algorithm 4.1) [19]. The first item pairing, "W" and "A", are compared to the item pairing of "W" and "Y". The "W" and "A" pairing is quite a favourable pairing and more favourable than "W" and "Y". This can be said of any

Algorithm 4.1: Rule-set Packing Algorithm.

Input : Set of items to be packed.
Output: A Packing.

1 *Allocate shape types to each item*;
2 *Sort input set of items by decreasing height, separate items into data sets of same height*;
3 *Identify items for rotation in first data set & rotate them*;
4 → newly rotated items gain new shape type;
5 *Identify & remove block shaped items from data set*;
6 *Pack items utilising NFDH strip packing algorithm*;
7 *Utilise Rule-set Packing*;
8 → step through data-set and sort items by rule-set value utilising bubble sort algorithm;
9 → repeat Rule-set Packing until bubble sort shows no improvement;
10 *insert block shaped items into packed list and shift items*;
11 *repeat steps 3 to 10 for each data set of the same height until empty*;

pairing with "W" that may occur further down the list, so the "A" is placed adjacent to the "W" as is. Moving on to the next pairing, the "A" and "Y", is quite a favourable pairing as well, so the "Y" is placed adjacent to the "A". The next item pairing are the two "Y"s placed adjacent to each other. This pairing is rather poor as well as pairing"Y" with the next item in the list, the "Z". However, pairing the "Y" with the item that follows, namely "J" is quite a favourable pairing, so "J" is placed adjacent to the first "Y" and the second "Y" and "Z" are shifted down the packing list. The second "Y", the "Z" and second "J" are all unfavourable when placed adjacent to the first "J", but the "L" is quite favourable and is placed adjacent to the first "J" and the second "Y", the "Z" and the second "J" are all shifted down the packing list. This packing process of comparing each pairing with the next possible pairing is repeated throughout the packing list until all items are packed. Figure 4.5(c) shows the packing after the bubble sort of pairings are completed. The end space waste on the first level is 0.92 and the free space on the final level is 5.82 units respectively. There is only minimal space wasted for these packed items. Again, on the left of item "W" 0.4 units of space is wasted, only 0.1 units of space is wasted at the top between items "W" and "A" as well as between items "Y" and "J", and finally the space wasted on either side of the last item "Y" is a combined 0.36 units, which is a collective waste of 0.96 units. The second level wastes space at the bottom of items "F" and "P" with a collective waste of 2 units, however it is important to note that this packing is quite compact given the item set for packing. Keeping in mind that two block items must still be packed, this is already an improvement on the initial packing.

Block items "I" and "M", having since been sorted by width, are inserted into the packing at locations where block items would generate still favourable pairings (Step 6 of Algorithm 4.1). The "I", for example, is inserted into the first location that is favourable, which is between the first "J" and "L". Placing the "I" between the first "J" and "L" does not change the favourable status of the "J" and "L" since the "J" and "I" pairing and "I" and "L" pairing give the same favourable pairing values. The "M" is inserted in a similar manner into the location between the second "J" and the "F". The resultant packing can be seen in Figure 4.5(d) where 0.47 units are end space waste on the first level. It is important to note that the top most level has 3.73 units free space which allows for more items to be placed there if need be. When the initial packing's end space waste of 0.73 units on the first level and free space of 1.74 units on the top level, is compared to the end space waste of the final packing which is 0.47 units on the first level and free space of 3.73 units on the top level, one can see a significant collective improvement of 1.73 units. The collective waste for the first packing between the items on the first level is 2.62

units, whereas this collective waste for the top level is 3.56 units. Compare this to the collective waste of the final packing for the first level, which is 1.07 units, and the top level, which is 1.76 units, we see a significant improvement of 1.54 units on the first level and 1.80 units on the top level respectively. Furthermore one can see that the items are placed as tightly as possible and that there is more free space on the top most level for additional items not yet considered for packing.

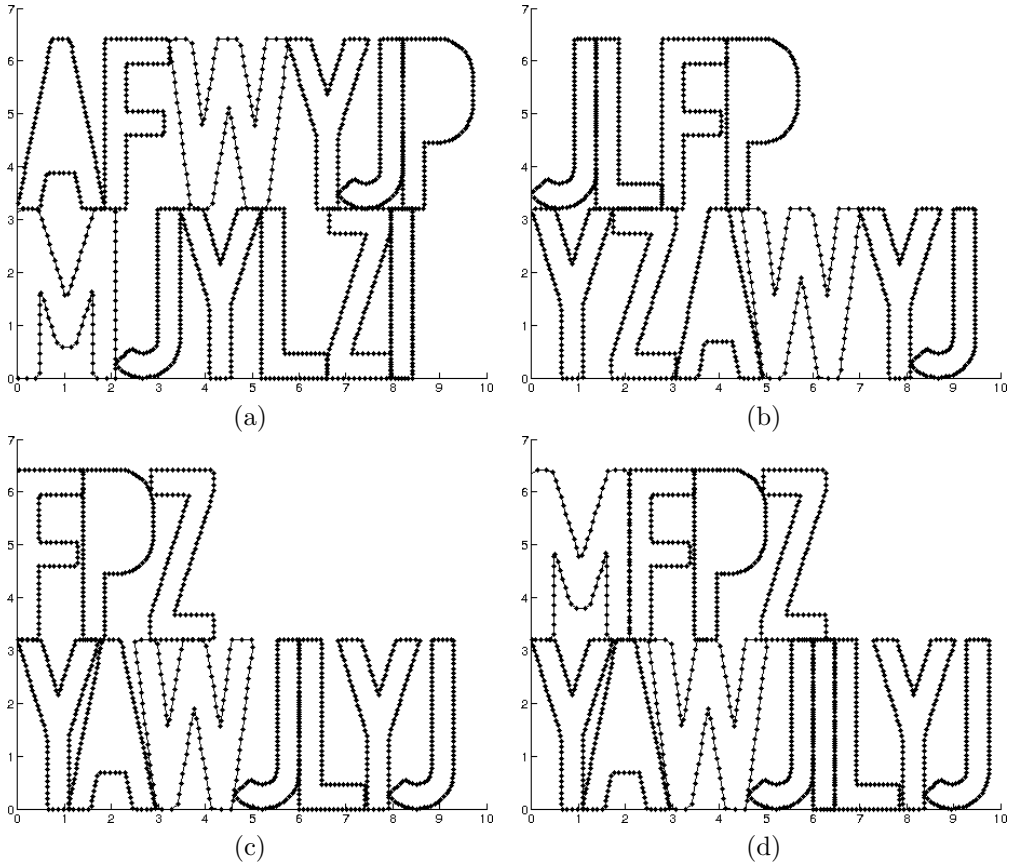


FIGURE 4.6: *Example of a rule-set packing for the same set of items than in Figure 4.5, but with (a) a different initial solution, (b) the rule-set sorting iteration and NFDH packing, (c) packing iteration utilising the bubble sort technique and (d) the final packing after the block shape items have been inserted.*

The outcome of Algorithm 4.1 is dependant on the initial solution of the items packed. To illustrate this, suppose the same set of items must be packed, but the order in which the items are initially placed is different. This initial solution variant has also been chosen arbitrarily and can be seen in Figure 4.6(a) with 1.57 units of end space waste on the first level and 0.28 units of free space left over on the top level. The collective waste of this initial solution variant is 2.62 units on the first level and 3.56 units on the top level respectively. Figure 4.6(b) shows the items sorted by the rule-set technique, Figure 4.6(c) shows the packing of compared pairings using the rule-set values and Figure 4.6(d) shows the final packing with the block items inserted. The final packing has an ESW of 0.249 units and a FS of 3.73 units respectively. The collective waste in the final step of this second example utilising the same item set, is 1.07 units for the first level and 1.76 units for the second level. Comparing these values with those from the first example there is not much of a difference between the two, where the space saved on the first level of the first example is a little more. The difference in the packings of the two examples is because the pairing of "Y" and "J" is better than the "W" and "J" pairing for wasting space. Thus,

although the examples utilise the same rule-set technique, they have variations in the outcome solutions due to the initial solutions utilised.

4.3 Rotating items

So far, only items that have not been rotated were used in the packing algorithm. In this section possible rotations and the method with which items are rotated, are discussed. Depending on the items to be packed, it may be advantageous to rotate some items by 180 degrees in order to obtain a tighter packing of items than before. An example of this was illustrated in Figure 1.3(b) in §1.1, where significant space may be saved by placing two "L"'s adjacent to each other, with the one "L" having been rotated 180 degrees. In this thesis rotations of 180 degrees only are considered due to the lines on the reflective layer of vinyl as described in §1.3. When packing many items into the strip, it is important to determine which items should be packed in its original orientation, and which items should be rotated. The items that form part of a shape group that, when rotated 180 degrees, become a shape group that already exists within the predefined shape list, are the items that may be rotated. Some items, like the letter "A" for example, is in the Figure 4.2(a) shape group, however once the item has been rotated 180 degrees, it becomes an item that falls within the shape group found in Figure 4.2(f). Items "A" and "J", and their rotated counterparts, can be seen in Figure 4.7.

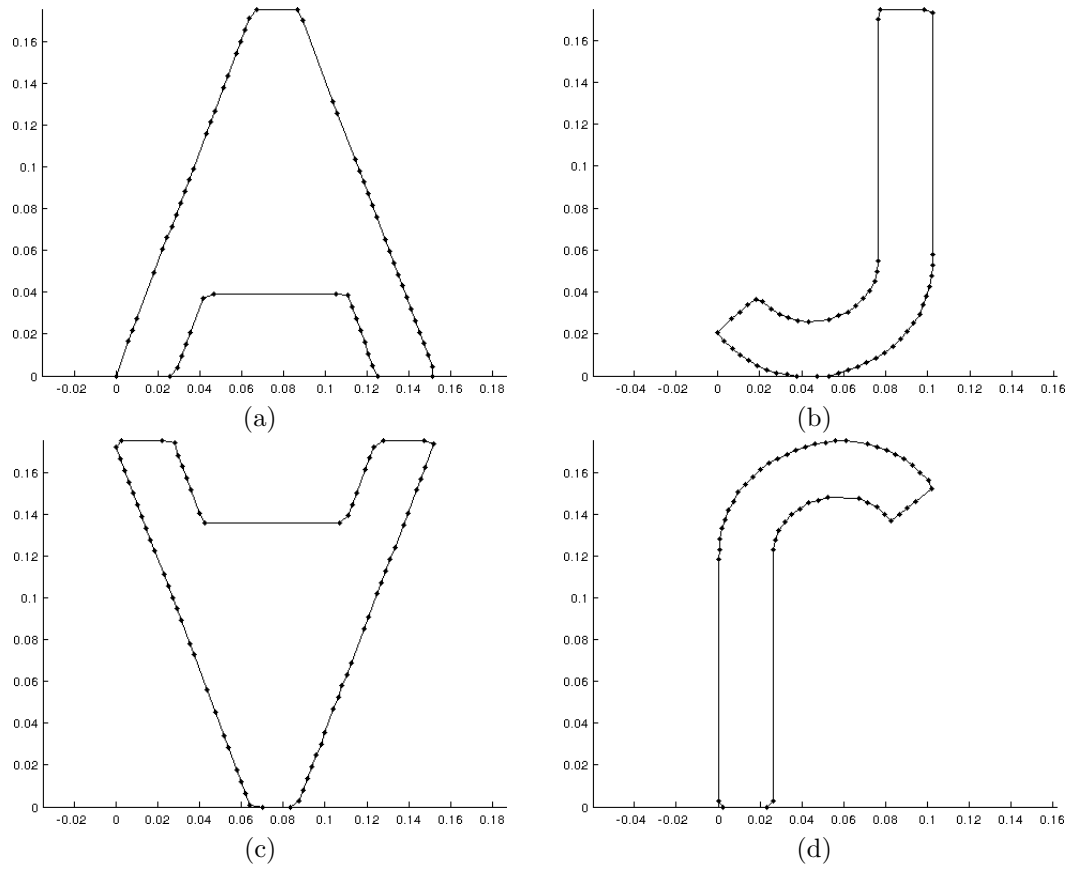


FIGURE 4.7: *Examples of items (a) "A" and (b) "J" and their rotated counterparts in (c) and (d), respectively.*

The important step in the rotation process is to decide which items should be rotated. Rotating

all possible items for the sake of rotations will not necessarily improve the packing and may in fact result in a poorer packing. The rotation procedure used here, initially identifies and counts each item that may be rotated. Items that have space available at the top are, for example, the shape groups shown in Figure 4.8 (a) to (c). The rotated counterparts of these shape groups

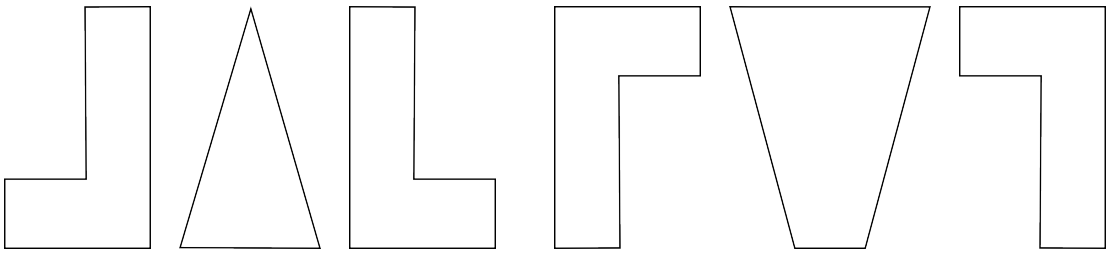


FIGURE 4.8: *(a) to (c) Shape groups that have space available at the top and their rotated counterparts in (d) to (f) that thus have space available at the bottom and visa versa.*

are given in Figure 4.8 (d) to (f) and thus items that fall within these shape groups have space available at the bottom. The items from shape groups that have space available at the top are placed in an item set, called the *top set*, and are counted. The same is done for the items from the shape groups that save space at the bottom, the *bottom set*. If the number of top set items that have been counted outnumbers [is outnumbered by] the number of bottom set items by two or more then an item from the top set is rotated [an item from the bottom set is rotated]. This rotating procedure is repeated until the number of top set items and the bottom set items differ by at most one. These shape groups are the only ones considered to be rotated since they have the greatest possibility of improving on the packing, and their rotated shape groups already exist within the predefined shape groups. The shape group from Figure 4.8(f) (which was not part of the list of shape groups from Figure 4.2) has been added for the lower case letter "q".

To illustrate the potential space savings that this rotation process can produce, two example packings of twenty copies of the letter "J" are shown in Figure 4.9. In Figure 4.9(a) the twenty

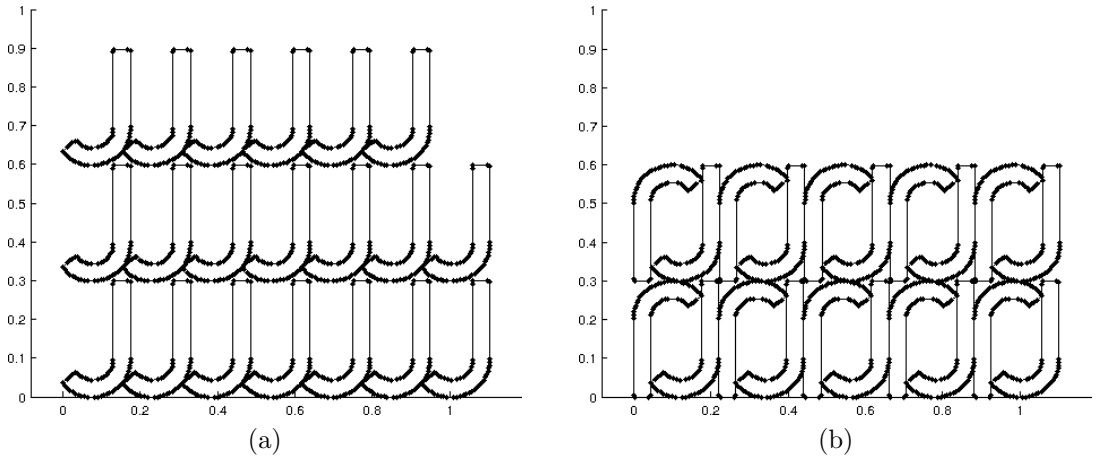


FIGURE 4.9: *Example of twenty "J"s (a) not rotated and (b) rotated, illustrating the potential space savings that rotations might bring.*

copies of "J" are placed as is. This packing results in three levels of packed items and a lot of space wasted between every pair of "J"s. In Figure 4.9(b) the same set of twenty "J"s are packed, however, this time some of the "J"s are rotated. Half of the "J"s have been rotated utilising the rotation procedure defined earlier and produces only two levels of items, which is

one level less than the non-rotated set of items packed. The space between each "J", rotated and otherwise, is utilised in the best possible manner.

In this chapter the methodology for the placement of items is discussed. Items are grouped into shape groups that share similar shape properties. These shape groups are assigned values with relation to shape groups adjacent to them. Some items are rotated 180 degrees and moved into different shape groups. These items that have been grouped into shape groups are sorted and packed utilising the newly developed rule-set algorithm.

CHAPTER 5

Results

Contents

In this chapter the results of the rule-set packing algorithm applied to various benchmarks are given. Each benchmark consists of items with differing font types, differing text heights, as well as whether or not rotations are allowed for the packing strategy. The details of these properties are discussed in §5.1. The benchmarks are divided into actual item lists obtained from Kohler Signs presented in §5.2, and randomly generated benchmarks presented in §5.3. The chapter is concluded with a summary of the findings of all the benchmarks.

5.1 Benchmark properties

In the road sign manufacturing industry various fonts are used in the production of road signs, depending on what type of sign is being manufactured. At Kohler Signs, three types of fonts are utilised namely *DinA*, *DinB* and *ModB* and may be seen in Figure 5.1 [14]. The items in

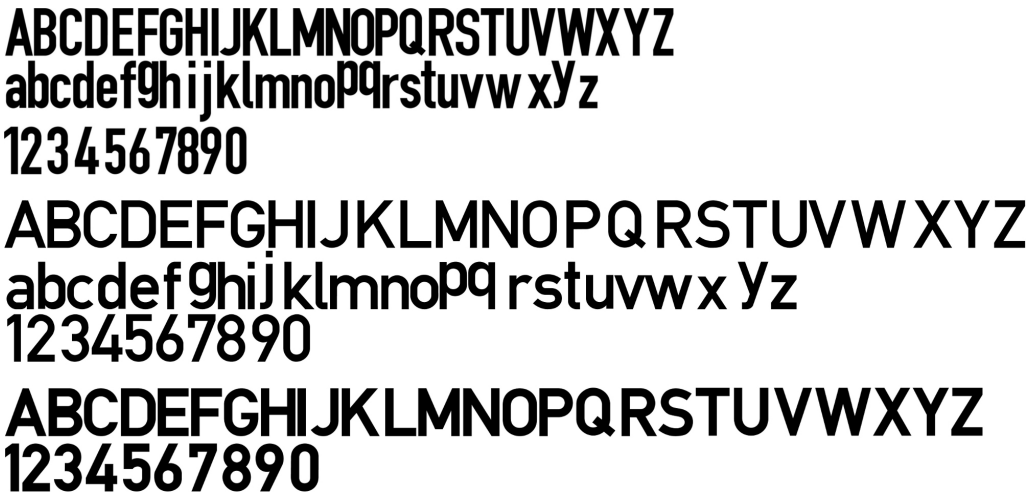


FIGURE 5.1: *Font types used by Kohler Signs, from top to bottom, DinA, DinB and ModB respectively.*

the *DinB* font appears the same as those in font *DinA* except the items in *DinB* are wider. The *ModB* font is a bolder version of *DinB*. The fonts *DinA* and *DinB* consist of upper-case letters, lower-case letters and numbers, whereas the *ModB* font only consists of upper-case letters and numbers.

The text items on a sign also consist of a set text height of one of 8 possible heights. In Figure 5.2 these different heights for the letter "A" are illustrated, starting from 420mm followed by 350mm, 280mm, 210mm, 175mm, 140mm, 112mm and lastly 105mm. The *text height* indicates the maximum height that any item from the particular font may be. For example, the upper-case

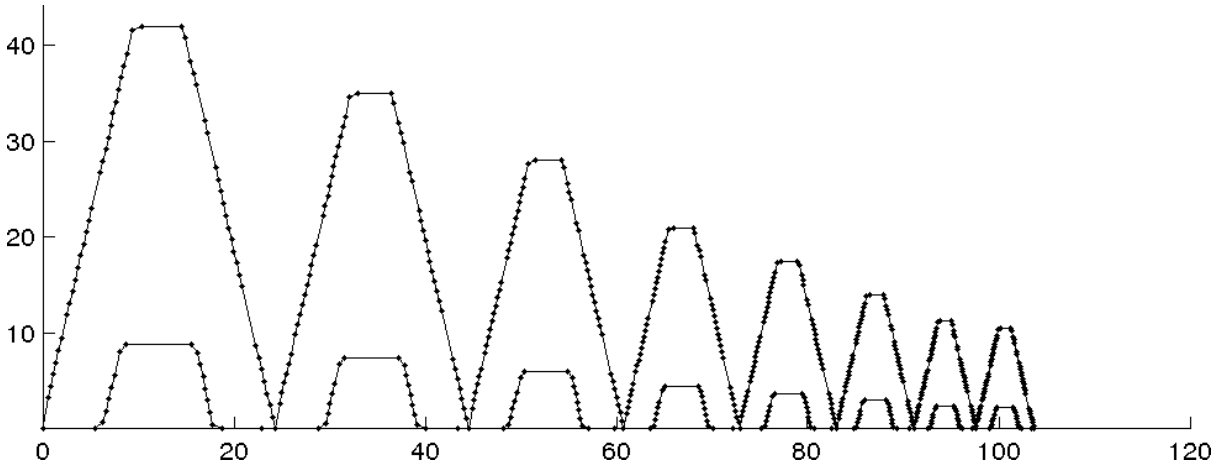


FIGURE 5.2: *The letter "A" with different text heights, from 420mm for the first item, decreasing to 105mm for the last item.*

letter "Z" will be the max height for a particular font say 420mm, but the height of the lower-case letter "z" will not be 420mm but rather 300mm. Other items, like the letter "Y" will be the same height for its upper-case and lower-case types. Both of these cases are illustrated in Figure 5.3. Therefore, the maximum height of all items to be packed in that instance determine the text height to start with and the shorter items to be packed are scaled accordingly. When

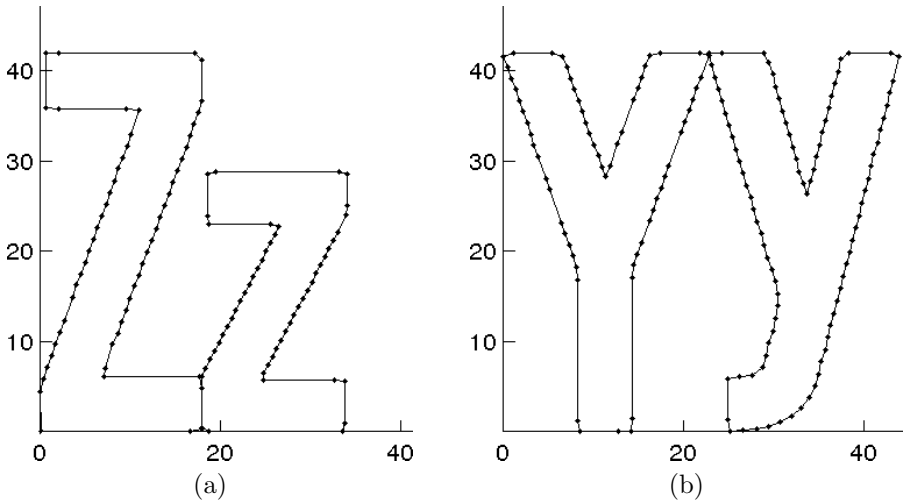


FIGURE 5.3: *Examples of the possible difference in height for upper-case and lower-case letters for (a) the letter "Z" where the heights differ, and (b) the letter "Y" where the two heights are the same.*

Kohler Signs produces a sign they include not only the text items that need to be placed upon the sign but also various other types of images like arrows and borders. These "other" images are

not included in the packing process for the rule-set algorithm but can easily be added afterwards by the user. Items like the letter "i" and "j" consist of two pieces, namely the *letter stroke* and *letter point* and may be packed, according to the algorithm, as two separate entities. However the letter point of these types of items are so small that they may be added freely after execution of the packing algorithm in any open spaces. The letter stroke of the item "i" is the same height as the other lower-case items for its font type, and the letter stroke of the item "j" is the same height as the other upper-case and tall lower-case items of the same font type, which justifies separating the letter points from the letter strokes even further. Thus only the letter stroke parts of these types of items are packed with the rule-set packing algorithm.

The text heights and fonts are fixed for each particular sign manufacturing job that Kohler Signs have to complete. For each benchmark a *non-rotation* and a *rotation case*, as described in §4.3, will be calculated and compared to determine if there are indeed any merits to allowing rotations for the placement of items in the packing problem. Test cases will thus consist of text heights, fonts and allowed rotations from examples that constitute real world applications supplied by Kohler Signs as well as randomly generated computer examples. The properties by which a "good" packing is defined, are the ones that minimise waste as well as computational time. Waste is minimised by the 3 parameters which were described in Chapter 4, namely the end space waste (ESW), the free space (FS) and the collective waste (CW), respectively. In Figure 5.4 the difference between a bad and good packing in accordance with these 3 packing measures, are illustrated. All the items have a height of 420mm and are packed in Figure 5.4(a)

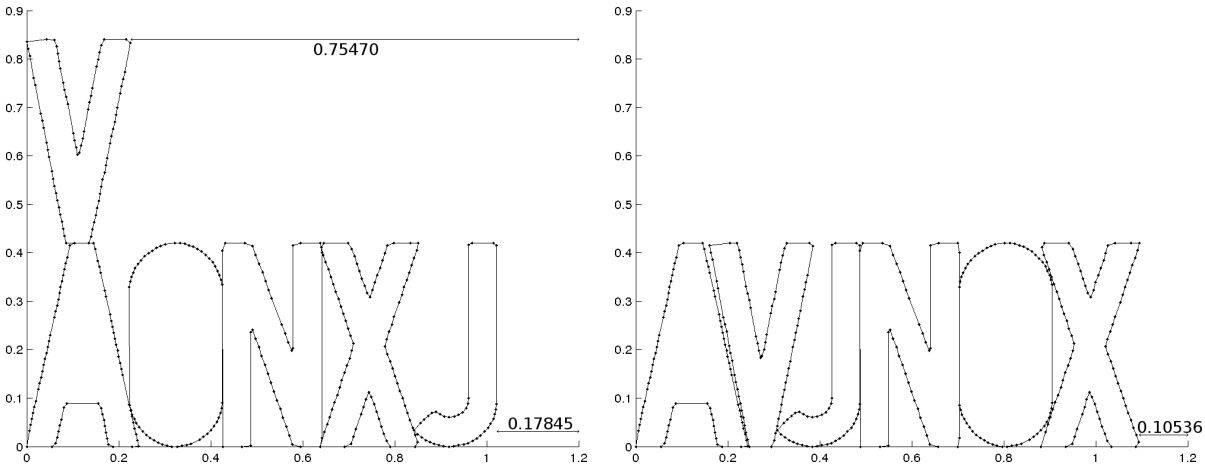


FIGURE 5.4: *Two packings indicating the differences in the 3 packing measures. The bad packing in (a) has an ESW of 178.45mm, FS of 754.7mm and CW of 353.5mm on the first level and nothing on the second. The good packing in (b) has an ESW of nothing (since there is only one level), FS of 105.36mm and CW of 143.23mm respectively.*

in a bad packing and in Figure 5.4(b) in a good packing. The bad packing has an ESW of 178.45mm, FS of 754.7mm and CW of 353.5mm on the first level and nothing on the second since it is the only item on that level. The good packing however does not have an ESW since there is only one level and thus the FS for this packing is 105.36mm and the CW is 143.23mm. Two levels are utilised for the bad packing for all the items, whereas in the good packing the items are placed more compactly, which results in only one level being utilised. The good packing improves on the bad packing by a whole level since the "V" in the bad packing is just too wide to fit on the first level of its packing. In terms of the collective waste comparison of the two packings, the CW of the good packing improves on the bad packing by 210.27mm. To illustrate that value more clearly, note that the "V" in this example, has a width of 224.53mm while the

good packing saves a CW waste of 210.27mm which is almost as much space as the item "V" would use. This leads to the "V" being on its own level in the bad packing.

The numerical computing environment Matlab version R2011b[23] is used to implement the packing algorithm as well as the image processing of the letters and images. The computer used to run these simulations has an Intel Core i7-3770 CPU running at 3.40GHz with 7.7Gb of RAM, with an operating system running 64-bit Ubuntu version 12.04, with a Kernel running Linux version 3.2.0-31.

5.2 Kohler Signs benchmarks

The packing of items on the vinyl is done by hand by the graphic designer at Kohler Signs. Large text height items to be packed takes the graphic designer roughly 5 minutes per metre of running vinyl to pack and the items that have a small text height takes roughly 12 minutes per metre of running vinyl to pack [14]. The number of items to be packed also influences the packing time for Kohler Signs in a negative way, *i.e.* the more items there are to be packed, the longer the average time per item it takes the graphic designer at KS to pack the items.

The benchmark given in Figure 5.5 by Kohler Signs, took roughly 20 minutes to generate and were determined by hand by the graphic designer. The packing solution that Kohler Signs have generated is shown in Figure 5.5(a), while the two solutions for the same set of items determined by the rule-set technique with no rotations allowed and rotations allowed is shown in Figure 5.5(b) and (c), respectively. It is important to note that in this solution for KS, the height is the fixed dimension of 1200mm and the width varies, which is why the KS packing has been rotated by 90 degrees, so that the fixed dimension in these examples are on the same axis. Kohler Signs' solution is 1329mm long and took 20 minutes for the graphic designer to determine. The first solution determined by the rule-set technique, which prohibits rotations, took 4 minutes 35 seconds to run and is 1080mm long whereas the second solution, which allows rotations, took 5 minutes 9 seconds and is 1030mm long. This shows an improvement in computation time of roughly 15 minutes. Figure 5.5 illustrates the improvements of the rule-set technique algorithm for rotations prohibited and rotations allowed. The packing where rotations are allowed improves on the packing that prohibits rotations since allowing rotations places items more compactly, as illustrated in Figure 5.5 where the item "d" which starts the fourth level in the packing where rotations are prohibited, but in the figure where rotations are allowed the items are packed more tightly so that the "d" is placed in the third level, thus saving space equal to the difference in height of tall and small items.

Since the vinyl printer cuts vinyl sections off in a straight horizontal line, this difference in space saved is saved for a whole level of the vinyl. The graphic designer can add items that are not included into the rule-set algorithm by hand at the end. Figure 5.5 illustrates the effect of adding border items (quarter semi-circle shapes) to the resultant packing solution by hand, with 2 border items being placed within the second level "L" in the non-rotation result and the rest on a new level at the top whereas 4 border items are placed at the end of the last level of the rotations permitted result and the rest within a new level at the top. These additions result in 1180mm from the roll of vinyl in the rotations prohibited case and 1130mm from the roll of vinyl in the rotations permitted case, which is an improvement on the Kohler Signs packing using 1329mm from the roll of vinyl.

Two more examples received from Kohler Signs are given in Figure 5.6 and Figure 5.7 respectively. In these benchmark cases 67 items and 152 items, respectively, need to be packed. The case displayed in Figure 5.6 took the rule-set algorithm with rotations allowed 4 minutes and 45

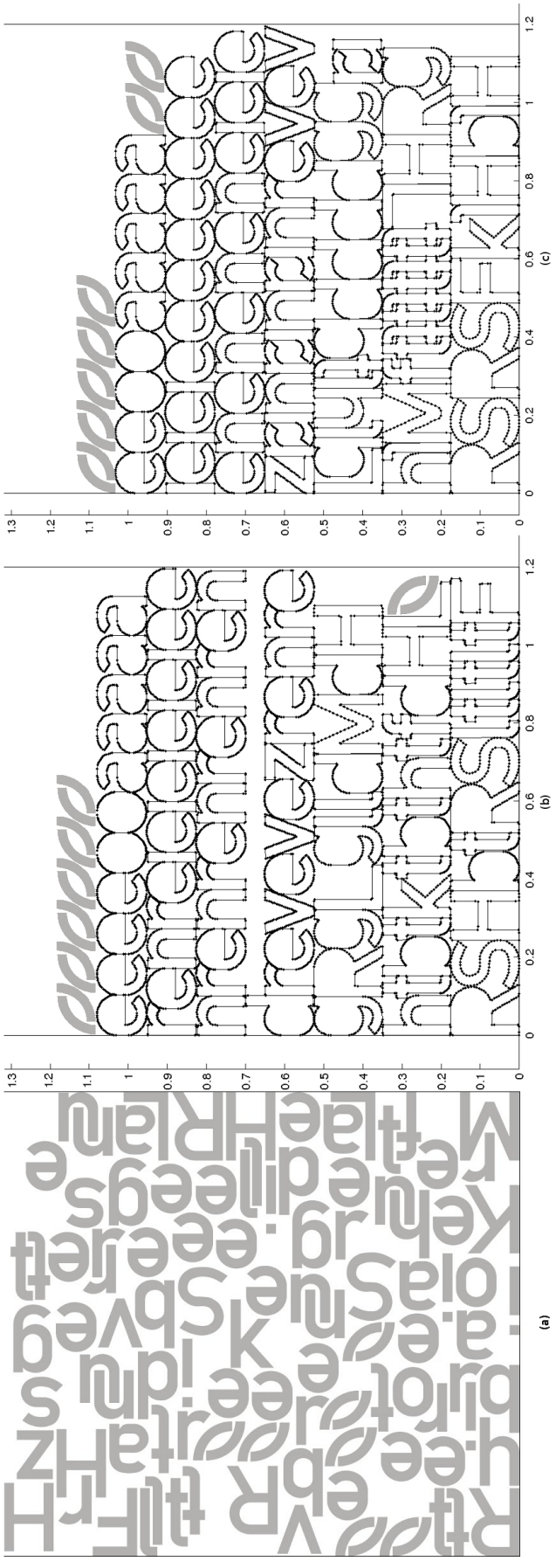


FIGURE 5.5: *A packing solution that (a) Kohler Signs has generated as well as the (b) not rotated and (c) rotated solutions from the rule-set technique for the same set of items. The fixed dimension is 1200mm wide and the height of the vinyl used in (a) is 1329mm, (b) is 1180mm and (c) is 1130mm, respectively. The darker items in (b) and (c) were added by hand after the execution of the rule-set algorithm.*

seconds to solve and the one displayed in Figure 5.7 took the rule-set algorithm 10 minutes and 40 seconds to solve, which comparatively took the graphic designer of KS roughly 15 minutes to do the first benchmark and 30 minutes for the second. The Kohler Signs solution utilises 1400mm of vinyl for the first benchmark and 3000mm in the second benchmark. The solution obtained from the rule-set algorithm utilises 844mm in the first benchmark and 1891mm in the second benchmark. However, it is important to note that the rule-set algorithm only takes into

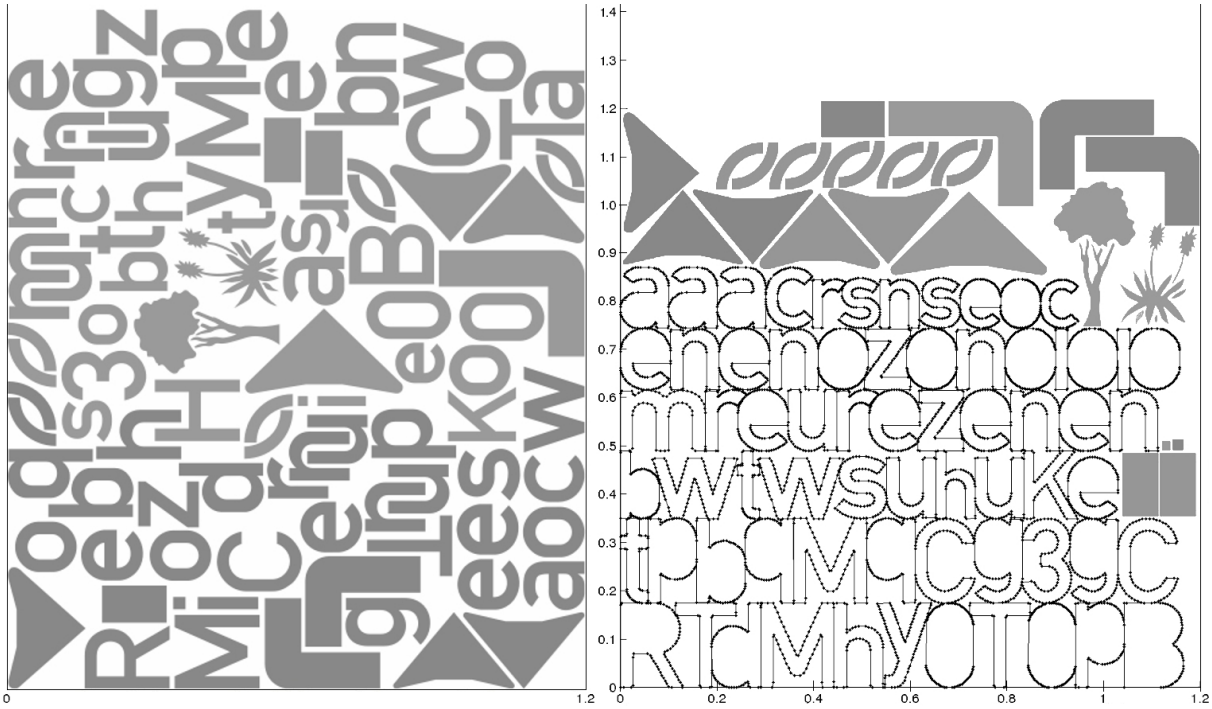


FIGURE 5.6: *A smaller packing solution that Kohler Signs has generated as well as the rotated solution from the rule-set technique. As in Figure 5.5 the fixed dimension is on the horizontal axis and is 1200mm wide. The Kohler Signs solution uses 1400mm of length of the vinyl roll, while the rule-set technique uses 1216mm.*

account the items that form part of the predefined alphabet, and not the symbols like the arrowheads and bends. The graphic designer can add these symbols to the packing by hand after the rule-set algorithm had been executed, or these symbols can be added to the known alphabet in future work. As seen in Figure 5.6 and Figure 5.7, respectively, the addition of symbols by hand to the packings result in improved packing solutions of 1216mm and 2597mm vinyl being utilised respectively. This results in a space saving of 184mm in the first benchmark and 403mm in the second, respectively, in comparison to the KS solutions.

5.3 Randomly generated benchmarks

Besides the 3 examples from KS given in §5.2, some random lists of items to be packed were also generated. In these benchmarks, an arbitrarily chosen number of items are packed in each case. The text height as well as font type is also chosen arbitrarily for each item within each randomly generated benchmark. Note that the *DinA* and *DinB* fonts consist of 62 items to be selected from and the *ModB* font consists of 36 items to be chosen from. There are 8 different universal text heights to select from, which range from 105mm to 420mm. Two types of packings, one that does not allow the rotation of items, and the other that does, are also considered in this

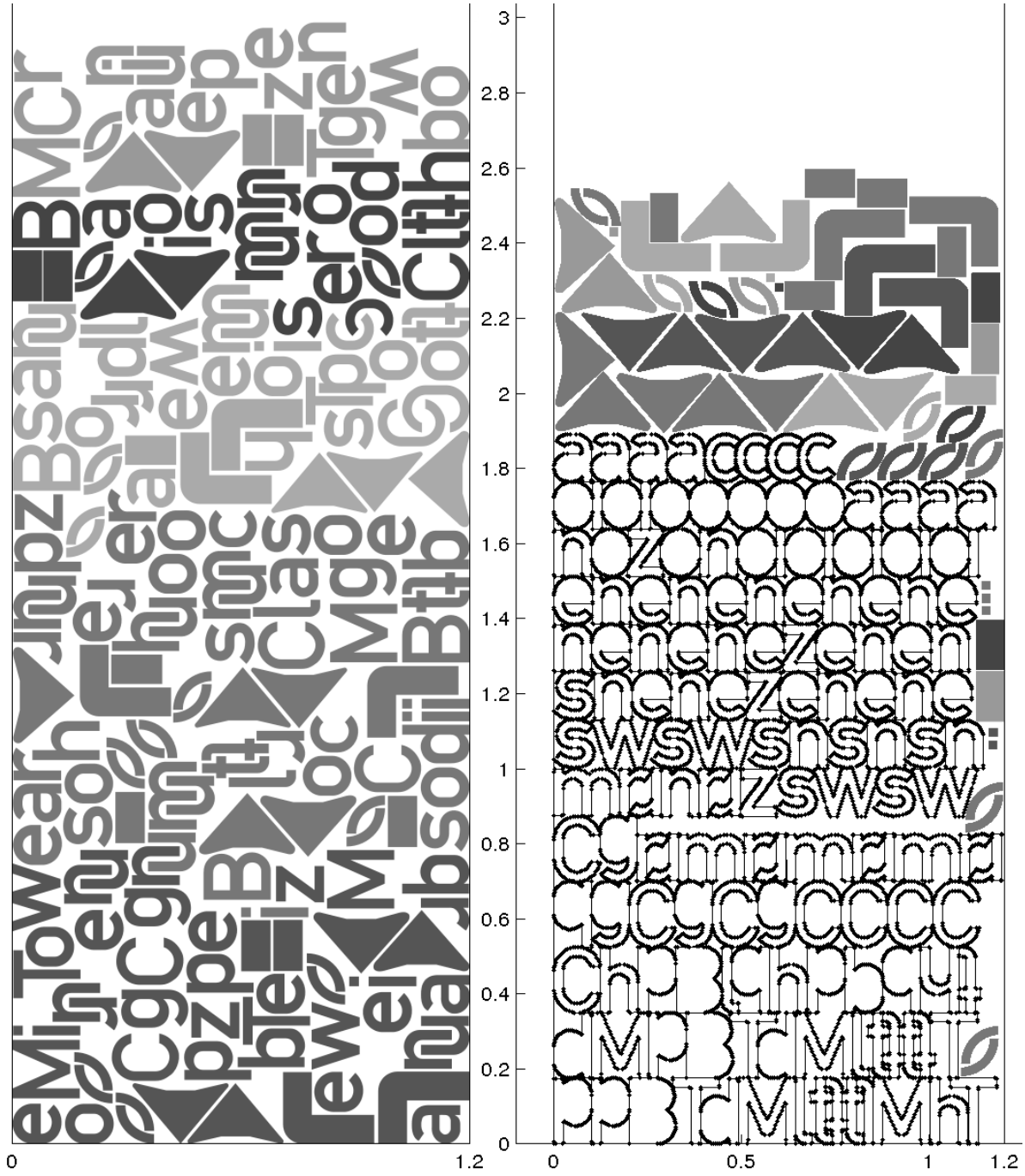


FIGURE 5.7: *A large packing solution that Kohler Signs has generated as well as the rotated solution from the rule-set technique. Again the fixed dimension is 1200mm and indicated on the horizontal axis. The Kohler Signs solution uses 3000mm of length of the vinyl roll, while the rule-set technique uses 2597mm.*

randomly generated benchmark formulation. Since KS pack items for cutting by hand and not in level algorithm approach, it is important to test the rule-set algorithm to other benchmark tests that are packed in a similar manner, to see how the algorithm compares.

First, the same set of randomly generated items of the same font, are generated for each text height to illustrate the difference that the text height might do. For all these cases, rotations were first prohibited and then allowed for the rerun of the specific case. Thus, 16 examples of

the same randomly generated item set, “v”, “8”, “v”, “f”, “7”, “I”, “Z”, “x”, “y”, “0”, “y”, “x”, “W”, “p”, “S”, “v”, “o”, “x”, “g”, “3”, “r”, “w”, “l”, “g”, “A”, “3”, “I”, “4”, “7”, “q”, “j”, “L”, “x”, “3”, “T”, “P”, “m”, “p”, “B”, “W”, “I”, “g”, “A”, “8”, “x”, “M”, “c”, “E”, “G” and “X” are compared, *i.e.* for each of the text heights 105mm, 112mm, 140mm, 175mm, 210mm, 280mm, 350mm and 420mm and for both prohibited and allowed rotations. The length of vinyl used by the packings range from 0.21m for the case with items of the text height 105mm, to 3.411m for the case with text height 420mm. These packings are illustrated in Figure B.1 to Figure B.4 in Appendix B. As an example the packings with rotations allow for the cases with text height 105mm and 210mm are displayed in Figure 5.8 In each of these benchmarks the

FIGURE 5.8: *Example packings for the same set of items where rotations are allowed, with text heights of 105mm (a) and 210mm (b) respectively.*

space saved by utilising rotations may be seen. There exists more free space or even a whole level free by allowing some items to be rotated, as evident in Figures B.1 - B.4(b) and (d) in comparison with Figures B.1 - B.4(a) and (c) in Appendix B.

In order to test the algorithm, 5 instances were generated for each of the cases where 10, 20, 40, 60, 80, 100 and 150 items need to be packed. In each case the items to be packed were randomly chosen from the arbitrarily chosen font type *DinB* and text height 175mm. In all 35 cases only the rotational packings were considered. Table 5.1 to Table 5.11 display, for the initial packing as well as the final packing, the end space waste (ESW), free space (FS), collective waste (CW) and the running time for each simulation of the various randomly generated item sets. The final packings for these solutions are illustrated in Figure B.5 to Figure B.11 in Appendix B.

The final solutions FS is better than the initial FS for all test cases above. The only time this is not the case is when the initial solution utilised an additional level for packing, resulting in more

FS but in doing so generating more wasted space overall. Analysing the CW values for the initial and final packings, it may be seen that the final packings CW is much lower than the initial solution for every test case. On average the CW of the final packing is 64.21%, 69.82%, 60.77%, 60.83%, 61.67%, 64.16% and 66.49%, respectively, of the CW of the initial packing for the case where 10, 20, 40, 60, 80, 100 and 150 items should be packed, respectively. This equates to an average improvement of 63.99% CW. Lastly, the ESW for the final solution performs better than the initial solution in most cases. Emphasis is more on the number of levels to be packed and overall space saving, than the ESW per level, but it is still a good indicator nonetheless.

10 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	final	percent	
1	n/a	n/a	110.98	154.34	387.06	256.77	66.34%	0.7090
2	n/a	n/a	90.92	207.26	582.25	318.71	54.74%	0.6998
3	n/a	n/a	181.11	267.90	371.43	183.23	49.33%	0.8133
4	n/a	n/a	172.33	185.57	377.17	314.84	83.47%	0.6438
5	n/a	n/a	51.65	82.56	363.19	243.87	67.15%	0.5910
average							64.21%	0.6914

Table 5.1: *The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 10 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 5 instances may be found in Figure B.6 in Appendix B.*

20 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	final	percent	
1	49.97	22.09	529.38	617.72	1) 249.08 2) 209.82	1) 232.26 2) 50.32	1) 93.25% 2) 23.98%	1.5900
sum					458.9	282.58	61.58%	
2	53.65	88.11	219.39	269.79	1) 600 2) 390.18	1) 400 2) 318.71	1) 66.67% 2) 81.68%	1.2132
sum					990.18	718.71	72.58%	
3	73.46	1.69	190.81	460.78	1) 415.76 2) 296.97	1) 261.94 2) 131.61	1) 63% 2) 44.32%	1.8658
sum					712.73	393.55	55.22%	
4	124.03	23.50	256.50	419.40	1) 397.67 2) 205.81	1) 367.74 2) 153.55	1) 92.47% 2) 74.61%	1.8585
sum					603.48	521.29	86.38%	
5	77.62	6.68	166.38	309.80	1) 146.11 2) 420.36	1) 234.84 2) 180.65	1) 160.73% 2) 42.98%	1.2487
sum					566.47	415.49	73.35%	
average							69.82%	1.5552

Table 5.2: *The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 20 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 5 instances may be found in Figure B.6 in Appendix B.*

The various real world and randomly generated benchmarks illustrate that the rule-set algorithm saves significant production time and vinyl space. Saving roughly 10-15 minutes per production order allows the graphic designer to process more orders which would also increase productivity. The rule-set algorithm also simplifies the cutting process for the vinyl cutters, since the items are placed upon horizontal levels that are guillotinable, which could allow for a simpler and faster cutting process for the cutters.

In this chapter the results of packing items utilising the rule-set algorithm have been presented.

40 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
1	1) 63.51 2) 27.83 3) 43.53	1) 13.21 2) 37.00 3) 0.16	485.69	700.06	1) 457.5 2) 316.25 3) 408.75 4) 203.75	1) 232.26 2) 320.00 3) 264.52 4) 116.13	1) 50.77% 2) 101.19% 3) 64.71% 4) 57%	2.6523
sum	134.87	50.37			1386.25	932.91	67.3%	
2	1) 98.12 2) 97.81 3) 39.34	1) 70.78 2) 86.88 3) 5.37	83.52	404.55	1) 197.52 2) 376.4 3) 429.81 4) 345.34	1) 221.94 2) 264.52 3) 234.84 4) 78.71	1) 112.36% 2) 70.28% 3) 54.64% 4) 22.79%	3.1717
sum	235.27	163.03			1349.07	800.01	59.3%	
3	1) 119.23 2) 41.36 3) 63.35	1) 8.56 2) 58.82 3) 12.67	44.66	456.55	1) 429.81 2) 187.58 3) 513.04 4) 357.76	1) 218.06 2) 166.45 3) 392.26 4) 68.39	1) 50.73% 2) 88.74% 3) 76.46% 4) 19.12%	4.7472
sum	223.94	80.05			1488.19	845.16	56.79%	
4	1) 17.27 2) 0.19 3) 45.06	1) 60.59 2) 93.75 3) 80.97	501.58	659.08	1) 515.38 2) 515.38 3) 211.54 4) 143.59	1) 220.65 2) 219.35 3) 268.39 4) 65.81	1) 42.81% 2) 42.56% 3) 126.87% 4) 45.83%	2.8867
sum	62.52	235.31			1385.89	774.2	55.86%	
5	1) 56.76 2) 30.09 3) 24.46	1) 10.77 2) 32.37 3) 92.66	785.83	947.83	1) 203.75 2) 243.75 3) 331.25 4) 193.75	1) 185.81 2) 229.68 3) 192.26 4) 20.65	1) 91.2% 2) 94.23% 3) 58.04% 4) 10.66%	3.1380
sum	111.31	135.8			972.5	628.4	64.62%	
average							60.77%	3.3192

Table 5.3: *The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 40 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 5 instances may be found in Figure B.7 in Appendix B.*

60 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
1	1) 8.05 2) 5.81 3) 24.28 4) 35.74 5) 52.23	1) 16.82 2) 4.02 3) 33.65 4) 33.02	1180.51	194.02	1) 382.55 2) 284.56 3) 438.93 4) 277.85 5) 265.77 6) 8.05	1) 139.35 2) 274.84 3) 318.71 4) 312.26 5) 72.26	1) 36.43% 2) 96.58% 3) 62.62% 4) 112.38% 5) 27.19% 6) 0%	5.0158
sum	126.11	87.51			1657.71	1073.55	64.76%	
2	1) 10.28 2) 65.44 3) 5.85 4) 78.5	1) 40.24 2) 86.98 3) 10.48 4) 70.69	0.81	167.91	1) 358.97 2) 334.62 3) 274.36 4) 332.05 5) 430.77	1) 216.35 2) 249.03 3) 229.68 4) 193.55 5) 109.68	1) 60.27% 2) 74.42% 3) 83.71% 4) 58.29% 5) 25.46%	5.3953
sum	160.07	208.39			1730.77	998.29	57.68%	

Table 5.4: *The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 60 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 2 instances may be found in Figure B.8 in Appendix B.*

Properties of the benchmark testing consists of items of various heights and fonts as well as

60 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
3	1) 63.51 2) 27.83 3) 43.53 4) 21.17 5) 36.08	1) 48.01 2) 1.92 3) 30.15 4) 35.97 5) 35.16	624.36	875.40	1) 465.25 2) 356.03 3) 417.02 4) 317.73 5) 293.62 6) 79.43	1) 234.84 2) 243.87 3) 396.13 4) 116.13 5) 268.39 6) 77.42	1) 50.48% 2) 68.5% 3) 94.99% 4) 36.55% 5) 91.41% 6) 97.47%	5.0158
sum	192.12	151.21			1929.08	1336.78	69.3%	
4	1) 45.25 2) 104.28 3) 49.68 4) 5.28 5) 134.28	1) 23.90 2) 76.31 3) 58.77 4) 73.81 5) 78.03	583.69	1.048.43	1) 361.43 2) 321.43 3) 340 4) 424.29 5) 311.43 6) 234.29	1) 98.06 2) 258.06 3) 326.45 4) 166.45 5) 108.39 6) 11.61	1) 27.13% 2) 80.28% 3) 96.01% 4) 39.23% 5) 34.8% 6) 4.96%	5.3275
sum	338.77	310.82			1992.87	969.02	48.62%	
5	1) 93.87 2) 51.3 3) 5.36 4) 32.15 5) 103.26	1) 117.94 2) 29.55 3) 57.83 4) 29.01 5) 25.19	398.67	626.56	1) 287.5 2) 297.22 3) 236.11 4) 401.39 5) 251.39 6) 397.22	1) 161.29 2) 291.61 3) 357.42 4) 215.48 5) 129.03 6) 38.71	1) 56.1% 2) 98.11% 3) 151.38% 4) 53.68% 5) 51.33% 6) 9.75%	6.3932
sum	285.94	259.52			1870.83	1193.54	63.8%	
average							60.83%	5.4295

Table 5.5: *(Table 5.4 continued.) The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 60 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 3 instances may be found in Figure B.8 in Appendix B*

80 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
1	1) 10.58 2) 83.54 3) 119.77 4) 86.05 5) 86.95 6) 83.99 7) 63.21	1) 30.85 2) 85.02 3) 68.40 4) 87.00 5) 24.85 6) 4.71 7) 54.91	536.90	1100.39	1) 478.5 2) 261.68 3) 282.24 4) 246.73 5) 192.52 6) 340.19 7) 332.71 8) 97.2	1) 168.70 2) 147.83 3) 50.43 4) 194.78 5) 255.65 6) 126.96 7) 151.30 8) 0	1) 35.26% 2) 56.49% 3) 17.87% 4) 78.94% 5) 132.79% 6) 37.32% 7) 45.48% 8) 0%	8.3770
sum	534.09	355.74			2231.77	1095.65	49.09%	
2	1) 112.46 2) 53.64 3) 95.19 4) 88.4 5) 8.84 6) 71.65 7) 6.13	1) 21.40 2) 40.29 3) 76.40 4) 97.84 5) 26.31 6) 42.87	1156.34	387.64	1) 300 2) 209.43 3) 305.66 4) 277.36 5) 479.25 6) 349.06 7) 501.89 8) 5.66	1) 170.08 2) 201.57 3) 138.58 4) 428.35 5) 462.99 6) 261.42 7) 85.04	1) 56.69% 2) 96.25% 3) 45.34% 4) 154.44% 5) 96.61% 6) 74.89% 7) 74.89% 8) 0%	6.7935
sum	436.31	305.11			2428.31	1748.03	71.99%	

Table 5.6: *The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 80 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 2 instances may be found in Figure B.9 in Appendix B.*

80 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
3	1) 10.39 2) 125.42 3) 48.98 4) 17.96 5) 45.52 6) 63.7 7) 6.09	1) 122.41 2) 49.80 3) 117.41 4) 44.71 5) 53.12 6) 14.17	841.67	102.07	1) 415.09 2) 462.26 3) 390.57 4) 226.42 5) 332.08 6) 311.32 7) 301.89 8) 56.6	1) 152.76 2) 129.13 3) 154.33 4) 363.78 5) 440.94 6) 97.64 7) 96.06	1) 36.8% 2) 27.93% 3) 39.51% 4) 160.67% 5) 132.78% 6) 31.36% 7) 31.36% 8) 0%	8.9503
sum	318.06	401.62			2496.23	1434.64	57.47%	
4	1) 4.4 2) 54.56 3) 42.77 4) 12.27 5) 66.67 6) 57.57 7) 46.61	1) 18.02 2) 117.08 3) 102.90 4) 27.30 5) 77.00 6) 36.60	1073.01	95.25	1) 356.6 2) 367.92 3) 309.43 4) 354.72 5) 371.7 6) 354.72 7) 322.64 8) 0	1) 195.28 2) 247.24 3) 299.21 4) 247.24 5) 325.98 6) 118.11 7) 137.01	1) 54.76% 2) 67.2% 3) 96.7% 4) 69.7% 5) 87.7% 6) 33.3% 7) 33.3% 8) 0%	8.3572
sum	284.85	378.9			2437.73	1570.07	64.41%	
5	1) 27.15 2) 16.56 3) 78.17 4) 78.95 5) 78.64 6) 44.84 7) 15.6	1) 64.91 2) 85.26 3) 57.27 4) 11.93 5) 46.08 6) 82.12	1073.78	255.81	1) 298.11 2) 411.32 3) 545.28 4) 571.7 5) 411.32 6) 249.06 7) 303.77 8) 0	1) 118.03 2) 240.98 3) 300.00 4) 360.66 5) 370.49 6) 222.95 7) 95.08	1) 39.59% 2) 58.59% 3) 55.02% 4) 63.09% 5) 90.07% 6) 89.52% 7) 89.52% 8) 0%	11.1492
sum	339.91	347.57			2790.56	1708.19	61.21%	
average							61.67%	8.7254

Table 5.7: *(Table 5.6 continued.) The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 80 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 3 instances may be found in Figure B.9 in Appendix B.*

100 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
1	1) 37.71 2) 96.15 3) 80.79 4) 15.35 5) 8.28 6) 75.84 7) 78.72 8) 70.66	1) 64.33 2) 35.48 3) 64.68 4) 131.13 5) 91.03 6) 60.03 7) 28.30 8) 73.25	9.48	335.75	1) 427.37 2) 273.68 3) 385.26 4) 284.21 5) 416.84 6) 355.79 7) 330.53 8) 221.05 9) 433.68	1) 164.36 2) 326.73 3) 148.51 4) 304.95 5) 164.36 6) 427.72 7) 219.80 8) 110.89 9) 102.97	1) 38.46% 2) 119.38% 3) 38.55% 4) 107.3% 5) 39.43% 6) 120.22% 7) 66.5% 8) 50.17% 9) 23.74%	15.8782
sum	463.5	548.23			3128.41	1970.29	62.98%	

Table 5.8: *The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 100 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for this instance may be found in Figure B.10 in Appendix B.*

whether or not rotations are permitted. Three space saving measurements have been defined, namely end space waste, free space as well as collective waste. Running time of the packing algorithm is also observed for each packing test. This chapter has shown that the rule-set

100 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
2	1) 29.69 2) 85.73 3) 9.45 4) 43.22 5) 49.16 6) 50.34 7) 92.35 8) 37.59	1) 55.54 2) 3.10 3) 12.12 4) 96.61 5) 100.45 6) 105.91 7) 89.03 8) 79.72	130.90	711.84	1) 309.47 2) 334.74 3) 376.84 4) 256.84 5) 309.47 6) 326.32 7) 381.05 8) 328.42 9) 315.79	1) 73.27 2) 273.27 3) 160.40 4) 142.57 5) 211.88 6) 205.94 7) 237.62 8) 100.99 9) 63.37	1) 23.68% 2) 81.64% 3) 42.56% 4) 55.51% 5) 68.47% 6) 63.11% 7) 62.36% 8) 30.75% 9) 20.07%	12.6113
sum	397.53	542.48			2938.94	1469.31	49.99%	
3	1) 91.28 2) 113.28 3) 39.51 4) 85.28 5) 68.55 6) 61.56 7) 132.5 8) 68.51 9) 98.94	1) 114.51 2) 29.53 3) 122.03 4) 59.61 5) 94.43 6) 51.63 7) 78.88 8) 39.02 9) 12.87	417.99	1021.56	1) 327.06 2) 136.47 3) 360 4) 167.06 5) 287.06 6) 223.53 7) 305.88 8) 362.35 9) 232.94 10) 169.41	1) 93.33 2) 144.44 3) 173.33 4) 148.89 5) 251.11 6) 408.89 7) 413.33 8) 200.00 9) 204.44 10) 13.33	1) 28.54% 2) 105.84% 3) 48.15% 4) 89.12% 5) 87.48% 6) 182.92% 7) 135.13% 8) 55.2% 9) 87.77% 10) 7.87%	12.6992
sum	759.41	602.51			2571.76	2051.09	79.75%	
4	1) 43.51 2) 47.03 3) 48.8 4) 34.39 5) 7.01 6) 62.92 7) 50.24 8) 9.15 9) 52.93	1) 31.60 2) 25.52 3) 32.31 4) 82.71 5) 19.81 6) 52.39 7) 77.62 8) 80.26	1100.39	481.07	1) 276.74 2) 288.37 3) 309.3 4) 590.7 5) 541.86 6) 169.77 7) 316.28 8) 376.74 9) 353.49 10) 34.88	1) 102.97 2) 229.70 3) 211.88 4) 398.02 5) 400.00 6) 334.65 7) 122.77 8) 71.29 9) 63.37	1) 37.21% 2) 79.65% 3) 68.5% 4) 67.38% 5) 73.82% 6) 197.12% 7) 38.82% 8) 18.92% 9) 17.93% 10) 0%	9.0065
sum	355.98	402.22			3258.13	1934.65	59.38%	
5	1) 32.27 2) 75.06 3) 72.93 4) 112.23 5) 70.44 6) 22.32 7) 93.03 8) 96.92 9) 45.46	1) 134.61 2) 74.70 3) 45.25 4) 43.49 5) 116.97 6) 54.92 7) 140.11 8) 51.51 9) 64.87	870.72	1104.14	1) 427.91 2) 339.53 3) 306.98 4) 418.6 5) 339.53 6) 330.23 7) 290.7 8) 232.56 9) 251.16 10) 195.35	1) 286.96 2) 78.26 3) 265.22 4) 417.39 5) 428.26 6) 334.78 7) 126.09 8) 84.78 9) 126.09 10) 4.35	1) 67.06% 2) 23.05% 3) 86.4% 4) 99.71% 5) 126.13% 6) 101.38% 7) 43.37% 8) 36.46% 9) 50.2% 10) 2.23%	16.0483
sum	620.66	726.43			3132.55	2152.18	68.7%	
average							64.16%	13.2487

Table 5.9: *(Table 5.8 continued.) The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 100 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 4 instances may be found in Figure B.10 in Appendix B.*

algorithm saves both space and time when packing items. The horizontal level nature of the rule-set algorithm leads to a guillotinable packing algorithm that is both friendly to the vinyl printer when handling orders as well as simplifying the vinyl cutters job in cutting the items from the vinyl by hand.

150 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
1	1) 61.76 2) 127.23 3) 35.59 4) 93.89 5) 116.36 6) 97.63 7) 85.78 8) 24.17 9) 91.1 10) 12.52 11) 89.91 12) 3.88 13) 74.79 14) 75.57	1) 1.30 2) 51.27 3) 8.62 4) 111.09 5) 48.51 6) 3.98 7) 14.39 8) 90.14 9) 0.58 10) 114.93 11) 30.35 12) 32.65 13) 29.24	870.67	1056.04	1) 242.96 2) 369.72 3) 359.15 4) 257.04 5) 334.51 6) 436.62 7) 295.77 8) 408.45 9) 323.94 10) 246.48 11) 373.24 12) 323.94 13) 348.59 14) 383.8 15) 52.82	1) 43.48 2) 99.38 3) 170.81 4) 211.18 5) 114.91 6) 329.19 7) 301.24 8) 201.86 9) 549.69 10) 145.96 11) 86.96 12) 195.65 13) 55.90 14) 18.63	1) 17.9% 2) 26.88% 3) 47.56% 4) 82.16% 5) 34.35% 6) 75.4% 7) 101.85% 8) 49.42% 9) 169.69% 10) 59.22% 11) 23.3% 12) 60.4% 13) 16.04% 14) 4.85% 15) 0%	13.2135
sum	990.18	537.05			4757.03	2524.84	53.08%	
2	1) 102.61 2) 82.44 3) 65.81 4) 31.92 5) 12.31 6) 25.79 7) 67.2 8) 33.97 9) 91.18 10) 22.94 11) 109.49 12) 5.91 13) 65.39 14) 45.71	1) 106.98 2) 41.46 3) 63.08 4) 90.58 5) 38.54 6) 69.51 7) 50.63 8) 51.07 9) 1.17 10) 34.16 11) 41.20 12) 5.85 13) 37.01	1137.23	907.31	1) 372.34 2) 432.62 3) 365.25 4) 340.43 5) 496.45 6) 361.7 7) 297.87 8) 294.33 9) 216.31 10) 365.25 11) 216.31 12) 290.78 13) 393.62 14) 340.43 15) 14.18	1) 74.53 2) 108.70 3) 130.43 4) 400.62 5) 251.55 6) 201.86 7) 291.93 8) 260.87 9) 329.19 10) 313.66 11) 170.81 12) 108.70 13) 118.01 14) 28.00	1) 20.02% 2) 25.13% 3) 35.71% 4) 117.68% 5) 50.67% 6) 55.81% 7) 98.01% 8) 88.63% 9) 152.18% 10) 85.88% 11) 78.97% 12) 37.38% 13) 29.98% 14) 8.22% 15) 0%	19.7842
sum	762.67	631.24			4797.87	2788.86	58.13%	
3	1) 0.41 2) 81.11 3) 17.48 4) 15.22 5) 83.82 6) 90.97 7) 122.87 8) 17.57 9) 12.46 10) 87.37 11) 91.81 12) 23.19 13) 101.33 14) 33.28	1) 36.59 2) 78.33 3) 0.88 4) 110.38 5) 20.22 6) 21.71 7) 42.44 8) 51.23 9) 6.66 10) 19.23 11) 4.18 12) 1.76 13) 50.57	1000.44	899.81	1) 407.8 2) 308.51 3) 269.5 4) 468.09 5) 117.02 6) 297.87 7) 304.96 8) 411.35 9) 290.78 10) 471.63 11) 248.23 12) 354.61 13) 237.59 14) 184.4 15) 28.37	1) 19.05 2) 212.70 3) 257.14 4) 79.37 5) 406.35 6) 285.71 7) 206.35 8) 301.59 9) 438.10 10) 377.78 11) 317.46 12) 260.32 13) 133.33 14) 19.05	1) 4.67% 2) 68.94% 3) 95.41% 4) 16.96% 5) 347.25% 6) 95.92% 7) 67.66% 8) 73.32% 9) 150.66% 10) 80.1% 11) 127.89% 12) 73.41% 13) 56.12% 14) 10.33% 15) 0%	25.3835
sum	778.89	444.18			4400.71	3314.3	75.31%	

Table 5.10: *The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 150 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 3 instances may be found in Figure B.11 in Appendix B.*

150 items	ESW(mm)		FS(mm)		CW(mm)			Running time(min)
	initial	final	initial	final	initial	percent	final	
4	1) 38.59 2) 8.4 3) 100.94 4) 45.88 5) 24.81 6) 9.78 7) 16.56 8) 39.13 9) 8.99 10) 59.34 11) 29.02 12) 2.05 13) 45.73	1) 27.12 2) 52.37 3) 41.27 4) 67.43 5) 106.51 6) 30.79 7) 46.23 8) 46.73 9) 91.63 10) 63.56 11) 46.54 12) 6.08	719.18	148.47	1) 158.68 2) 439.67 3) 340.5 4) 214.88 5) 343.8 6) 446.28 7) 400 8) 436.36 9) 224.79 10) 363.64 11) 287.6 12) 317.36 13) 310.74 14) 89.26	1) 103.45 2) 129.31 3) 189.66 4) 341.95 5) 387.93 6) 379.31 7) 344.83 8) 238.51 9) 425.29 10) 290.23 11) 109.20 12) 126.44 13) 106.32	1) 65.19% 2) 29.41% 3) 55.7% 4) 159.14% 5) 112.84% 6) 84.99% 7) 86.21% 8) 54.66% 9) 189.19% 10) 79.81% 11) 37.97% 12) 39.84% 13) 34.22% 14) 0%	19.0483
sum	429.22	626.26			4373.56	3172.43	72.54%	
5	1) 33.68 2) 14.14 3) 56.46 4) 83.73 5) 82.16 6) 27.44 7) 56.76 8) 4.39 9) 21.21 10) 65.33 11) 59.64 12) 71.07 13) 61.84	1) 42.12 2) 50.07 3) 43.18 4) 17.46 5) 37.20 6) 86.69 7) 48.66 8) 69.61 9) 100.99 10) 69.49 11) 79.91 12) 22.11 13) 21.23	25.12	788.60	1) 406.61 2) 366.94 3) 323.97 4) 370.25 5) 310.74 6) 337.19 7) 380.17 8) 277.69 9) 390.08 10) 234.71 11) 317.36 12) 347.11 13) 290.91 14) 419.83	1) 102.48 2) 152.17 3) 55.90 4) 217.39 5) 204.97 6) 195.65 7) 509.32 8) 531.06 9) 248.45 10) 574.53 11) 288.82 12) 236.02 13) 167.70 14) 18.63	1) 25.2% 2) 41.47% 3) 17.25% 4) 58.71% 5) 65.96% 6) 58.02% 7) 133.97% 8) 191.24% 9) 63.69% 10) 244.78% 11) 91.01% 12) 68% 13) 57.65% 14) 4.44%	14.3987
sum	637.85	688.72			4773.56	3503.09	73.39%	
average							66.49%	18.36564

Table 5.11: *(Table 5.10 continued.) The results of the ESW, FS, and CW for the initial packings vs the final packings, the percent of final CW over the initial CW, as well as the running time when packing 150 randomly generated item sets, that may be rotated, for font type* DinB *and text height 175mm. The final packings for these 2 instances may be found in Figure B.11 in Appendix B.*

CHAPTER 6

Conclusion

Contents

In this chapter a summary of the thesis is given in §6.1 after which some recommendations for KS, the case study on which the work done in this research project was based, are given in §6.2. Some challenges that have surfaced through the development phase of the rule-set algorithm together with ideas for future work as well as some remarks on why some of these were not included in this research project are discussed in §6.3.

6.1 Thesis summary

Chapter 1 opens with the road sign production industry including a detailed literature survey on the various types of road signs as well as the road sign production company Kohler Signs. A brief introduction to packing problems and the specific packing problem of this study, namely the problem of packing letters and numbers on a roll of vinyl, were also given.

A more detailed literature review of mostly 2D packing problems and packing heuristics are presented in Chapter 2. This includes regular packing algorithms and irregular packing algorithms, and therefore addressed the first problem objective as listed in §1.4.

In order to decide where to place items upon the roll of vinyl, the items must first be processed into the correct format for the packing algorithm. For this process some ideas from image processing, as described in Chapter 3, were used. This process entails that the original items first need to be cleaned, if required, together with representing each item in a matrix of zeroes and ones. These matrices are then reduced to fewer points by first taking the contour of the image and then interpolating these points further. Finally polygon thinning reduced the number of points even further by removing any redundant points. This concluded the second problem objective as listed in §1.4. The last part of Chapter 3 dealt with the parameter values used during the cleaning process as well as the interpolation step.

The third problem objective as listed in §1.4, namely the methodology of the packing algorithm presented in this thesis, is the topic of Chapter 4. More particularly, items are grouped into similar looking shapes and these shape groups all got an adjacency placement value which is the

rule-set value, depending on how close other shape groups can be placed next to it. The larger part of Chapter 4 is to describe the steps of the rule-set packing algorithm that pack irregular items into levels within a strip in a tight manner. The chapter concluded with the method with which the algorithm deals with rotations.

The results of the rule-set algorithm running various test cases is presented in Chapter 5. The test cases include examples from Kohler Signs as well as randomly generated test cases and the properties of these test cases are also described. A summary of results concluded the chapter and thus Objectives 4 and 5 listed in §1.4 were addressed in this chapter.

The algorithm saves more vinyl space compared to common practises as well as improving the time it takes to run such a task. The number of levels utilised is either fewer or the same as the initial solutions, both of which equate to less waste being generated when compared to KS graphic designers results. The letters are placed more tightly together when compared to initial solutions, by 63.99% of initial packing CW. Depending on the number of items to pack, the rule-set algorithm improves the running time upon common practices by 50% or more as discussed in §5.2. Running times vary on average from 0.6914min for 10 items to 18.36564min for 150 items.

6.2 Recommendations

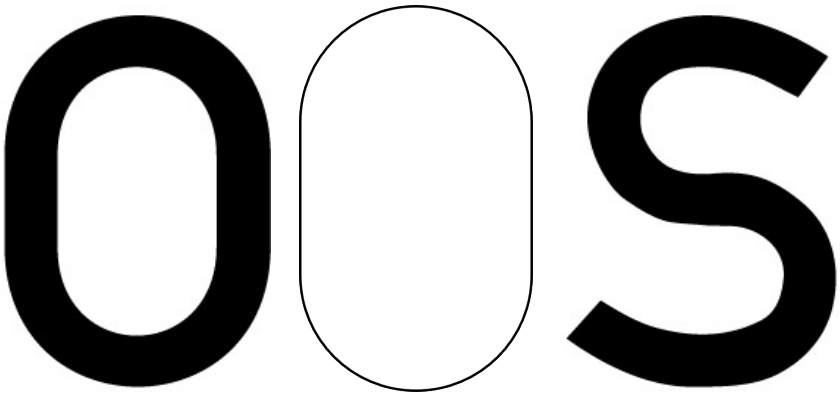
Utilising the *Rule-set Algorithm* for the placement of items upon a roll of vinyl shows some significant improvements versus the practice at KS. At KS, the items would be placed on the vinyl in any location available, which would make cutting the items out of the vinyl sheets difficult for the cutters to do without accidentally slicing into another item. Thus, items are placed upon the vinyl by the graphic designer in a well spaced out manner resulting in a border of free space occurring between each item, thereby wasting significant vinyl between each item. The rule-set algorithm places items on levels in a strip environment which allows the cutters to guillotine sections of items which results in long horizontal pieces of items. These long but easy to work with pieces of vinyl can then be cut out by the vinyl cutters. Another advantage of the fact that the packing is guillotinable, is that there is no need to waste more vinyl by spacing the items apart to reduce accidental slicing of items.

The placement of items on the vinyl for a job of approximately 100 items may take the graphic designer in industry roughly 20 minutes to pack all the items upon the vinyl. The same set of items, under the same packing conditions, may be packed by the rule-set algorithm in just under 10 minutes, which is half the time it takes the graphic designer to pack the same set of items by hand. Significant space savings are also exhibited by utilising the rule-set packing algorithm over the placement of items by hand by the graphic designer. Since items are placed upon levels, this allows the graphic designer to add additional items as he or she chooses, in order to fill up the remaining space upon that level.

Thus, the rule-set algorithm saves both vinyl as well as production time. It can also be speculated that due to the location of the items placed upon the vinyl, where items are placed in levels, that some time may also be saved in the cutting procedure by the vinyl cutters, but this cannot be said for certain until the cutting procedure is put to the test at KS.

6.3 Challenges and Future Work

Some challenges have occurred in the creation of the rule-set algorithm. One such challenge is to place items into unique shape groups in a simplified manner and giving each of these shape groups a rating when they are placed adjacent to other shape groups. Some items, like the upper-case letter "S" is placed in the same shape group as the upper-case letter "O" but are quite different. The letter "S" may possess the same qualities as the letter "O" in that they are both round at the top and bottom, except the letter "S" consists of gaps in the centre of it on either side, a property which the letter "O" does not share. This comparison can be seen visually in Figure 6.1. However, the upper-case letter "S" was placed within the round shape group since this is the most accurate shape group available for placement from the set of shape groups already predefined. The shape groups could be expanded on in the future to include a shape type that fits the item "S" more accurately, but this will increase the number of comparisons.



(a) (b) (c)

FIGURE 6.1: *Comparison of the upper-case letter "O" and upper case letter "S" items, with the round shape group they both fall in.*

An improvement to the algorithm would be to replace the shape groups with the items themselves. One could replace these shape groups with a list of all possible items that may be packed instead. This means that each item would need to be compared with each other item where item A would have to be compared with item B from both sides, resulting in two comparisons. As the packing algorithm stands at the moment (which includes rotated items as separate unique items) this strategy would consist of a 90 by 90 matrix of compared items, which is 8100 unique comparisons. This matrix library of item comparisons is rather large, but would result in a more accurate packing. Since this library would be predefined before running the rule-set algorithm, improving the algorithm in this manner would increase the computation time significantly. In a real world application, like the production of road signs at Kohler Signs, this is not an option as it delays production time.

An improvement to this packing procedure would be to expand on the alphabet, or number of items, to be packed in the future. This could potentially expand on the number of unique shape groups required as well. At the moment only numbers and letters are looked at, but in the sign industry symbols are also utilised. These symbols consist of various location types like parks and recreation, the borders of the signs themselves as well as directional arrows to name a few. Adding more symbols to the rule-set algorithm would expand the use of the algorithm but would require more time to group and compare each symbol. These symbols could also be of a different height, when compared to the two variable heights from the already established

letters and numbers, adding another level of complexity to the packing.

How item rotations are dealt with is also an important challenge faced in this packing procedure. At the moment items that may be rotated are identified initially, after which a selection of those items are rotated, due to the shape type they fall under. An even number of rotated items is wanted by the algorithm. Look at shape type "J" and "L" for example. These two shape groups form part of the top set rotation group, as defined in §4.3, and are thus counted together. If there are more than a significant number of "J" shape types to be packed then rotating half of these "J" shapes is good enough for the packing algorithm. However, if there are an even number of "J" and "L" type shapes to be rotated, then there is a fair chance that only the "J" shapes or only the "L" shapes (or a significant portion thereof), will be rotated, since the selection of which items within this group should be rotated is random. This is a quick and dirty way of allowing rotations and this process could be improved upon in the future. By improving the method of selecting which items are rotated, and the way in which these items are rotated, the packing algorithm could save more space, but possibly take much longer to run due to the relative complexity of rotating items. Another way to improve the packing is to deal with rotations in a different manner. Perhaps determining which items are best to rotate for overall packing purposes, or packing all the different combinations of rotated items and seeing which is best. Or, compare each item and its rotated counterpart with the current item during the bubble sort procedure.

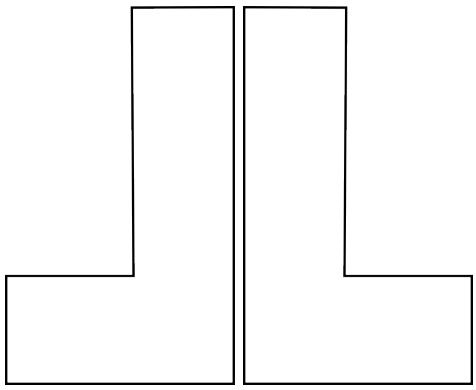


FIGURE 6.2: *Shape types "J" and "L" that form part of the rotation top set when deciding which items to rotate, but the "J" has space available at the top left and the "L" has space available at the top right.*

Another challenge is to decide where *block shaped items* should be inserted. These insertions shift the rest of the packed items down in the strip, thus adjusting the end packing result. Items that were adjacent to each other that may, for example, have been saving significant space, might become separated at the end of one level and the start of the next level, due to this newly inserted item shifting all the items down. It could also improve the packing overall, due to filling up unused space upon a level. Block items could be inserted into locations within the packing that may shift the rest of the items down the line on the same level, and shift other items onto new levels, thus resulting in new space savings that may be more efficient than before. The width of the block item and the ESW for each level may also be used to decide in which level should a particular block shape item be inserted.

An idea for future work would be to pack the items into the levels independently of the strip. That is, packing as many items as possible into each level, minimising the waste within each level independently to all other levels, as well as minimising the number of levels required for packing. One challenge with this methodology is in determining the number of levels to pack from the offset. This could perhaps be determined by calculating the number of items of each height

type and multiplying this by the average width of these items. This would determine packing areas for each item on average. Adding these areas would result in a theoretical packing area, which could be placed together to determine the total height. An estimate for the number of levels could be determined from this by dividing the average height and width. Even so, running through all of the possibilities of each item being placed at each possible location within each level could take a long time to run and would thus not be applicable in a real world industrial setting.

References

[1] Adamowicz M & Albano A, 1976, *Nesting two-dimensional shapes in rectangular modules*, Computer Aided Design, **8**, pp. 27–33.

[2] Albano A, 1977, *A method to improve two-dimensional layout*, Computer-aided Design, **9**, pp. 48–52.

[3] Arrive Alive, 2012, *Traffic signs of South Africa*, [Online], [Cited February 24th, 2012], Available from `http://www.arrivealive.co.za/pages.aspx?i=2885`.

[4] Art Jr RC, 1966, *An approach to the two dimensional irregular cutting stock problem*, (Unpublished) Technical Report 36. Y08, IBM Cambridge Scientific Center, Cambridge, Massachusetts, USA.

[5] Bossenger W, 2010, *The Three-Dimensional Bin Packing Problem*, HonsBComm, Stellenbosch University, Stellenbosch.

[6] Brink W, 2012, Applied Mathematics lecturer at *The University of Stellenbosch*, [Personal Communication], Contactable at `wbrink@exchange.sun.co.za`.

[7] Dagli CH & Tatoglu MY, 1987, *An approach to two-dimensional cutting stock problems*, International Journal of Production Research., **25**, pp. 175–190.

[8] Dyckhoff H, 1990, *A typology of cutting and packing problems*, European Journal of Operational Research, **44**, pp. 145–159.

[9] Engen, 2012, *Road Traffic Signs, Signals and Markings*, [Online], [Cited March 16th, 2012], Available from `http://www.engen.co.za/home/apps/content/products_services/driver_education/signs.aspx`.

[10] Freeman H & Shapira R, 1975, *Determining the Minimum-area Encasing Rectangle for an Arbitrary Closed Curve*, Commun. ACM, **18(7)**, July, pp. 409–413.

[11] Haims MJ & Freeman H, 1970, *A multistage solution of the template-layout problem*, IEEE Transactions on Systems Science and Cybernetics, **SSC-6**, pp. 145–151.

[12] Hifia M & MHallah R, 2003, *A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes*, International Transactions in Operational Research, **10**, pp. 195–216.

[13] Hopper E, 2000, *Two-dimensional Packing utilising Evolutionary Algorithms and other Meta-Heuristic Methods*, PhD, University of Wales, Cardiff.

[14] Kohler R, 2011, Assistant Manager/Graphic Designer at *Kohler Signs*, [Personal Communication], Contactable at `russell@kohlersigns.co.za`.

[15] Martin RR & Stephenson PC, 1988, *Putting Objects into Boxes*, Comput. Aided Des., **20(9)**, November, pp. 506–514.

[16] Ntene N, 2007, *An algorithmic approach to the 2D oriented strip packing problem*, DPhil Thesis, University of Stellenbosch, Stellenbosch.

[17] Terashima-Marin H, Ross P, Farias-Zarate C, Lopez-Camacho E & Valenzuela-Rendon M, 2010, *Generalized hyper-heuristics for solving 2D Regular and Irregular Packing Problems*, Annals of Operations Research, **179**, pp. 369–392.

[18] Traffic Signs, 2005, *Traffic Signs*, [Online], [Cited March 16th, 2012], Available from `http://www.trafficsigns.co.za/signs.php`.

[19] Wikipedia, 2013, *Bubble Sort*, [Online], [Cited September 17th, 2013], Available from `http://en.wikipedia.org/wiki/Bubble_sort`.

[20] Wikipedia, 2012, *Contour*, [Online], [Cited September 12th, 2012], Available from `http://en.wikipedia.org/wiki/Contour_line`.

[21] Wikipedia, 2013, *Edge Detection*, [Online], [Cited November 12th, 2013], Available from `http://en.wikipedia.org/wiki/Edge_detection`.

[22] Wikipedia, 2012, *Interpolation*, [Online], [Cited September 12th, 2012], Available from `http://en.wikipedia.org/wiki/Interpolation`.

[23] Wikipedia, 2012, *Matlab*, [Online], [Cited October 9th, 2012], Available from `http://en.wikipedia.org/wiki/MATLAB`.

APPENDIX A

Shape groups and rule-set values

This appendix contains the values for the shape groups for each item that is packed for the rule-set algorithm introduced in Chapter 4 as well as a description of the bubble sort algorithm which is utilising for sorting the shape groups with these rule-set values described herein.

The bubble sort algorithm is a sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the incorrect place. This process of passing through the list is repeated until no swaps are required, which indicates that the list has thus been sorted.

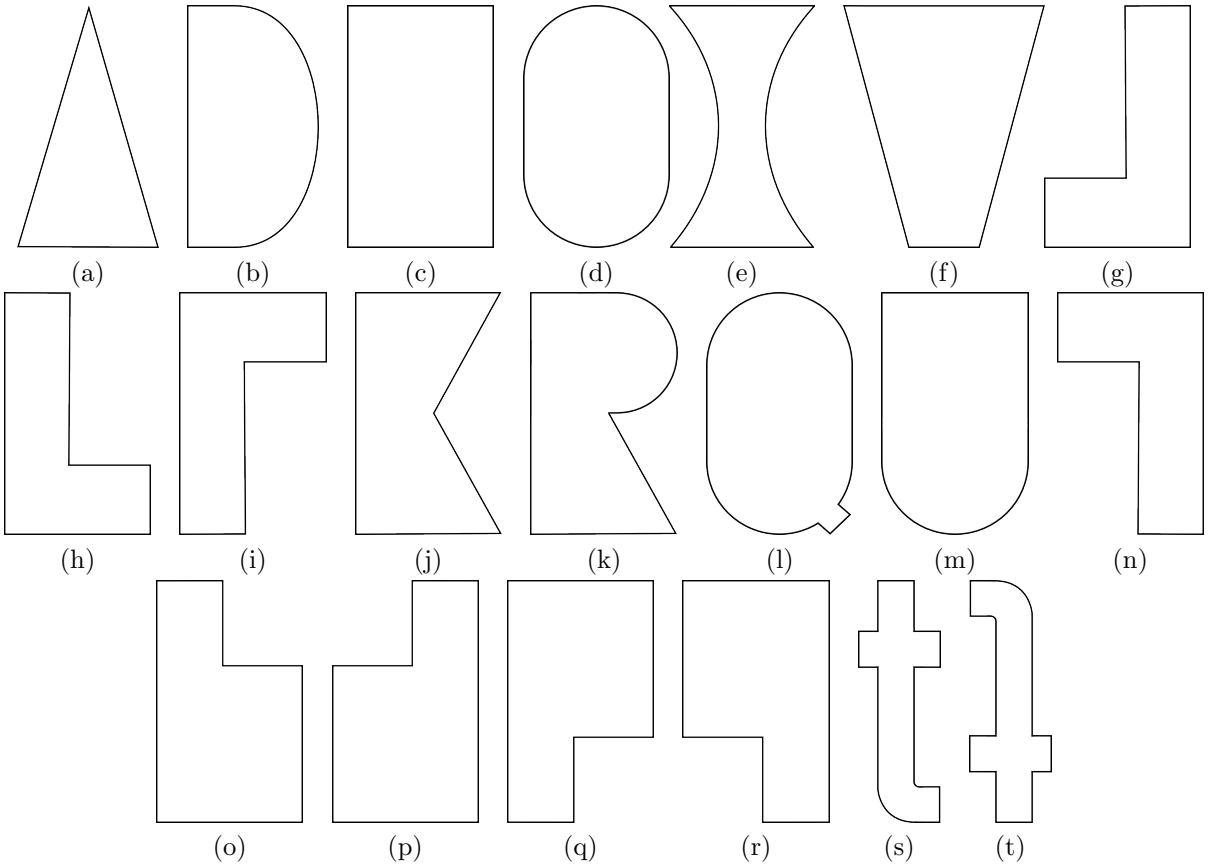


FIGURE A.1: *The shape groups for all upper-case letters, lower-case letters as well as the numbers utilised in the rule-set technique.*

Shape Group	Item(s):												
a	4	6	7′	9′	A	T′	V′	W′	Y′	v′	w′	y′	
b	B	D											
c	E	H	I	M	N	Z	i	m	n	u	z		
d	3	8	0	C	G	O	S	a	c	e	g	o	s
e	2	X	x										
f	4′	6′	7	9	A′	T	V	W	Y	v	w	y	
g	J	f′	j	r′									
h	1′	L	l										
i	J′	f	j′	r									
j	K												
k	R												
l	Q												
m	5	U											
n	1	L′	l′										
o	b	h	k	q′									
p	F′	P′	d	p′									
q	F	P	d′	p									
r	b′	h′	k′	q									
s	t												
t	t′												

Table A.1: *All the upper-case letters, lower-case letters as well as the numbers that fall within the shape groups from Figure A.1, where an item with ′ indicates the 180 degree rotated version of that item falls within that shape group.*

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
a	-2	-1	-1	1	-2	2	-2	-1	-1	-1	-1	1	1	2	-1	-2	-1	1	1	1
b	1	-1	-1	-2	2	1	1	-1	-1	-1	-1	-2	-2	1	-1	-2	-1	-2	-1	-2
c	-1	2	2	-1	-1	-1	-1	2	2	2	2	-1	-1	-1	2	-1	2	-1	-1	-1
d	1	-1	-1	-2	2	1	1	-1	-1	-1	-1	-2	-2	-2	-1	-2	-1	-2	-2	-2
e	-2	-1	-1	2	-2	-2	-2	-1	-1	-1	-1	2	-2	-2	-1	-2	-1	-2	1	-2
f	2	-1	-1	1	-2	-2	2	-1	-1	-1	-1	1	-2	-2	-1	2	-1	-2	1	-2
g	-1	2	2	-2	-2	-2	-2	2	2	2	2	-2	-2	-1	1	-2	1	-2	-1	-1
h	-2	-1	-1	1	-2	2	-2	-1	-1	-1	-1	1	1	2	-1	-2	-1	2	2	2
i	2	-1	-1	1	-2	-2	2	-1	-1	-1	-1	1	-1	-2	-1	2	-1	-2	2	-2
j	-2	-1	-1	1	-2	-2	-2	-1	-1	-1	-1	2	-2	-2	-1	-1	-1	-1	1	-2
k	-2	-1	-1	1	-2	1	-2	-1	-1	-1	-1	1	1	-2	-1	-1	-1	-1	-2	-1
l	-2	-1	-1	-2	1	2	-2	-1	-1	-1	-1	1	1	-2	-1	-1	-1	-1	-2	1
m	-1	-1	-1	-2	-2	-2	1	-1	-1	-1	-1	1	-2	-1	-1	-1	-1	-1	-2	-2
n	-1	2	2	-1	-1	-1	-1	2	2	2	2	-1	-1	-1	2	-1	2	-1	-1	-1
o	-2	-1	-1	-2	-2	2	-2	-1	-1	-1	-1	-2	-2	2	-1	-2	-1	-2	1	-2
p	-1	2	2	-1	-1	-1	-1	2	2	2	2	-1	-1	-1	2	-1	2	-1	-1	-2
q	1	-1	-1	-2	-2	-2	2	-1	-1	-1	-1	-2	-2	-2	-1	-2	-1	-2	-2	-2
r	-1	2	2	-1	-1	-1	-1	2	2	2	2	-1	-1	-1	2	-1	2	-1	-1	-1
s	-2	-1	-1	-2	-2	1	-2	-1	-1	-1	-1	-2	-2	2	-1	-2	-1	-2	-2	2
t	1	-1	-1	-2	1	1	2	-1	-1	-1	-1	-2	-2	2	-1	-2	-1	-2	2	-2

Table A.2: *The Rule-set values for each shape group from Figure A.1. Each row indicates that particular shape group is on the left hand side and each column indicates that particular shape group is on the right hand side. A negative value indicates the shape group items selected in this order is an unfavourable packing, whereas a positive value indicates that the shape group items selected in this order is a favourable packing.*

APPENDIX B

Graphs to accompany the results

This appendix contains all the supplementary tables and figures for reference from Chapter 5. In the first four figures, Figures B.1 - B.4 the packing of the item set “v”, “8”, “v”, “f”, “7”, “I”, “Z”, “x”, “y”, “0”, “y”, “x”, “W”, “p”, “S”, “v”, “o”, “x”, “g”, “3”, “r”, “w”, “l”, “g”, “A”, “3”, “I”, “4”, “7”, “q”, “j”, “L”, “x”, “3”, “T”, “P”, “m”, “p”, “B”, “W”, “I”, “g”, “A”, “8”, “x”, “M”, “c”, “E”, “G” and “X”, for each of the 8 text heights, are illustrated. Table B.1 indicates the final vinyl lengths for each of these packings. Table B.1 indicates the final vinyl

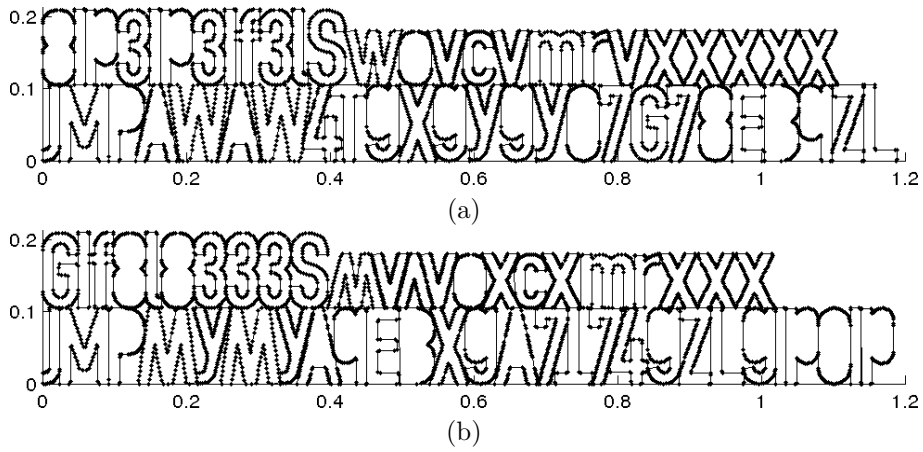


FIGURE B.1: *The item set where rotations are both prohibited (a) and allowed (b) packed for 105mm text height.*

Text height	Figure	No Rotation	Running time	Rotation	Running time
105mm	A.1 (a) and (b)	210mm	240.30s	210mm	208.27s
112mm	A.2 (a) and (b)	303mm	239.61s	224mm	216.19s
140mm	A.2 (c) and (d)	378mm	239.89s	378mm	220.09s
175mm	A.3 (a) and (b)	646mm	244.24s	646mm	215.39s
210mm	A.3 (c) and (d)	777mm	248.44s	777mm	233.18s
280mm	A.3 (e) and (f)	1512mm	251.34s	1512mm	233.42s
350mm	A.4 (a) and (b)	2242mm	262.04s	2242mm	233.87s
420mm	A.4 (c) and (d)	3411mm	263.73s	3411mm	238.82s

Table B.1: *The final vinyl length used for the packings illustrated in Figures B.1 - B.4 where rotations are prohibited and allowed for varying text heights, but for the same set of items.*

length for packing the same set of items, that vary in text height as well as whether rotations are prohibited or allowed for that packing, and within which figure that corresponding packing solution is displayed.

FIGURE B.2: *The same item set as in Figure B.1 where rotations are both prohibited ((a) and (c)) and allowed ((b) and (d)), packed for 112mm ((a) and (b)), and 140mm ((c) and (d)) text heights respectively.*

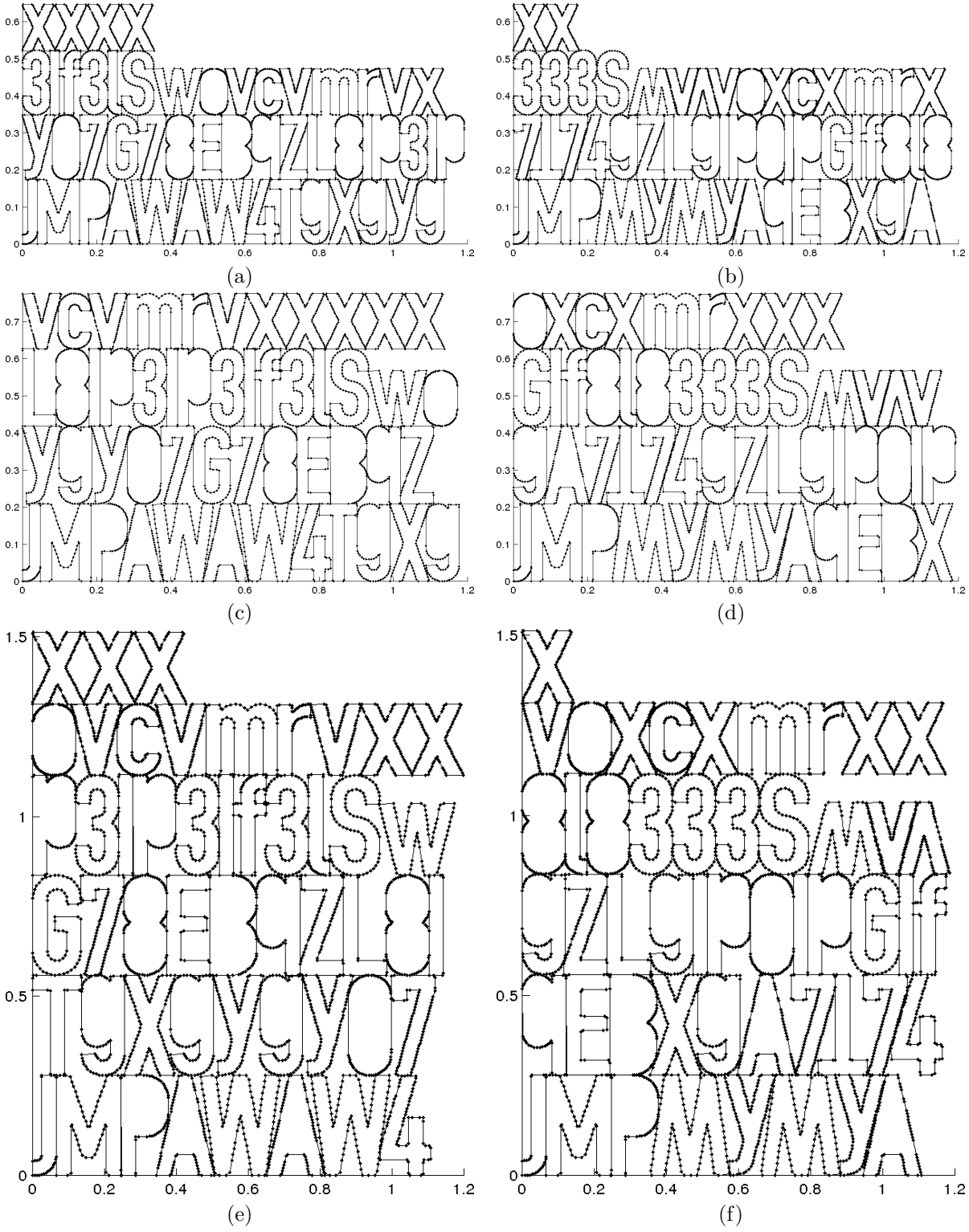


FIGURE B.3: *The same item set as in Figure B.1 where rotations are both prohibited ((a), (c) and (e)) and allowed ((b), (d) and (f)), packed for 175mm ((a) and (b)), 210mm ((c) and (d)), and 280mm ((e) and (f)) text heights respectively.*

FIGURE B.4: *The same item set as in Figure B.1 where rotations are both prohibited ((a) and (c)) and allowed ((b) and (d)), packed for 350mm ((a) and (b)) and 420mm ((c) and (d)) text heights respectively.*

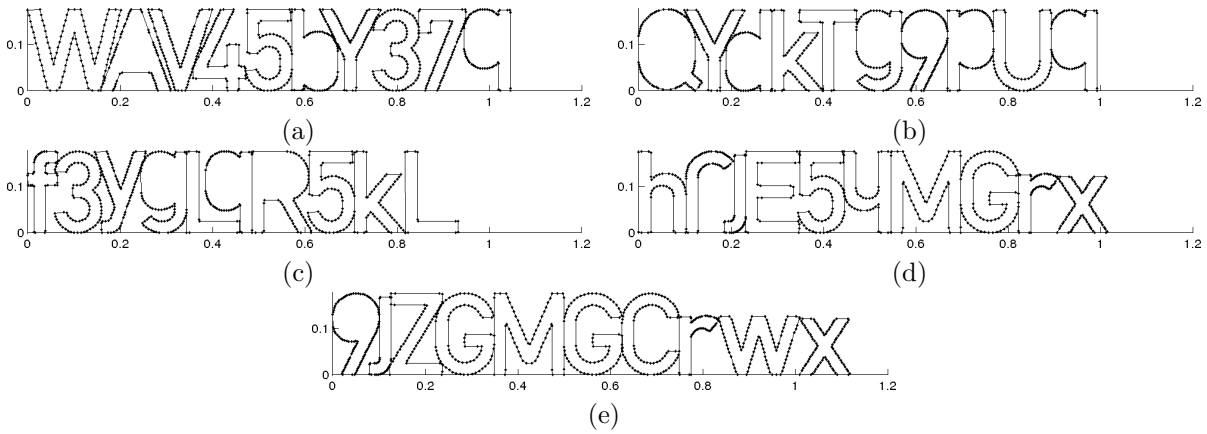


FIGURE B.5: *Final packing of 10 randomly generated items. The space savings for these 5 10-item instances are summarised in Table 5.1 in Chapter 5.*

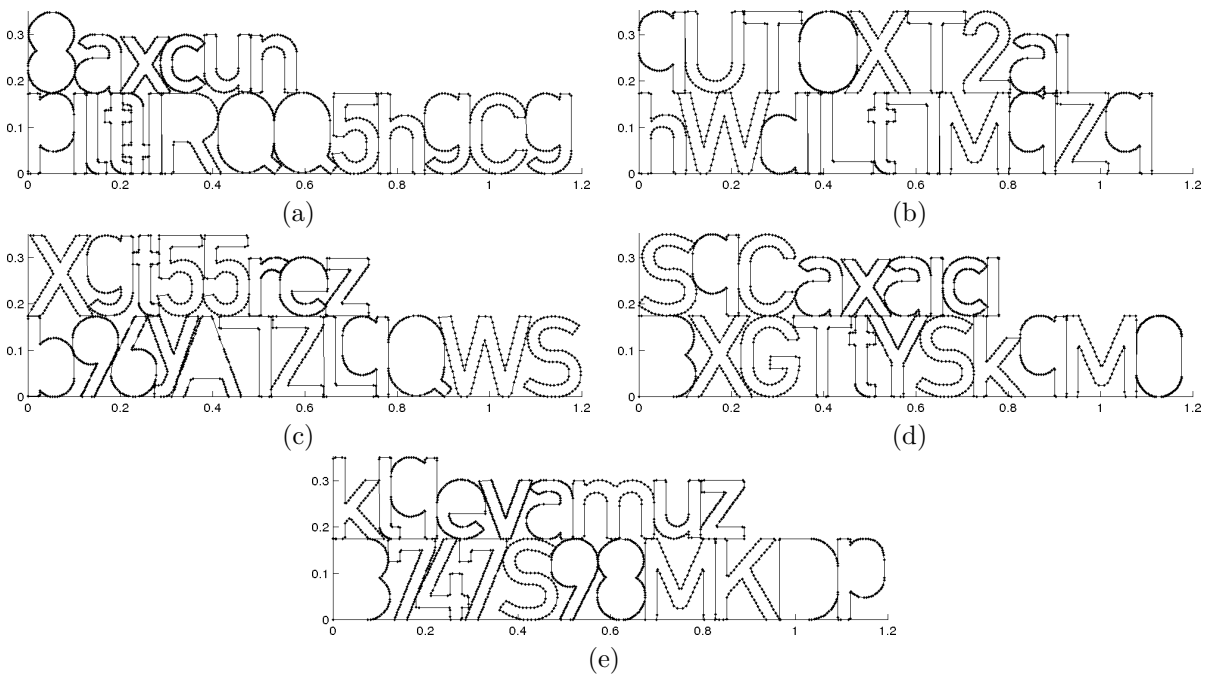


FIGURE B.6: *Final packing of 20 randomly generated items. The space savings for these 5 20-item instances are summarised in Table 5.2 in Chapter 5.*

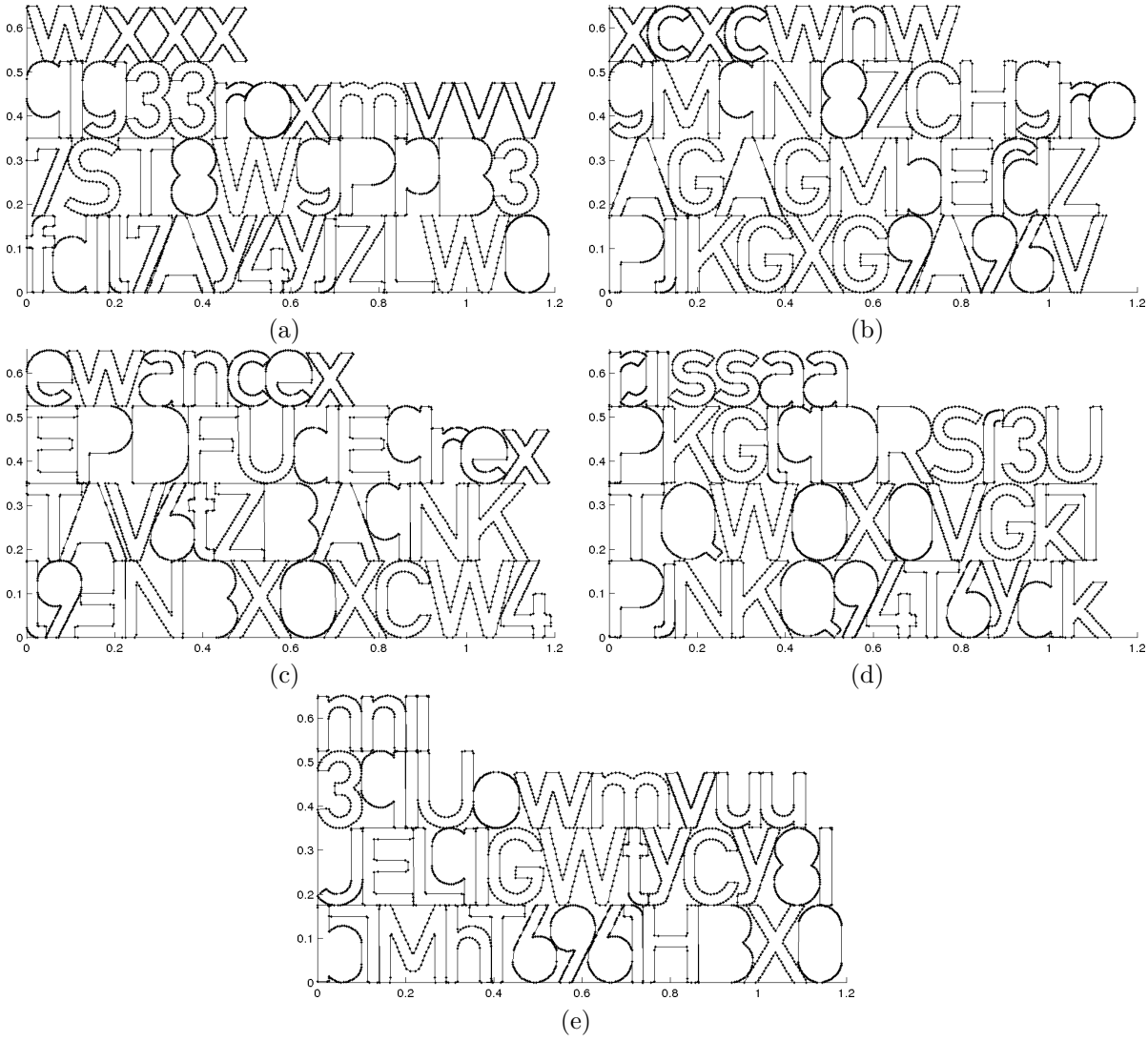
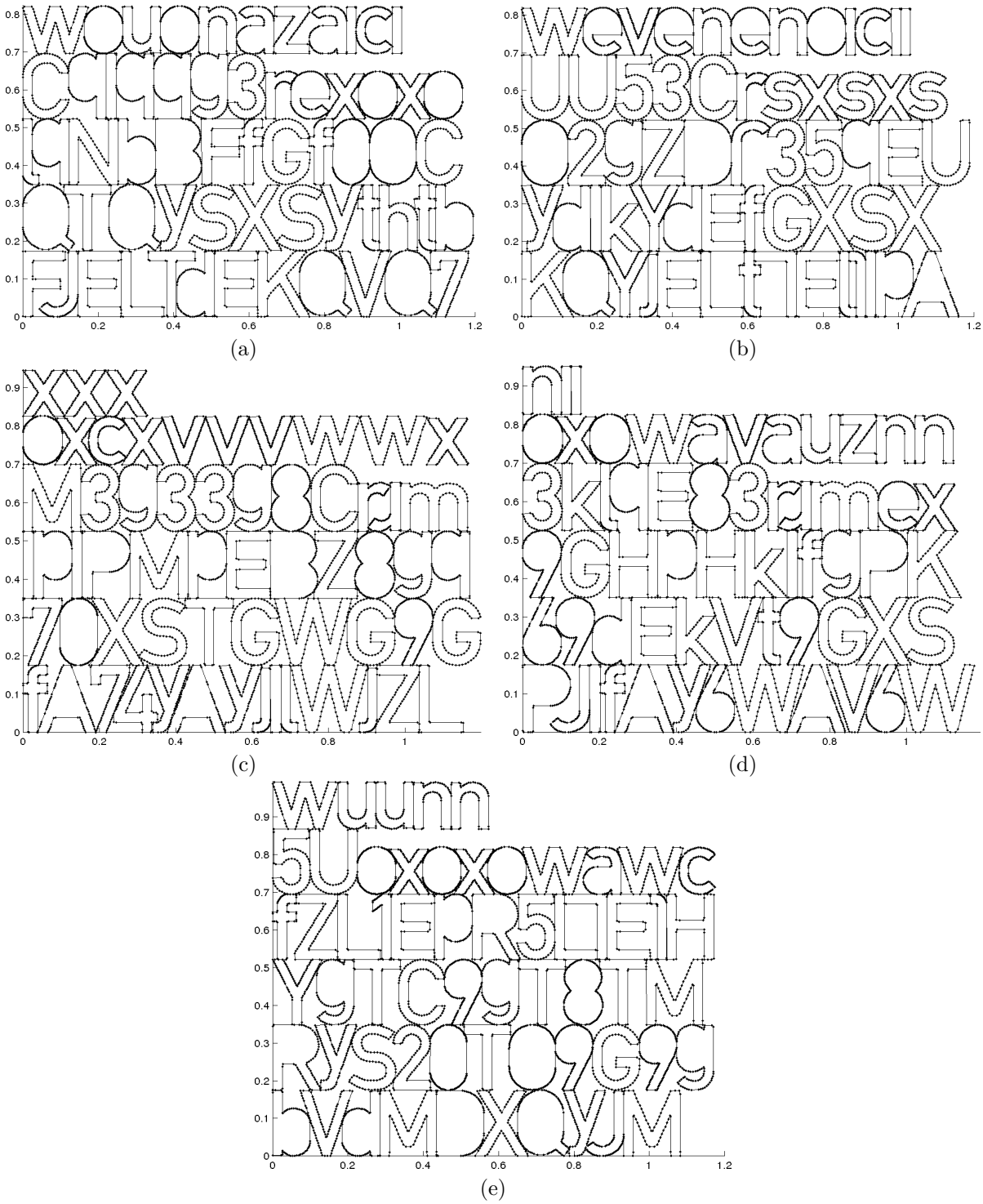


FIGURE B.7: *Final packing of 40 randomly generated items. The space savings for these 5 40-item instances are summarised in Table 5.3 in Chapter 5.*



(a) (b) (c) (d) (e)

FIGURE B.8: *Final packing of 60 randomly generated items. The space savings for these 5 60-item instances are summarised in Tables 5.4 and 5.5 in Chapter 5.*

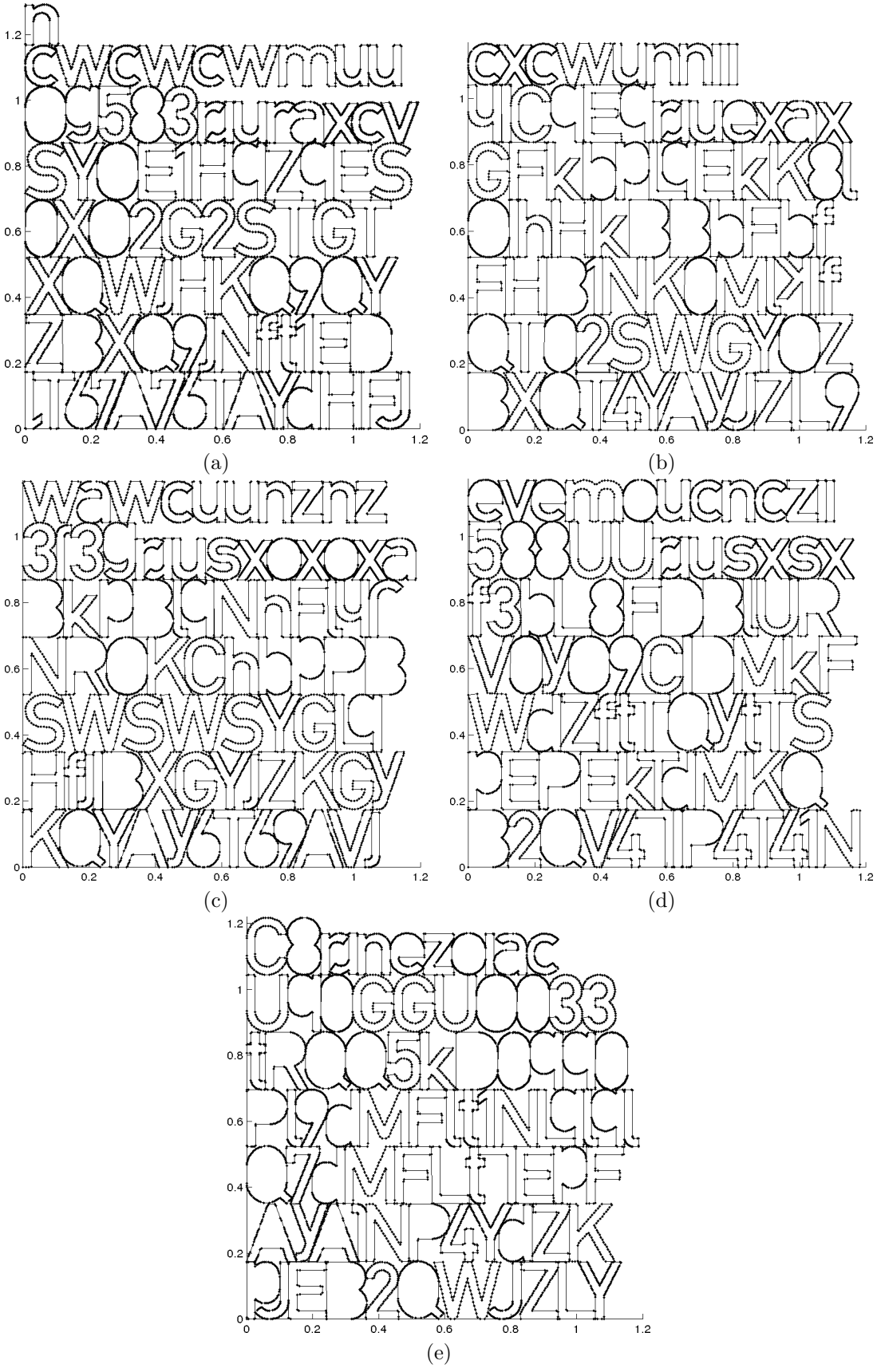


FIGURE B.9: *Final packing of 80 randomly generated items. The space savings for these 5 80-item instances are summarised in Tables 5.6 and 5.7 in Chapter 5.*

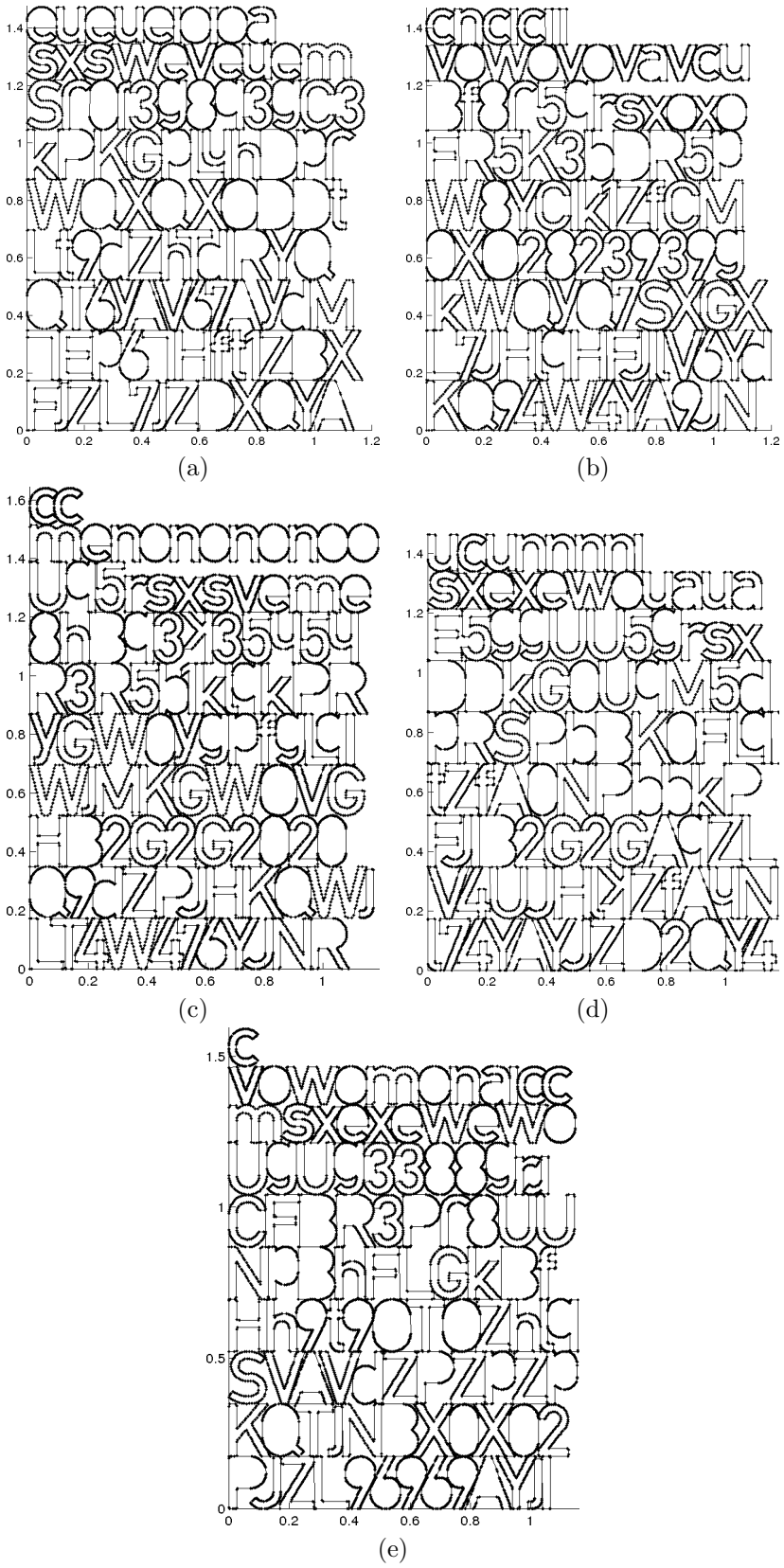


FIGURE B.10: *Final packing of 100 randomly generated items. The space savings for these 5 100-item instances are summarised in Tables 5.8 and 5.9 in Chapter 5.*

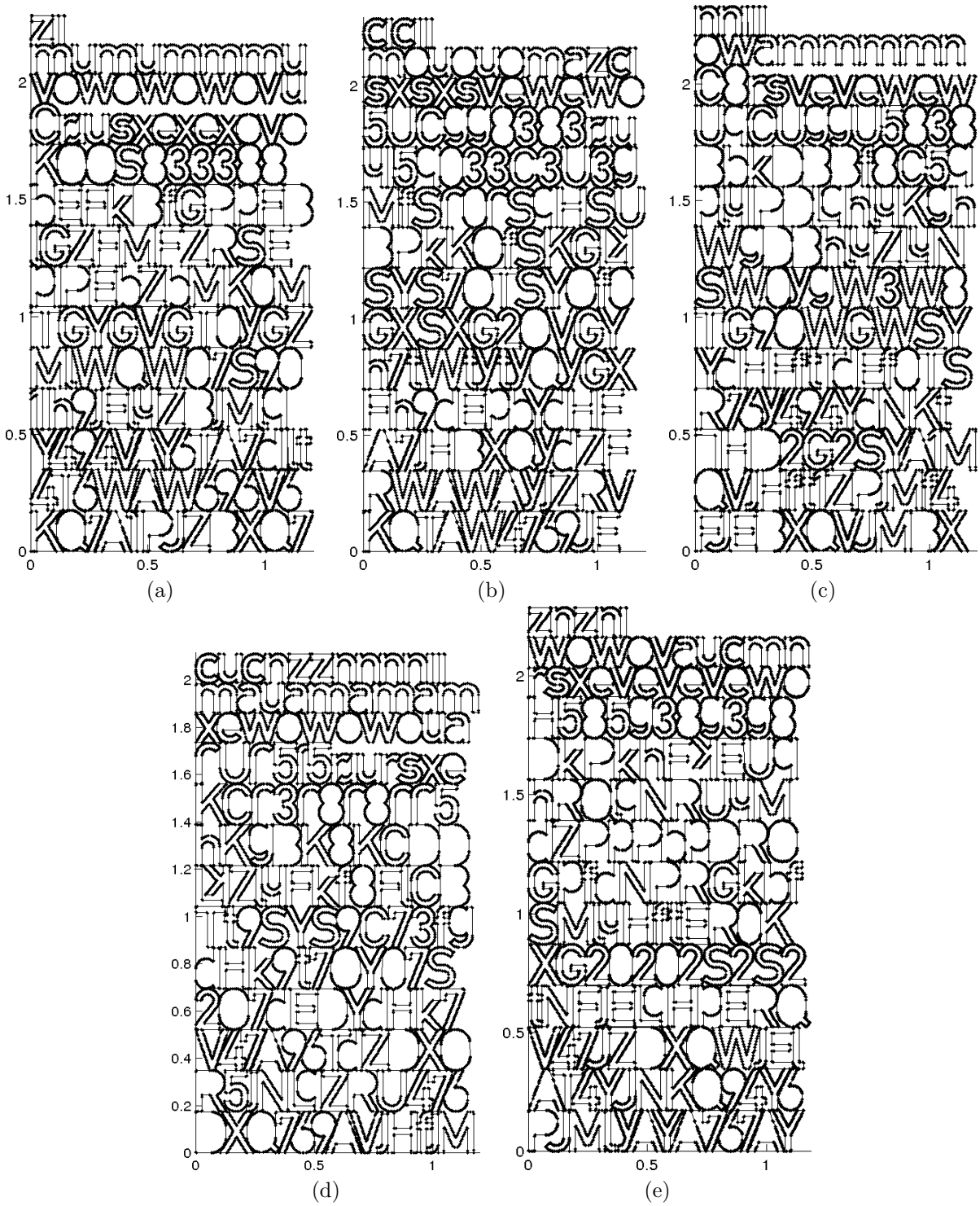


FIGURE B.11: *Final packing of 150 randomly generated items. The space savings for these 5 150-item instances are summarised in Tables 5.10 and 5.11 in Chapter 5.*