GENERATING OPTIMIZED CUTTING LAYOUTS OF DRYWALL PANELS USING BUILDING INFORMATION MODELING

A Thesis

by

BARDIA JAHANGIRI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Julian H. Kang
Committee Members,	Zofia K. Rybkowski
	Wei Yan
Head of Department,	Joe P. Horlen

May 2016

Major Subject: Construction Management

ABSTRACT

The construction industry generates a substantial amount of solid waste. Cutting building materials into smaller pieces to fit the design is one of the sources of waste in new construction. This type of waste is known as leftover or residual. Drywall leftover is an example of such waste. Currently, contractors do not perform a detailed analysis of how many drywall panels would be required. Moreover, they do not use a consistent system for reusing their scrap and often cut a needed piece from a brand new panel instead of using available scrap.

Building Information Model (BIM), as an object-oriented representation of the building contains all the required data and can be utilized to provide drywall crews with layouts indicating how to cut the panels into the required pieces so the leftover could be reduced. Also, some commercially available software applications, such as Autodesk Revit provide a platform to automate processes such as optimization by implementing algorithms through their Application Programming Interface (API).

Similar problems have been studied in other fields and industries. Bin packing problem in mathematics and Nesting process in the cutting industry are examples of such research. As the result, automated optimization methods that utilize Evolutionary Algorithms (EA) are introduced to address these problems. There is an opportunity to apply Evolutionary Algorithms to solve a similar problem in the construction industry. This study investigates if it is feasible to implement EA-based optimization methods on a BIM platform to develop an automated optimization tool.

In light of available tools and methods, an automated optimization tool is developed as a Revit add-in. It extracts geometrical data from BIM and receives dimensions of available drywall panel(s) from the user. The algorithm, finds the most desirable arrangement of panels and the number of full panels is calculated. The outline of smaller pieces that need to be cut out of full panels are also determined. Then by utilizing an EA-based optimization method, it generates the cutting layouts.

The add-in is tested on a certain number of simple models for several iterations and the generated cutting layouts show very optimal leftover. On a very specific model containing twenty pieces that need to be cut out of full panels, the add-in application spent 100 minutes to generate the cutting layouts, which resulted in 36% reduction in the leftover, compared to the layouts generated in the initial iterations. The test proved that the proposed algorithm is able to optimize cutting layouts. It demonstrates that utilizing such optimization algorithm on a BIM platform could be considered as an effective way to reduce the material waste.

To my late father Farhad Jahangiri.

You never said I'm leaving,

You never said goodbye.

You were gone before I knew it,

And only God knew why.

ACKNOWLEDGEMENTS

I would first like to thank my committee chair, Dr. Kang who always, by his inquiries and comments, made me delve into new subjects. I would also like to thank my committee members, Dr. Rybkowski and Dr. Yan for their guidance and support throughout the course of my graduate studies as well as this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

NOMENCLATURE

BIM	Building Information Modeling
EA	Evolutionary Algorithm
DE	Differential Evolution
API	Application Programming Interface
C&P	Cutting and Packing
1D	1-Dimensional
2D	2-Dimensional
3D	3-Dimensional

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

There has been many efforts and innovations to address chronic problems of construction such as low productivity and insufficient quality (Koskela, 1997) in the past two decades. In the ongoing paradigm shift (Tommelein, 2015) which is a fundamental shift of the construction industry from conventional state to a more efficient state, two major developments are very influential.

First is a conceptual approach to construction management that originally comes from a production philosophy in manufacturing. Lean construction encompasses the idea of identifying and eliminating all kinds of waste (Womack, 1999). It is about removing any waste without reducing customer value. According to Koskela (1992), waste includes any unnecessary task as well as the use of materials in larger quantities than necessary – i.e. material loss. Therefore, the focus in lean construction is on reduction in waste and increase in value to the customer.

On the other hand, Building Information Modeling, as a transformative information technology, is affecting the development of construction industry. BIM simulates the construction project in a virtual environment (Li, et al., 2008; Azhar, 2011) containing an object-oriented model of the building as well as all the data related to the building elements (Sabahi, 2010). In a building information model, elements are expressed as objects that exhibit form, function, and behavior. Also, some commercially available BIM applications, such as Revit provide a platform to automate processes such as

optimization by implementing algorithms through their Application Programming Interface (API).

Lean construction and BIM are not dependent on one other and either of those can be adopted without the other. However, there seems to be synergies between them. BIM is expected to provide the foundation for some of the results that lean construction is expected to deliver (Sacks, 2010). Especially in the case of eliminating material waste in construction, considering the embedded data and automation platform, BIM has features and capabilities that would be intrinsically instrumental.

Currently, the waste generated within the construction industry accounts for a substantial portion of solid waste stream in the United States. Studies have shown that about 10% of the waste generated in new construction is the result of cutting the building materials into pieces during the construction process (Poon, 2007). This type of waste is also known as leftover or residual and includes leftover material scraps from cutting stock material into smaller pieces to fit the design (Gavilan & Bernold, 1994).

While drywall is a major component of construction waste, in almost all construction projects there is no detailed analysis of how many drywall panels would be required. Furthermore, drywall crews working on the job site do not use any cutting layout or a consistent system for reusing their scrap and often cut a needed piece of sheetrock from a brand new panel instead of using available scrap.

However, all required pieces can be identified in advance using the embedded data in building information model and layouts to cut those pieces out of drywall panels can be

generated. If those generated layouts are already optimized, drywall leftover could be reduced to accomplish the main objective of lean construction.

1.1. Motivation

Drywall is a major component of material waste in new construction and the process of cutting drywall panels into smaller pieces is one of the main reasons for drywall waste. Similar problems have been studied in other fields and industries. Bin packing problem in Mathematics and Nesting process in the cutting industry are examples of such research. As the result, optimization methods that mostly utilize Evolutionary Algorithms (EA) are introduced to address those problems.

Building information modeling is providing us with all the required data regarding building elements including their form, function and material. Also, some commercially available software applications, such as Revit provide a platform to automate processes such as optimization by implementing algorithms through their Application Programming Interface (API).

Methods and algorithms proposed to address similar problem in other industries could be adopted and customized to develop an automated optimization tool for the same purpose in the construction industry.

1.2. Research Question

There is an opportunity to reduce construction waste through generating optimized layouts to cut drywall panels. Similar problem has been already studied and addressed in other industries by utilizing evolutionary algorithms. One may speculate if the proposed optimization methods coupled with evolutionary algorithms can be implemented on a Building Information Modeling platform, so that the drywall panel cutting layouts can be optimized automatically

1.3. Research Objectives

This objectives of study include: a) developing an add-in drywall cutting optimization application on a BIM platform using EA-based optimization techniques, and b) proving if this application can actually suggest a drywall cutting plan that generates minimum waste.

1.4. Delimitations and Assumptions

The following are the assumptions and delimitations that were considered throughout this study:

a) The add-in application was developed in the Autodesk Revit platform.

b) A Revit model containing 4 walls was used for the test. The application can handle irregular shapes but the test included only rectangular shapes.

c) The only factor studied was material wastage in the construction industry and cultural trends as well as other constraints – such as framing – were not considered.

2. REVIEW OF LITERATURE

2.1. Construction Waste and Its Composition

Construction and Demolition waste is generated when new structures including buildings and projects such as streets, highways and bridges are built and when such structures are renovated or demolished (EPA, 2003). However, building-related C&D waste only refers to the waste generated in construction, renovation and demolition of residential and nonresidential buildings. Building-related C&D waste generated in the United States was estimated to be 170 million tons in year 2003. Table 1 shows the amount of waste classified by different activities and building types.

	Residential	**Nonresidential**	**Total**	
	Million tons	Million tons	Million tons	Percent
Construction	10	5	15	9%
Renovation	38	33	71	42%
Demolition	19	65	84	49%
Total	67	103	**170 Million tons**	
Percent	39%	61%		

Table 1: Estimated building-related C&D waste generated in the U.S. (EPA, 2003)

There is not much information available on C&D generation in the United States and the existing information is limited to case studies conducted at specific points in time (McKeever, 2004). Furthermore, waste amounts are rarely described in terms of volume because the volume can change due to compaction or other processing. The amounts are generally compared in terms of weight because it generally remains constant (EPA, 2003).

Although the amount of waste generated in new construction activities is smaller than renovation and demolition activities, it seems to be more important to study and reduce because construction waste contains a big amount of chemical waste relative to demolition waste. In addition, the cost reduction caused by reducing the construction waste has direct and tangible benefit for contractors and owners (Bossink & Brouwers, 1996).

The waste generated in construction of residential buildings is twice the amount of waste for nonresidential construction (Figure 1). Moreover, it is more practical to find ways to reduce the residential construction waste due to the homogeneity of residential buildings. Therefore, this study will focus on construction of residential buildings.

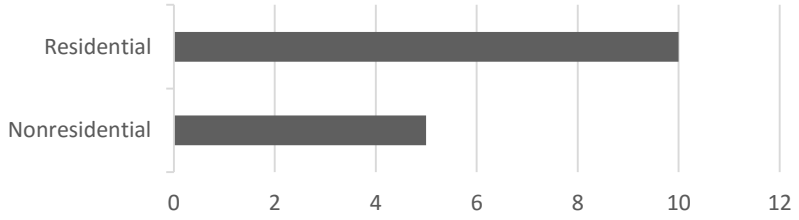


Figure 1: Construction waste in residential and nonresidential sector (million tons)

There has been much research interest on construction waste recycling but construction waste minimization has received less attention. It should be taken into consideration that the single most effective way of dealing with any solid waste is not to create it in the first place (Gavilan & Bernold, 1994). Source reduction, which is avoiding the generation of waste, saves not only money but also landfill space.

Several studies show that two major components of residential construction waste are wood and drywall (Castelo Branco, 2007; EPA, 2003; Sandler, 2003). McKeever (2004) studied the wood waste in the United States for 2002 and estimated the C&D wood waste at 35.7 million tons. He also estimated the generated wood waste in new residential to be 3.7 million tons, which is 10% of C&D wood waste. Depending on the economic state of the country, there are fluctuations in the number of construction projects and consequently in the amount of waste generated in each sector. For example, C&D wood waste was 36.4 Million tons in 2010 (Falk & McKeever, 2012).

Sandler (2003) studied the waste of drywall and estimated the drywall waste in residential new construction to be 1.4 Million tons per year. This means, in the new residential category, the generated waste of drywall is roughly 38% of wood waste. It should be considered that wood waste consists of the waste generated in several types of activities including framing, flooring, siding, paneling, roofing, cabinetry, decking, etc. Dimensional lumber is the major component of wood waste in residential new construction and studies show that in new residential construction the waste of drywall is more than the waste of lumber in terms of weight. Composition of building C&D debris provided by EPA (1998; 1995) shows the waste of drywall has a larger amount than of dimensional lumber. Also in typical construction waste composition provided by NAHB (1995) estimated weight of drywall waste and solid sawn wood (including lumber) are 2,000 pounds and 1,600 pounds respectively. Therefore, the scope of this research will be limited to waste of drywall in residential new construction.

2.2. Cutting and Packing Problem

The cutting and packing problem is the problem of finding an arrangement of pieces (items) to cut from or pack inside larger objects (bins). The conditions are (a) all items must lie entirely within a bin and (b) the items must not be overlapping (Wong & Lee, 2009).

The cutting stock problem is a branch of cutting and packing problem and aims to find the optimal way to cut required pieces from stock material in the way that the total leftover would be minimum. This problem is a combinatorial optimization problem and has been applied to many fields including steel, textile, paper, wood, metal and glass industry (Cheng, et al., 1994). Generally, in any industry that deals with cutting specific pieces out of raw material, this problem can be tracked because in such industries, in order to minimize the material waste, a good cutting layout is always beneficial.

In cutting and packing problem, hence in cutting stock problem, both items and bins can be defined in one, two, three or even larger number of dimensions. In one-dimensional problem, the width or section of the piece to cut out is equal to the width or section of the stock, so the problem deals with determining the lengths (Timmerman, 2013). Cutting cripple and sill studs out of raw lumber is such problem.

In two-dimensional setting, the stock has a fixed width. However, the length of stock can be either constrained or infinite. If the stock length is infinite, the problem is also known as 2D strip packing problem and if the stock length is constrained, the problem is called 2D bin packing problem (Timmerman, 2013). Cutting required pieces out of a

huge roll of fabric in textile industry is a 2D strip packing problem and cutting drywall pieces out of drywall panels is a 2D bin packing problem (Figure 2).

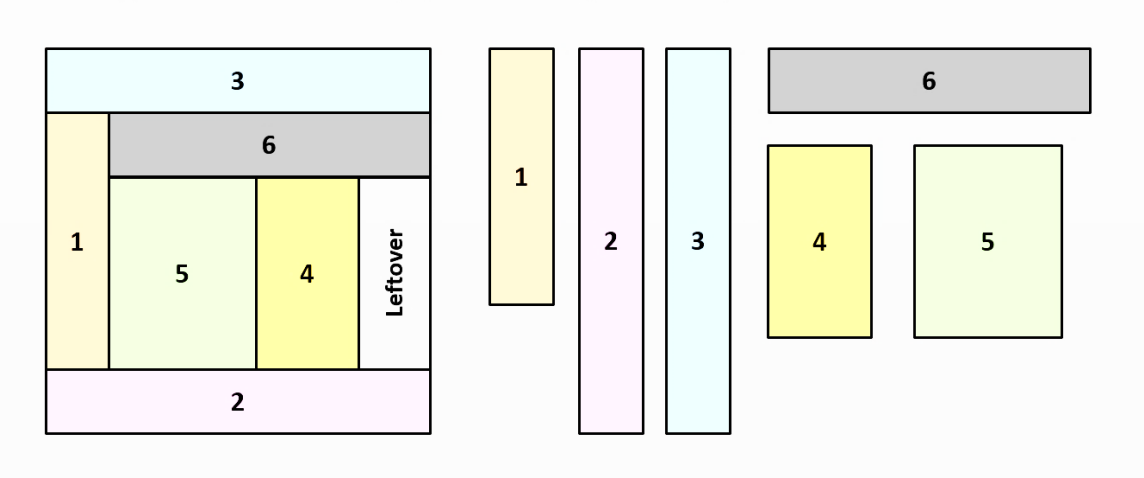


Figure 2: Visualization of 2D bin packing problem

2.2.1. Variations of 2D Bin Packing

2D bin packing problem has many variations based on the following factors:

Regular items vs. Irregular items: Items that need to be cut or packed can be regular or irregular. When the 2D objects to be cut are irregular, the above-mentioned problem is also known as the Nesting problem. Nesting strategies have been used in other industries for several years. In shipbuilding industry, in order to cut specified number of each of certain types of two dimensional shapes out of raw material, a set of cutting layouts need to be generated in such a way as to minimize the amount of leftover. Implementing nesting strategies in an automated way using computers and manufacturing machinery is a branch of Computer Aided Manufacturing (CAM) field.

Guillotine vs. non-guillotine: The guillotine cutting method refers to the procedure in which a planar (2D) panel is cut in such a way as to obtain two pieces of material. In

other words, only orthogonal cuts that bisect one component of the sheet are allowed. Guillotine method is more effective from a time standpoint. However, non-guillotine approach results in less leftover. Also, guillotine cutting is preferred among trade personnel.

Oriented vs non-oriented: Whether the items are allowed to rotate or not is the source of variations in 2D bin packing. In most cases, the items are allowed to rotate based on a certain degree such as {0, 90, 180, 270}.

2.2.2. Packing Algorithms

Zhang et al. (2011) describes the most common packing algorithms as follows:

Next-Fit (NF): The item is put in the active bin if it fits. Otherwise it is put into a new bin and the new bin is marked as the active bin.

First-Fit (FF): All the non-empty bins are checked and the item is put into the first bin it fits. Otherwise it is put into a new bin.

Best-Fit (BF): The item is put into a bin that is filled to maximum degree but still having enough vacant space for the item. Otherwise the item is put into a new bin.

First-Fit Decreasing (FFD): The items are sorted in non-increasing order. Then, they are packed according to FF.

Best-Fit Decreasing (BFD): The items are sorted in non-increasing order. Then, they are packed according to BF.

The performance of the above algorithms depend on the sequence of items to be packed. Applying the Next-Fit algorithm on a sequence of items might produce the exact same results as applying the Best-Fit algorithm on a permutation of the same items.

Among these algorithm, BFD is the one with the most efficient packing. The main reasons are that the items are sorted and also the algorithm checks all the bins. However, checking all the bins is a negative attribute of most of these algorithms from the time standpoint. Next-Fit is the fastest algorithm because it only checks the active bin. This means if it is fed with a proper sequence its performance could be as good as BFD while it runs in a shorter time.

2.3. Cutting Problem in the Construction Industry

Although many research has been conducted on how to manage or recycle construction waste after being generated on the jobsite, very little research was focused on eliminating waste at the design phase or early stages of the construction. Some studies have discussed dimensional coordination and standardization, minimizing the use of temporary works or avoiding late design modifications (Poon, 2007) but very few studies have proposed innovative solutions.

Manrique et al. (2011) studied the problem of optimizing lumber waste in framing designs. In this study, a combinatorial algorithm is proposed which constructs all possible solutions of cutting lumber, and then computes the amount of leftover associated with each possible solution. Finally, it finds and proposes the most optimal solution. In this method each solution is a set of cutting layouts of lumbers. The developed algorithm is called CUTEX and it uses the framing layouts generated by another algorithm called FRAMEX (Manrique et al., 2007). FRAMEX used the 3D-CAD models to generate the framing layouts for wood light frame residential buildings. CUTEX not only generates cutting schedule for lumbers but also it generates cutting layout for plywood. As the result,

the wood waste for studied projects was reduced by 96 percent. However, the combination of these two algorithms could be implemented only on small sets (i.e. small projects). Therefore, Manrique (2009) has suggested using evolutionary algorithms to find optimal cutting layouts.

Shahin and Salem (2004) studied one-dimensional cutting stock problem in the construction industry. They developed an optimization application based on genetic algorithms. It can be implemented in solving the one-dimensional cutting stock problem in the construction industry no matter the size of data. The data input for this algorithm consists of first, the table of standard lengths of the stock and second, the table of required lengths and the number of times each one is needed. Then, the algorithm generates the optimal cutting schedule. This method does not use CAD or BIM capabilities. This method is taking advantage of Genetic Algorithms which make it possible to process larger sets of data in a relatively short amount of time. However, as it is not using BIM or CAD, it does not automatically extract data from a model and in case that the number or composition of the required pieces are not defined or optimized, it is not feasible to utilize this method.

2.4. Evolutionary Algorithms

Solving a problem can be perceived as a search through a space of potential solutions. This search is supposed to result in the best solution. For small spaces, exhaustive methods that include comparing each and every one of the potential solutions, usually perform well. However, for larger spaces, a different approach should be used.

Evolutionary Algorithms are based on the concept of simulating the evolution of individual structures based on their perceived performance (fitness). EAs maintain a population of structures that evolve based on nature-inspired operations and focus on exploiting the available fitness information while exploring the search space.

The evolutionary algorithm maintains a population of individuals for iteration t. Each individual represents a potential solution to the problem at hand. Each solution is evaluated to give some measure of its "fitness". Members of the population undergo transformations to form new solutions (Exploration). For example crossover is a transformation, which create new individuals by combining parts from several (two or more) individuals. Or mutation is another transformation, which create new individuals by a small change in a single individual. Then, a new population (iteration t + 1) is formed by selecting the more fit individuals (Figure 3). After some number of generations the algorithm converges and it is hoped that the best individual represents a near-optimum (reasonable) solution (Dasgupta & Michalewicz, 1997).

The origins of evolutionary algorithms can be traced to at least the 1950s. However, the three most historically significant methodologies are "evolutionary programming", "evolution strategies", and "genetic algorithms" (Spears, 2000).

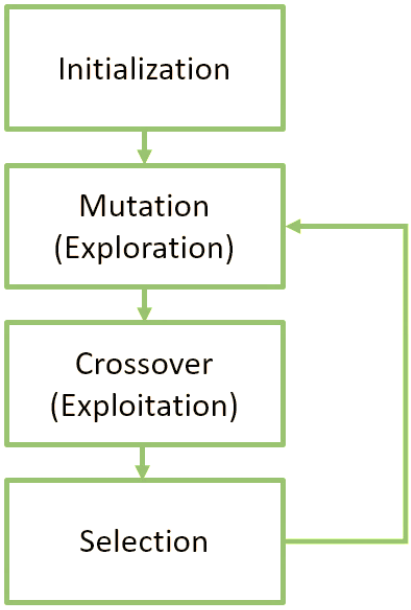


Figure 3: Main stages of an evolutionary algorithm

2.4.1. Differential Evolution

Differential Evolution is one the most recent branches of Evolutionary Algorithms. DE is based on the same principles as EA and while being simple, it is one of the most powerful tools for optimization (Feoktistov, 2006).

DE is mostly utilized for continuous optimization, where the search space is a continuous space instead of being limited by a finite number of feasible solutions (Feoktistov, 2006). DE in its simplest form, performs as the following.

- Consider solutions to the problem look like: $x = \{x_1, x_2, x_3, \ldots, x_s\}$ and solutions can be evaluated based on a fitness function: $f(x)$
- Initialization:

  A set of random solutions are generated

- Mutation:

  For each solution x in the population, three other solutions are randomly selected from the population → a , b , c

  Mutant vector is constructed v = a + F (b – c) → F is a constant ϵ [0,2]

- Crossover:

  Trial vector u is constructed

  $\forall\, i \in (1, s)$ Pick a random probability $rand \in (0, 1)$ → $u_i = \begin{cases} v_i, & rand \leq CR \\ x_i, & rand > CR \end{cases}$

- Selection:

  If f(u) is better than f(x), x is replaced with u

- The same is done for all solutions in the first iteration and next iteration runs

3. RESEARCH METHODOLOGY

There is an opportunity to reduce construction waste through generating optimized layouts to cut drywall panels. Similar problem has been already studied and addressed in other industries by utilizing EAs.

This study investigates if it is feasible to implement EA-based optimization methods on a BIM platform and develop an automated optimization tool. Then, the developed tool can be utilized to generate optimized cutting layouts for drywall panels and the leftover material could be minimized.

To answer the research question and fulfill the main objective of the study, it is designed to have two main stages:

- To develop a Revit add-in that identifies the required pieces of drywall and also generates the optimized layouts to cut such pieces out of full panels.
- To test the developed optimization tool on building information models and evaluate the generated cutting layouts in terms of leftover and time.

The way each stage was implemented and the details regarding methods, challenges and complications are explained in the following sections.

3.1. Developing the Add-in

Autodesk Revit provides a platform to automate processes such as optimization by implementing algorithms through its Application Programming Interface (API). The Revit .NET API allows to program with any .NET compliant language such as C#. Using Revit API, it is possible to gain access to all the embedded data in the model, analyze and edit the data and also create model elements. In other words, it enables us to create add-ins to automate repetitive tasks.

To design and create the optimization tool, a study was conducted on the various optimization methods proposed for similar problems in other industries. Based on the literature review, it was decided to follow a hybrid approach. Next-Fit packing algorithm was chosen considering that it is the fastest packing algorithm while its performance is equally desirable if it is fed with a certain sequence. This sequence can be called the optimized sequence and it can be determined through an evolutionary algorithm.

The algorithm has two main phases. In the first phase, it extracts geometrical data from BIM and having the dimensions of available drywall panel(s), it finds the most desirable arrangement of panels. The number of full panels is calculated and the outline of smaller pieces that need to be cut out of full panels are also determined and a stack of two dimensional polygons representing the pieces is established.

The objective of the second phase is to generate layouts to cut these pieces out of full panels in the way that leftover is minimized. This is the same as minimizing the number of full panels required to cut the pieces. Therefore, the algorithm is supposed to

solve a 2D bin packing problem. Required pieces are the items that have to be packed and the panels are the equal-sized bins.

The second phase of algorithm is designed based on Next-Fit packing. In Next-Fit packing, the item is put into the current bin if it fits in the bin. Otherwise a new bin is used which is declared as the current bin (Johnson, 1974).

There are three main procedures in the second phase. The packing procedure implements Next-Fit algorithm and "puts" items into bins. The fitting procedure is the one that decides whether the item fits into the bin or not. It also determines the location, the rotation factor and the mirror factor of the item upon placement. The fitting procedure is designed based on differential evolution.

The third procedure in this phase is sorting procedure. It is an EA-based algorithm that find the optimal sequence of items and is designed based on what Blum and Schmid (2013) have proposed.

The details of the first phase and the procedures of the second phase along with the challenges, are explained in the following sections.

.

3.1.1. Identification of the Required Pieces

The first step in the process of hanging drywall panels is planning the job. In this step, the crew determine the materials and the application method. They also measure the surfaces, determine the starting point and do markings. It is in this step that they decide to hang the panels horizontally (long dimension across studs or joists) or vertically (long dimension parallel to studs or joists). Moreover, they may decide to offset end joints in adjacent rows hanging the panels following a staggered pattern.

Similarly, the beginning part of the algorithm aims to determine the application method and the starting point. The building information model already contains the geometry of surfaces and all their dimensions. Dimensions of available panels are received from the user as inputs.

It is assumed that all the interior surfaces need to be covered by drywall. For each wall in the Revit model, based on its type parameter "Function", the faces that need to be covered are selected. If "Function" is interior, both interior and exterior faces of the wall and if "Function" is exterior, only interior face of the wall is selected and the following analysis is performed on it.

Firstly, for each face, a basepoint is determined in the way that it is the bottom-left corner of the face's bounding rectangle (Figure 4). Considering that the face can be represented by a polygon, the height of the basepoint is equal to the height of the lowest vertex. And its two other coordinates are the same as the leftmost vertex of the polygon. Then a coordinate transformation is performed and the face is represented with a 2-dimensional polygon with the basepoint located at the origin.

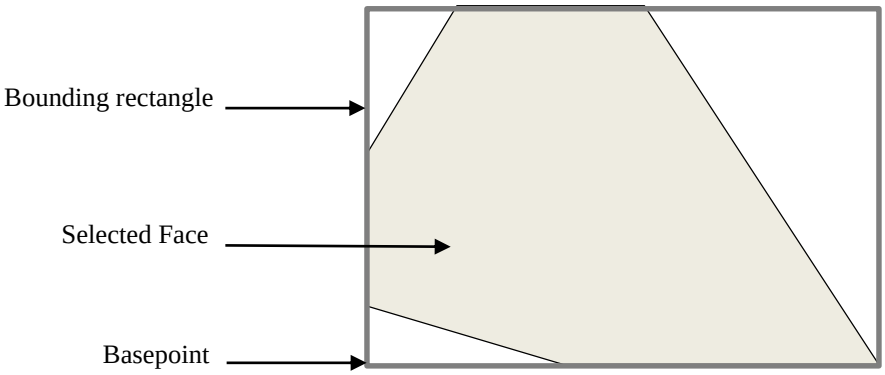


Figure 4: Location of the basepoint

Suppose there is a matrix of drywall panels with the specified dimensions and specified orientation on this face. The bottom left corner of the first panel in the matrix is defined as the starting point of the matrix. One possible arrangement is that the starting point and the basepoint are identical (Figure 5). For each panel, its intersection with the face can be determined. If the intersection and the panel are identical, it will be counted toward the number of full panels needed to cover that face. Otherwise, the intersection is stacked as one of the required pieces to cover the face.

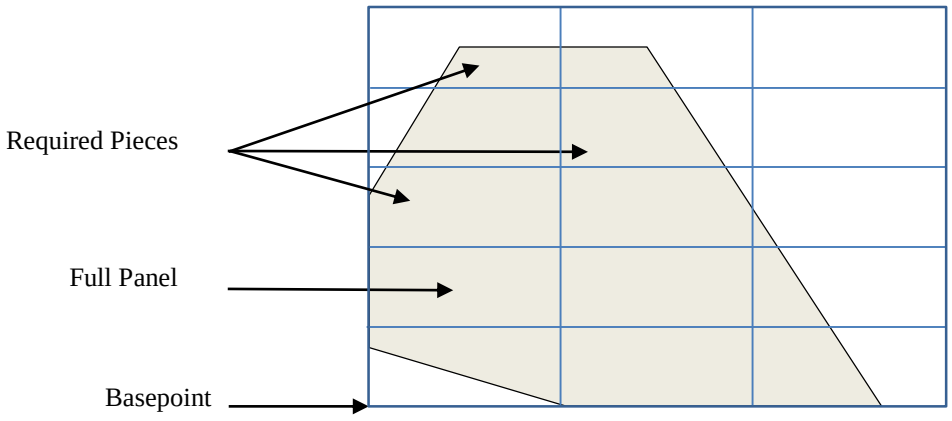


Figure 5: Matrix of drywall panels

The starting point is not necessarily on the basepoint. It can be any point on the plane. However, it can be assumed that the starting point is within a certain rectangle. This rectangle has the same dimensions as a drywall panel and its top right corner is located at the basepoint. Any starting point outside this rectangle corresponds to a starting point inside this rectangle (Figure 6).

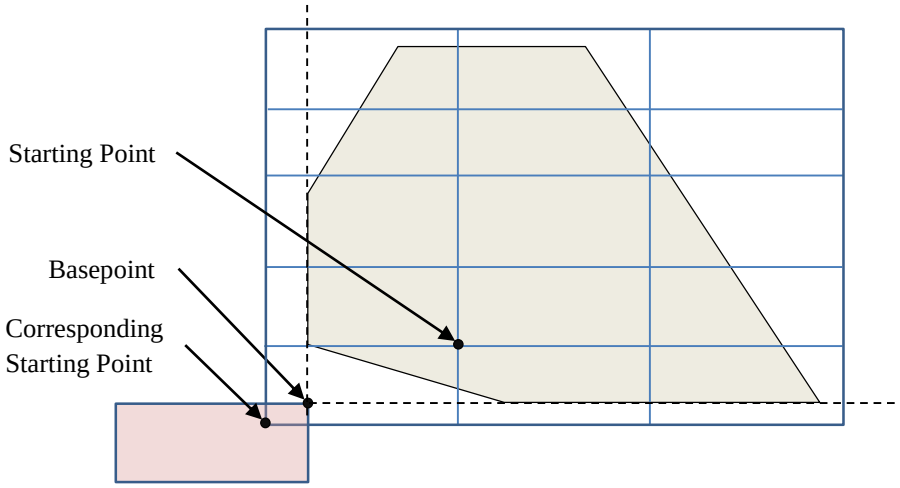


Figure 6: Location of starting point

Theoretically there is unlimited number of points in the mentioned rectangle and it is not possible to analyze all of them. Therefore, the algorithm analyzes a limited number of starting points. The basepoint and all points in increments of one inch are considered as possible starting points. As an example, for 10’ × 4’ drywall panels, the number of points the algorithm analyzes is: $(10\times12) \times (4\times12) = 5{,}760$

The corresponding arrangements of possible starting points are evaluated based on the number of full panels in each arrangement. The one with the maximum number of full panels is the most desirable.

For panels with different dimensions or different orientation – horizontal vs. vertical – the same analysis is performed using the corresponding matrix (Figure 7). Then the most desirable arrangements of different scenarios are compared based on the same criteria. Eventually for the given face, an arrangement with the maximum number of full panels is determined and the corresponding required pieces are added to the main stack of stencils. If the maximum number of full panels for two scenarios are the same, the one with the minimum number of stencils has priority.

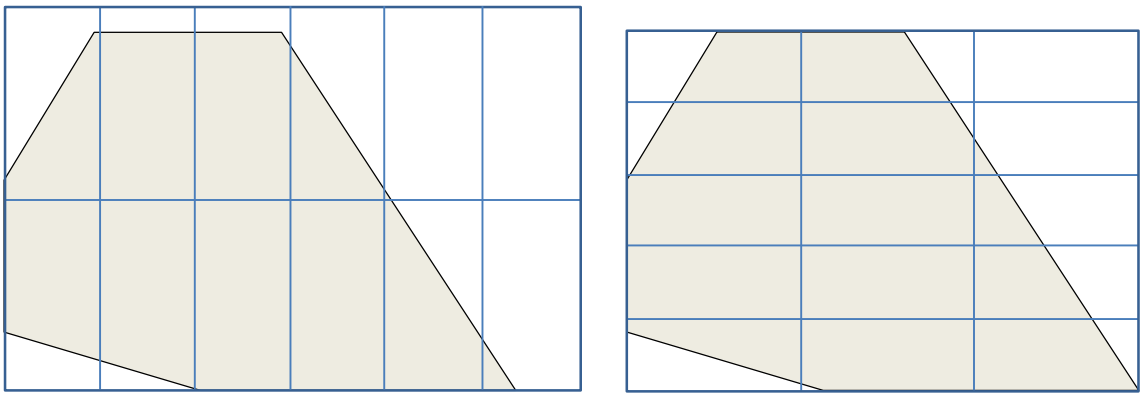


Figure 7: Arrangements of panels with different sizes and orientation

The same process of handling the faces, finding the most desirable arrangement of panels to cover the faces and adding the required stencils to the main stack is performed on each and every wall selected from the Revit model.

Eventually, a stack of 2D polygons representing the required pieces to cover the walls is available. For the purpose of geometry handling, a basepoint is determined for each polygon in the main stack of pieces. The bottom-left corner of the polygon's bounding rectangle is declared as its basepoint.

3.1.2. Packing Procedure

In the “Packing” procedure, items are packed into bins based on Next-Fit packing. In other words, “Packing” procedure aims to generate the cutting layouts by performing Next-Fit algorithm on a certain sequence of items that is already generated. Items are packed into bins one by one following the order of the input sequence.

Such sequence of items is the input of the procedure and the output is a set of layouts showing how the pieces are packed based on the input. Also, an objective value is reported which indicates how desirable the packing is using the specific input sequence. This value is used to compare different sequences.

The items are represented by polygons (not necessarily regular) that were stacked in the previous phase. Since each bin refers to a drywall panel with specified dimensions and orientation, they are geometrically represented by rectangles of the same dimensions. The dimension of the available drywall panel along the *x* axis is denoted by L_{panel} and its dimension along the *y* axis is denoted by W_{panel}.

Packing procedure is an iterative process which starts with fitting the first item into the first bin. Then, in each iteration, the algorithm performs the fitting procedure on the next item. The fitting procedure which is explained in the following section is an attempt to find the optimal position of the item following certain criteria. If the item does not fit into the current bin according to the fitting procedure, the algorithm fits the item into the next bin.

The fitting procedure reports an objective value (F) indicating how desirable the placement of each item is. In the “Packing” procedure, Index of Packing (denoted by IP)

for each bin is calculated to be the average of the objective values reported by fitting procesure for each item in that bin:

$$IP_{j\in B} = Avg.(F_i)$$

where:

B is the set of bins and the corresponding layouts created

F_i is the objective value reported by fitting procedure for item *i* in the *j*-th bin.

The objective of packing procedure is to provide an opportunity to compare different input sequences. The ultimate objective is to minimize the number of bins used. Therefore, it is the main criterion to evaluate input sequences. However, a secondary function in required to compare the input sequences that use the same number of bins. Therefore, the following objective function is designed to compare the input sequences that use the same number of bins:

$$\mathrm{E(sequence)} = \sum_{j\in B} IP_j$$

Larger values of IP_j indicate that the placements of the items in the *j*-th bin are more desirable. As the result between two input sequences that use the same number of bins, the one with larger E(sequence) value is more desirable.

3.1.3. Fitting Procedure

The fitting procedure aims to place an item into a bin considering that there are already other items in the bin. To place the item in the bin, the following values has to be determined:

a) The horizontal (x) and the vertical (y) coordinates of the item basepoint – these values define the position of the item and their range is limited to specified dimensions of drywall panels.

b) The rotation angle (r) of the item upon placement – based on the practical routines in the industry, only angles of 0, 90, 180, 270 degrees are considered in this study and are represented by values 0, 1, 2, 3.

c) The mirror factor (m) – this value indicates whether the item is mirrored upon placement or not and its value is either 0 (not mirrored) or 1 (mirrored).

Therefore, the solution is of the form s = [x, y, r, m] and clearly, there are unlimited number of possible solutions – i.e. a large search space (Figure 8).

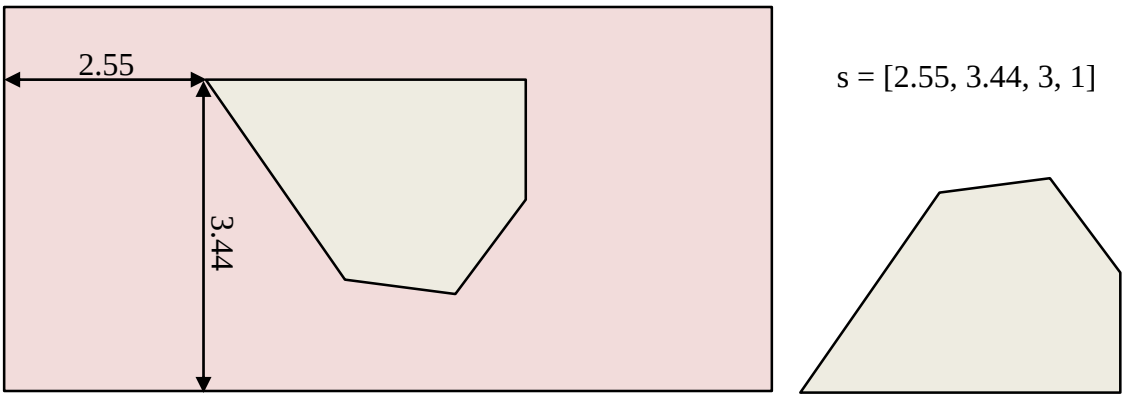


Figure 8: A potential solution in fitting procedure

Since it is not feasible to compare all the possible solutions using an exhaustive search, an evolutionary algorithm is utilized. Differential Evolution is selected because it is an appropriate evolutionary algorithm for continuous search spaces and also its implementation is relatively simple (Figure 9).

The DE algorithm is designed based on the following:

- The population size N is specified by the user. Since the possible solutions have four parameters (x, y, r, m), the parameter vectors representing the possible solutions have the form:

  $s_i = (x_i , y_i , r_i , m_i)$ $i = 1, 2, \dots , N.$

- The initial parameter vectors are generated randomly in the following ranges:

  $x_i \in [0, L_{panel}]$ $y_i \in [0, W_{panel}]$ $r_i \in \{0, 1, 2, 3\}$ $m_i \in [0, 1]$

- For a given parameter vector s_i in the population, three other vectors s_a , s_b , s_c are randomly selected such that indices i, a, b, c are distinct and mutant vector v_i is constructed such that:

  $v_i = s_a + F (s_b - s_c)$

- To keep the parameters within the appropriate ranges, if any parameter in v_i is not in the corresponding range, it is replaced by a random number in that range.

- Trial vector u_i is constructed through crossover between s_i and v_i

- If trial vector u_i has an equal or lower objective function value than that of its target vector, s_i , it replaces the target vector in the next generation.

Mutation:

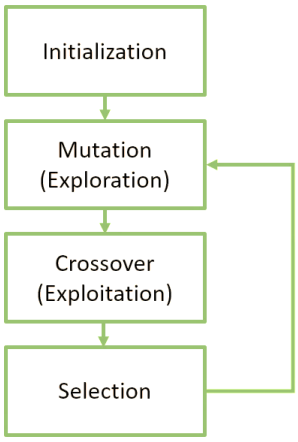
For each solution x in the population, three other solutions are randomly selected → a , b , c

Mutant vector is constructed $v_i = s_a + F\ (s_b - s_c)$

→ F is a constant ϵ [0,2]

Crossover:

Trial vector u is constructed



$\forall\ i \in (1, s)$ pick a random probability $rand \in (0,1)$→ $u_i = \begin{cases} v_i, & rand \leq CR \\ x_i, & rand > CR \end{cases}$

Figure 9: Mutation and crossover in DE

The objective function is proposed based on the criteria upon which the placement should be evaluated. The placement of the item is desirable when:

a) It is completely inside the bin.

b) It does not have any overlaps with other items that are already in the bin.

c) Its distance to the bottom left corner of the bin is the minimum possible.

d) Total length of its common sides with the items already in the bin is maximum.

For each placement of the item that is represented by a parameter vector s_i , three indices are calculated to measure the desirability of the solution.

Index of Satisfactory placement (denoted by IS) as its name implies is the main factor to determine the desirability of the placement. The requirements of a desirable placement of the item are (a) the item is completely inside the bin (b) the item upon

placement does not have any overlaps with other items that are already in the bin (Figure 10). In other words, if these two requirements are not met, the placement is not desirable at all. The proposed formula to calculate IS is the following:

$$IS = \frac{S_{in} - \mu S_{out}}{S_{in} + \mu S_{out}}$$

where:

S_{out} = Total area of overlaps the item has with other items in the bin + Total area of regions of the item that are not inside the bin

S_{in} = Total area of the item – S_{out}

μ = Constant of sensitivity

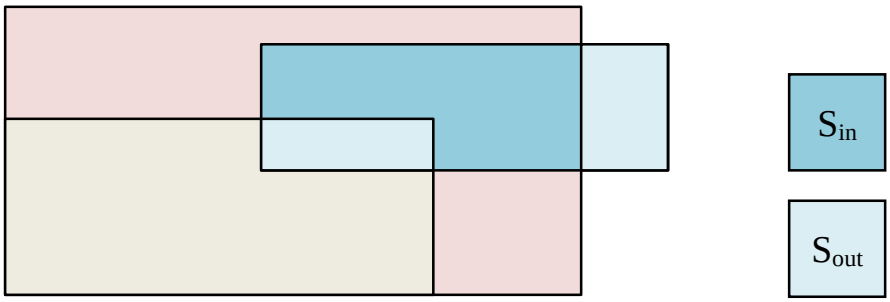


Figure 10: Index of satisfactory placement

Based on the formula if both requirements (a) and (b) are met then $S_{out} = 0$ and therefore IS = 1. Otherwise IS <1 and the larger value for μ results in IS being more sensitive to S_{out}. For example assume μ=2, if only one third of the item area is outside the bin or overlapped the value for IS would be zero. Upon testing different values for IS, it was decided that μ=5 results in a relatively better performance of the algorithm because it is neither too strict nor lenient.

Index of Distance (denoted by ID) indicates how close the item is to the bottom left corner of the bin. In other words it evaluates the third criterion of desirability (Figure 11). The proposed formula to calculate ID is the following:

$$\text{IS} = 1 - \frac{d}{D}$$

where:

d = The Euclidean distance between the basepoint of the item and the bottom left corner of the bin $\quad d = \sqrt{{x_i}^2 + {y_i}^2}$

D = The maximum Euclidean distance that the basepoint of the item can assume to the bottom left corner $\quad D = \sqrt{{L_{panel}}^2 + {W_{panel}}^2}$

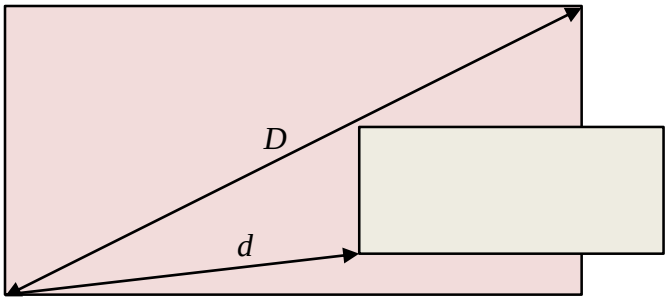


Figure 11: Index of distance

Based on the formula, if the basepoint of the item is placed at the bottom left corner of the bin then ID = 1 and if it is placed at the top right corner of the bin which is the least desirable, ID = 0 and therefore always: $0 \leq \text{ID} \leq 1$

Clearly $\frac{d}{D}$ could report the closeness of the item to the bottom left corner of the bin but the indices should be designed in such way that they behave similarly. In other words for the most desirable placement they all should be maximum.

Index of Adjacency (denoted by IA) evaluates how well criterion (d) of desirability is met. In other words it indicates how much is the length of the common sides between the item and other items in the bin and the bin itself. IA is supposed to make the more packed arrangements preferable (Figure 12).

The proposed formula to calculate ID is the following:

$$\text{IA} = \sum_{k \in M} S_{out_k}$$

where:

M = A set of small-scale movements (within 0.5") toward up, down and left

S_{out_k} = Total area of overlaps the item would have under movement k +

Total area of regions that would be outside the bin under movement k

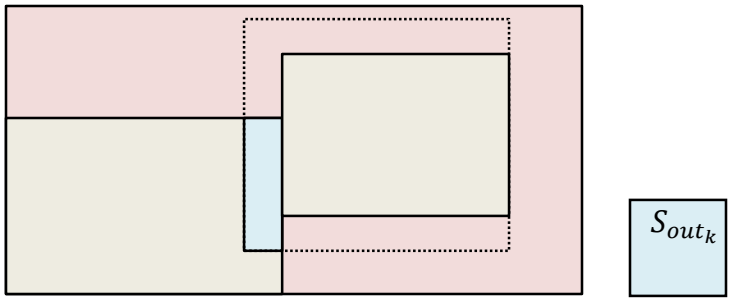


Figure 12: Index of adjacency

The larger values of IA indicate that the item, upon placement, has more common sides with other items in the bin and/or the bin itself. The maximum and minimum value for IA depends on the size of item and also the number and location of other items in the bin. However, it is designed in such way that the value is in the range of [0, 1] for pieces with common sizes.

As an example, imagine a piece that is slightly smaller than the full panel and the panel size is 4’ × 10’. Then it can be assumed that the required piece is a 4’×10’ rectangle. There is only one way to fit the piece into the bin and there would be no other items into that bin (Figure 13). In such case if the panel is moved 0.5” upward some portion of it would be outside the panel and the area of that portion is: 0.5” × 10’ ≃ 0.417 sf

Similarly, if it is moved 0.5” downward a portion of it with the same area would be outside the panel: 0.5” × 10’ ≃ 0.417 sf

If the panel is moved 0.5” toward left the area of the region outside the panel would be: 0.5” × 4’ ≃ 0.167 sf

Hence: IA ≃ 0.417 + 0.417 + 0.167 ≃ 1

Figure 13: Maximum value of IA

All indices are designed in a way that the higher values indicate more desirability. Since the value of each index is independent of other indices, the objective would be to maximize the summation of all indices. However, the indices do not have the same deciding role because for example an arrangement with no overlaps (large IS) but small IA is preferable to one with overlaps and large IA. Therefore, each index should have a different factor in the summation.

The objective function of the DE algorithm is designed as the following:

$$F(s) = (4 \times IS) + (0.5 \times ID) + (0.5 \times IA)$$

Or alternatively,

$$F(s) = (4 \times IS) + \frac{ID + IA}{2}$$

Upon testing the algorithm, it was decided to use the above factors because they results in relatively better performance of the algorithm. This process is similar to calibration.

Since both IA and ID are in the range of [0, 1], the value for $\frac{ID+IA}{2}$ is in the range of [0, 1]. However, IS has only an upper bound which happens when the item is placed fully inside the bin and does not have any overlaps with other items in the bin and in such case IS=1; Hence: $F(s) \geq 4$

Similarly, if $F(s) < 4$, it can be concluded that the item is overlapping with some other items or it is not fully inside the bin. Hence, it does not fit in the bin and based on Next-Fit algorithm it will be put into a new bin.

In other words, the proposed designation of objective function not only aims to find the most desirable placement of the item, but also it can determine if the item fits into the bin or not.

3.1.4. Sorting Procedure

The packing procedure was introduced as a process to generate layouts based on a sequence and evaluate that sequence. Sequence optimization procedure aims to find the sequence which results in the minimum number of bins being used. Note that any input sequence is a permutation of all items that must be packed.

If p is the number of pieces that must be packed and the pieces are assumed to be distinct, $p!$ is the number of permutations of those pieces. As an example, if 5 pieces must be packed, there are $5 \times 4 \times 3 \times 2 \times 1 = 120$ permutations of those pieces. Clearly, as the number of pieces increase, the number of permutations increase exponentially. Therefore, for large sets of items, it is not practical to generate all the permutations and compare them together. In this situation, Evolutionary Algorithms can be utilized and the literature shows they are capable of finding a nearly optimal solution in a relatively shorter time.

The proposed EA to find the nearly optimal sequence is based on what Blum and Schmid (2013) have proposed. They proposed the algorithm as part of a hybrid evolutionary algorithm to solve the 2D bin packing problem.

As mentioned, a solution in the context of this problem is an input sequence *s* for packing procedure. The first step of EA is generating the initial population of size P based on the following.

First, the input sequence in which items are ordered with respect to non-increasing area is denoted as the reference sequence. The position of an item in the reference sequence is called pos_i . Then a value v_i is calculated and assigned to each item i :

$$v_i = (n - pos_i)^2$$

To fill the positions of sequence s from 1 to n, an item is chosen randomly according to the following probability:

$$p(i) = \frac{v_i}{\sum_{i \in T} v_i}$$

where T is the set of items that are not yet assigned.

Selection of items based on this principle – i.e. fitness proportionate – is known as roulette wheel selection. As an example, consider that a sequence must be generated for three following pieces based on the above approach (Figure 14). Values pos_i and v_i are calculated:

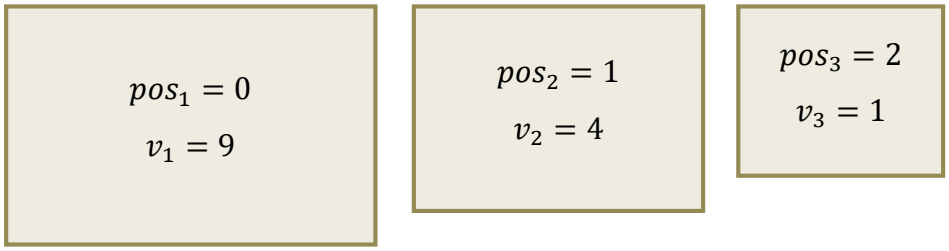


Figure 14: Selection of items in sorting procedure

Now to fill the first position of the sequence, one item is randomly selected. The probabilities to select items are: $\frac{9}{9+4+1}$, $\frac{4}{9+4+1}$, $\frac{1}{9+4+1}$

Assume the one in the middle is selected to fill the first position in the sequence. To fill the second position, one item from the other two is randomly selected based on the following probabilities: $\frac{9}{9+1}$, $\frac{1}{9+1}$

The last position is filled with the only item left and the probability is 1.

All P members of the initial population are generated according to the above steps. Based on this algorithm, the large pieces are more likely to show up in the early positions of the sequence.

The next step in the EA algorithm is the crossover operation which applies recombination to a certain number of population members. The number of solutions that go under crossover is determined by c_{rate} which is a parameter of the algorithm. For example in the proposed algorithm in this study, $c_{rate} = 0.7$ and it means that the best 70 percent of solutions go under crossover and they are denoted by P_c.

To perform the crossover, first the solutions in P_c are ranked based on their evaluation by packing procedure. For each solution s from P_c a crossover partner s_c is chosen from P_c by means of roulette wheel selection that was described in the initialization stage.

Given two solutions s and s_c, a new solution s_o, known as offspring is generated as explained in the following. Assume k, l and r to be the current positions in s, s_c and s_o respectively. The algorithms starts with $k = l = r = 1$ and in each iteration to fill r the following is done.

K and l are compared. If they are the same, r is filled with the same item and r is incremented. Also K and l move to the next positions until reaching an item which is not yet in s_o.

In case K and l are not the same, r is filled by randomly choosing between k and l with a probability of 0.75 given to the item from the better of the two solutions. Then r is

incremented. Also, either *k* or *l*, whichever was chosen to fill *r*, moves to the next positions until reaching an item which is not yet in s_o.

Figure 15 shows how an offspring is generated from two solutions – permutations of 1, 2, … , 7.

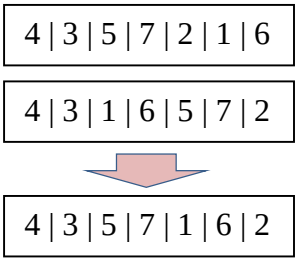


Figure 15: Order-based crossover in sorting procedure

Afterward, s_o is evaluated by packing procedure and if it is more desirable – i.e. smaller number of bins or larger value of E – than *s*, it replaces *s* in the population. When the crossover and comparison was performed on all the members in P_c , a number of new solutions are generated in the same way as the initial solutions in order that the number of solutions in the population stays the same.

3.2. Testing the Add-in

After developing the add-in, it was necessary to test it and evaluate its performance. This is different than debugging and the trial-error process to calibrate the objective functions.

The algorithm was tested at two different levels. In the first phase, the packing procedure, as the heart of the add-in, was tested being fed with one randomly generated sequence. The objective of this test is to evaluate the performance of differential evolution. DE was utilized as the optimization method to find the most desirable placement – including location, orientation and mirror factor – of an item into a bin. In the developed add-in, DE implements Next-Fit packing. Testing of the packing procedure was carried out in six rounds. The variable in each round was the number of iterations that DE runs to pack an item into a bin. For example, in the first round this number was set to 100. It means that for each item, DE runs 100 iterations to find the most desirable placement. Table 2 shows the number of iterations in each round.

Round	1	2	3	4	5	6
Number of Iterations	100	120	130	140	160	180

Table 2: Number of iterations in each round

After the packing procedure proved to have a satisfactory performance, the overall performance of the developed add-in was tested. Since, there was evidence confirming the acceptable performance of the packing procedure, this set of tests were an indicator of the performance of sorting procedure.

Testing the overall performance of the add-in was carried out in four rounds. The variable in each round was the number of iterations that EA algorithm of the sorting procedure runs to find the most desirable sequence of items. Number of iterations that DE algorithm in the fitting procedure runs was set to 180 based on the first phase of testing.

All tests were carried out on a simple Building Information Model containing a rectangle room with walls of the same height. Figure 16 illustrates the isometric view of the subject model. For the purpose of these tests, available drywall panels were assumed to be 4'× 10' and these dimensions were given to the add-in. All the walls were considered to be exterior walls. Therefore, their interior face were analyzed by the add-in. Phase one of the algorithm identified 20 pieces that need to be cut out of full panels.

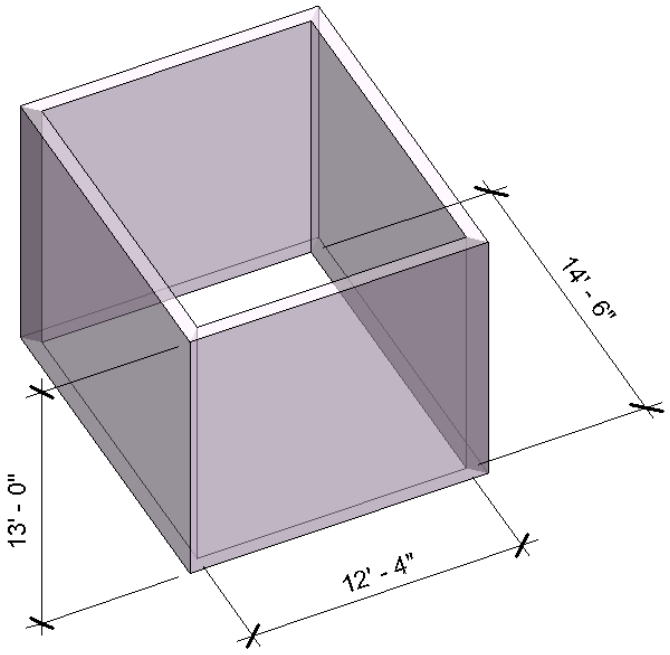


Figure 16: Subject building information model for testing

4. OUTCOME OF THE TEST

4.1. Layouts Generated by Packing Procedure

The following are the layouts generated by packing procedure (Figures 17 – 22).

Number of Iterations: 100

Processing Time: 2 min 30 sec

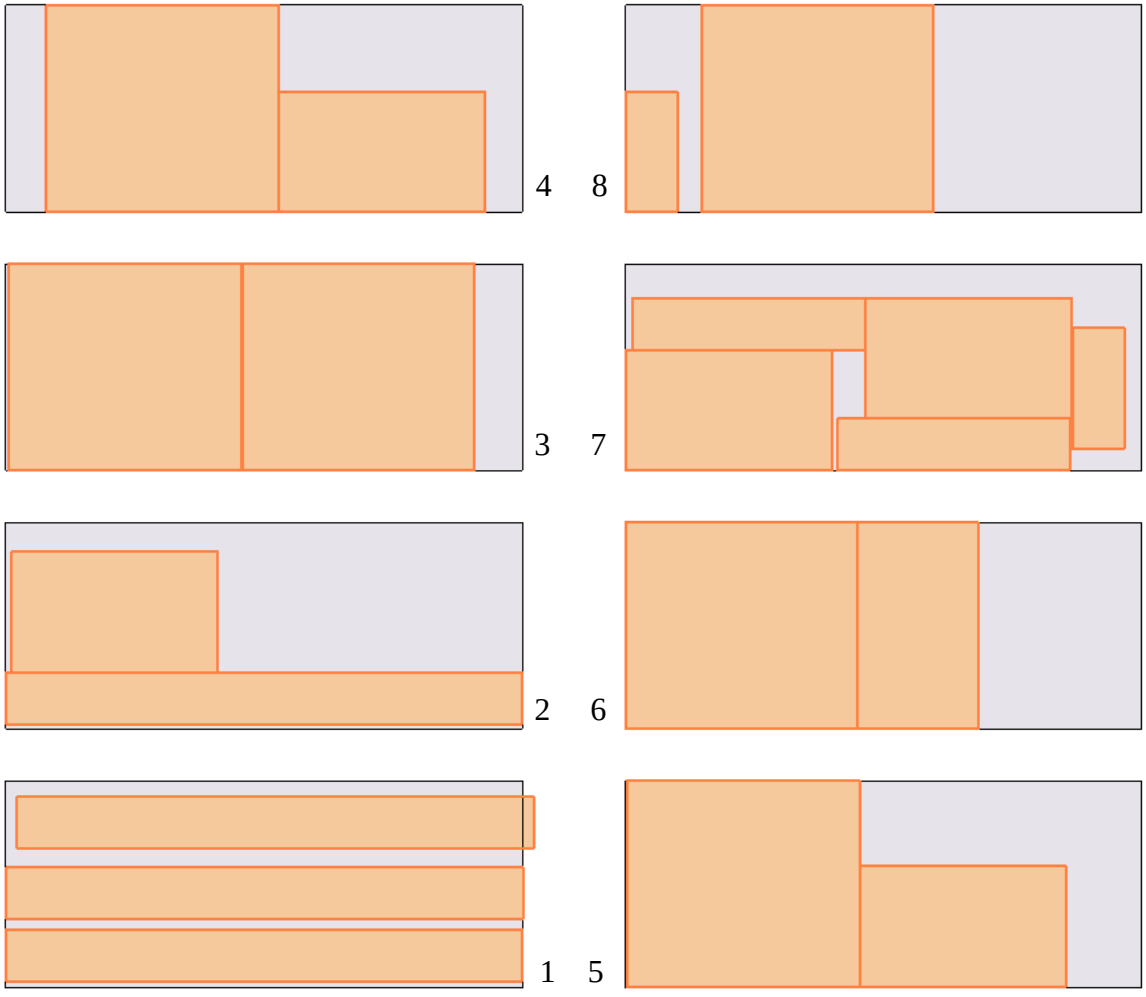


Figure 17: Layouts generated by packing procedure - 100 iterations

Number of Iterations: 120

Processing Time: 2 min 45 sec

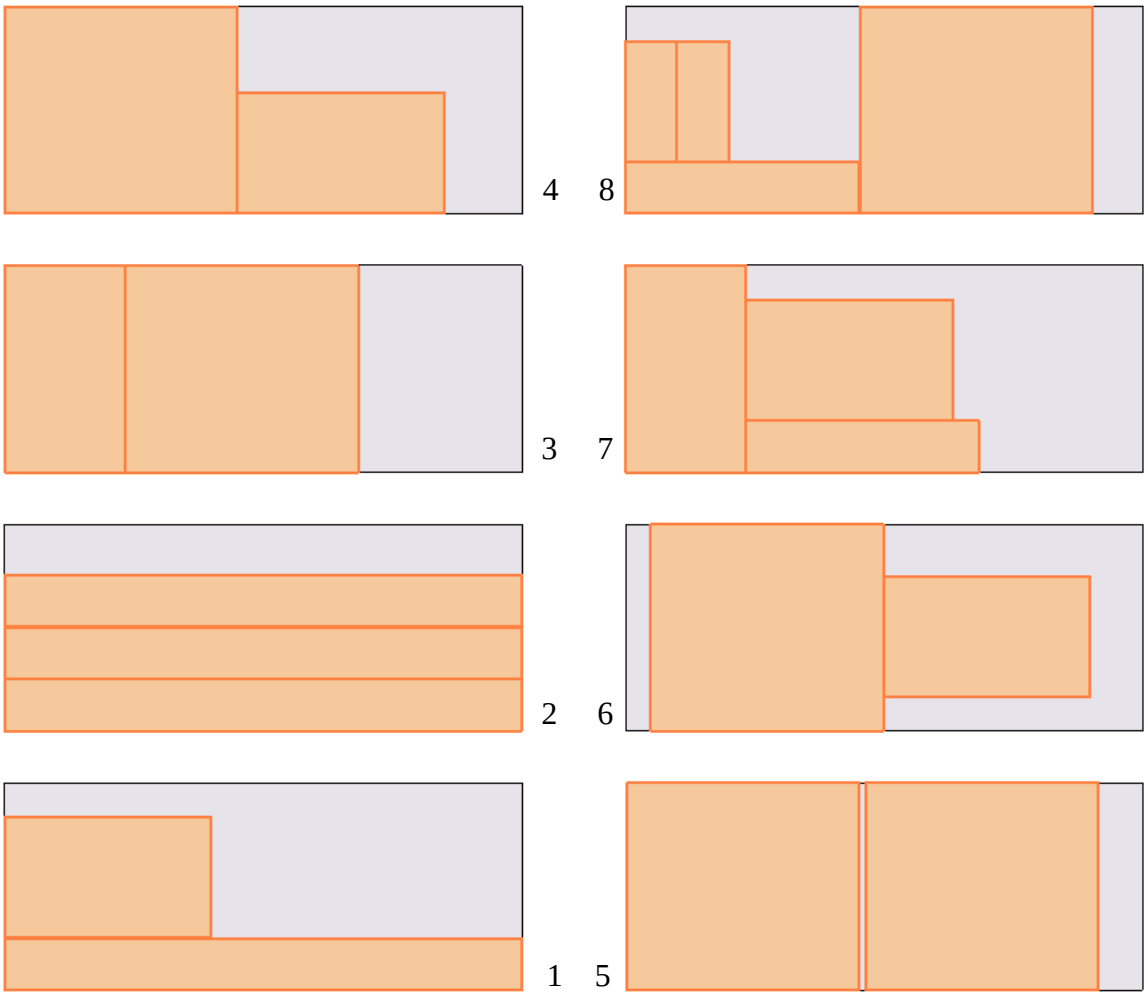


Figure 18: Layouts generated by packing procedure - 120 iterations

Number of Iterations: 130

Processing Time: 2 min 55 sec

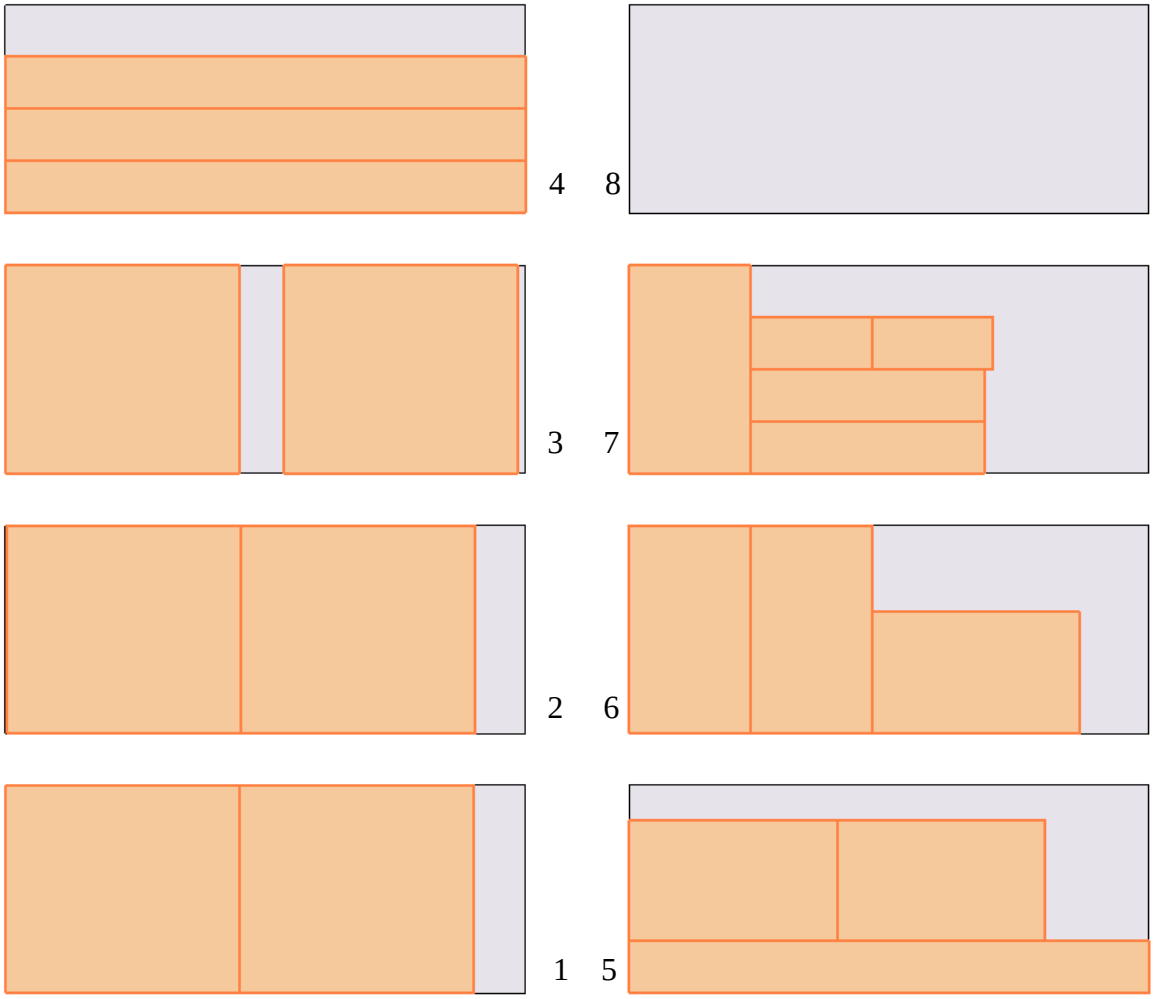


Figure 19: Layouts generated by packing procedure - 130 iterations

Number of Iterations: 140

Processing Time: 3 min 15 sec

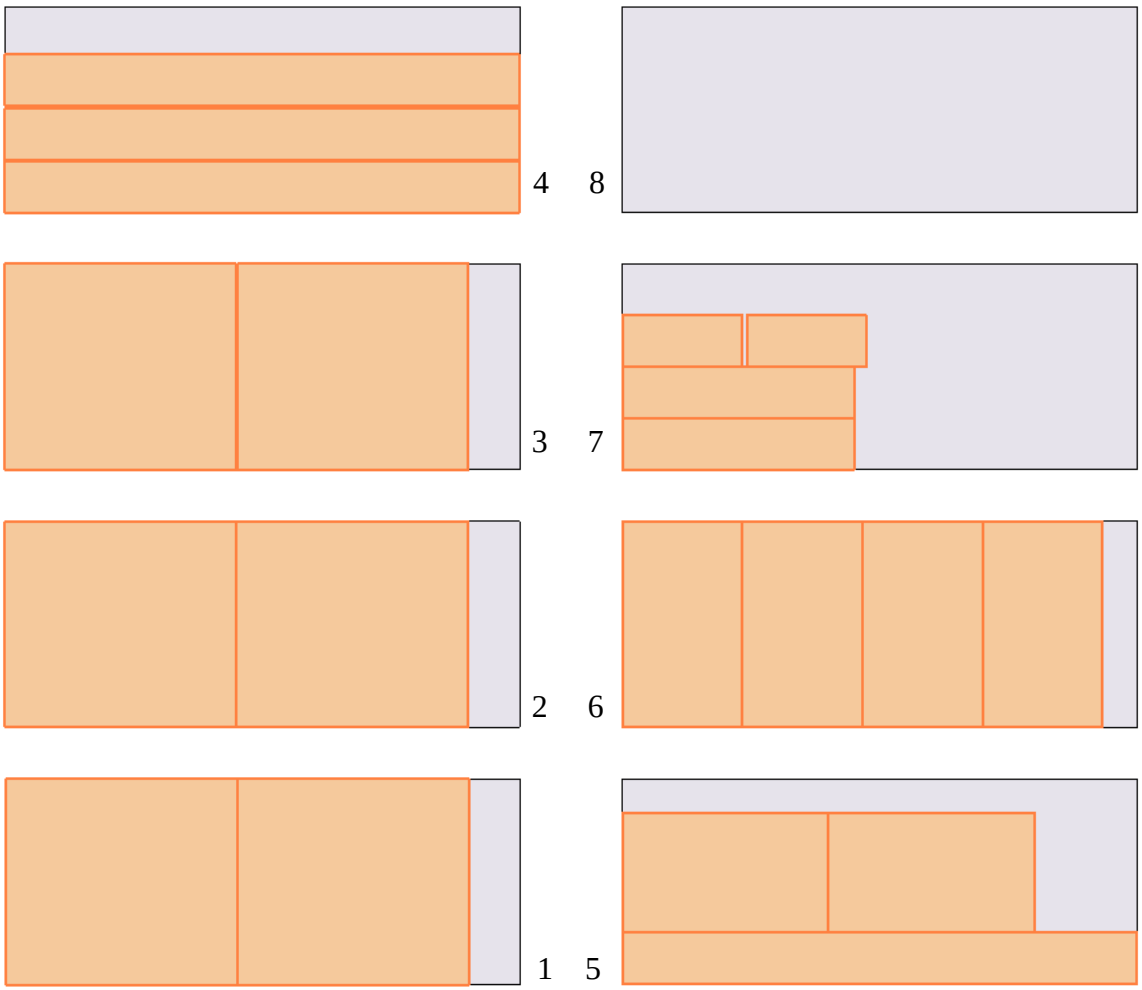


Figure 20: Layouts generated by packing procedure - 140 iterations

Number of Iterations: 160

Processing Time: 3 min 45 sec

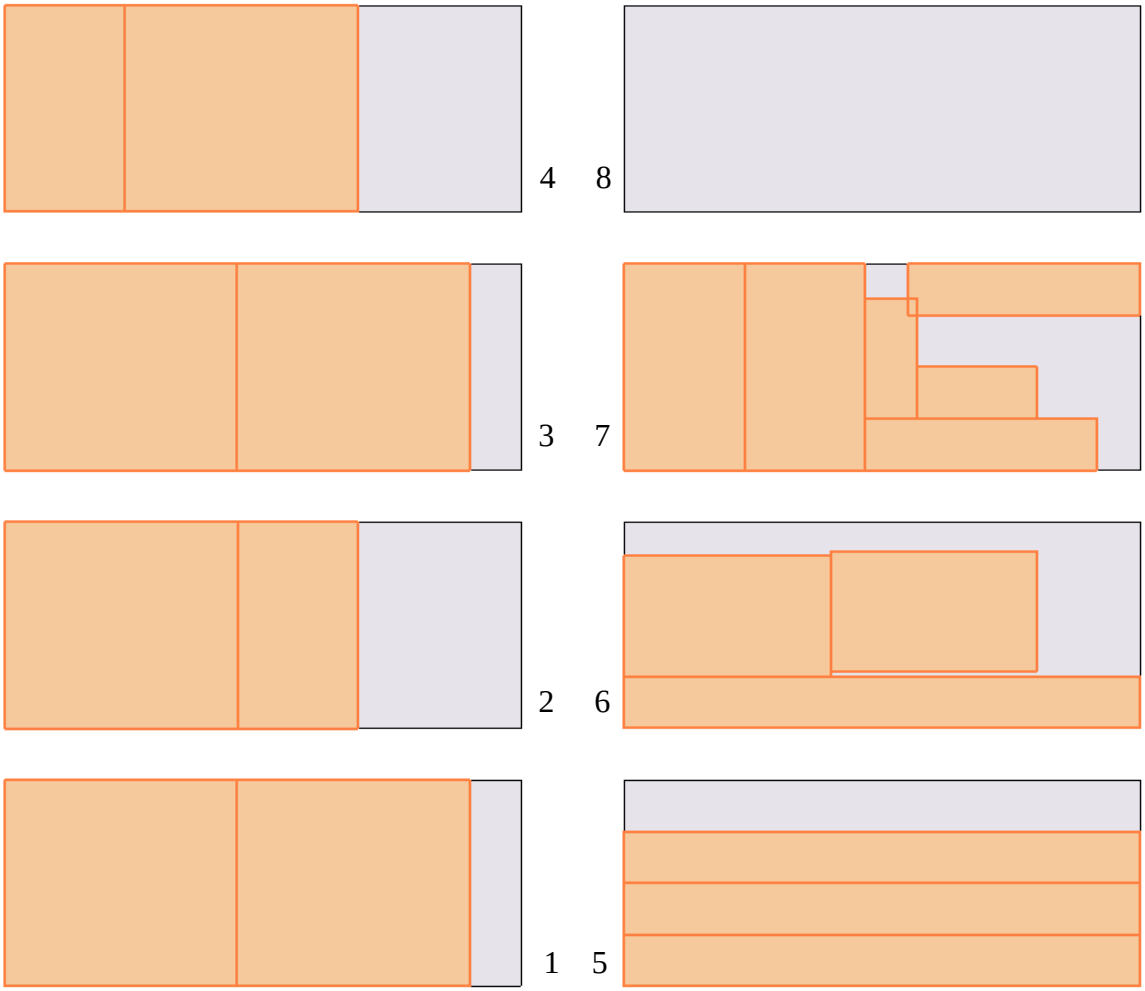


Figure 21: Layouts generated by packing procedure - 160 iterations

Number of Iterations: 180

Processing Time: 4 min

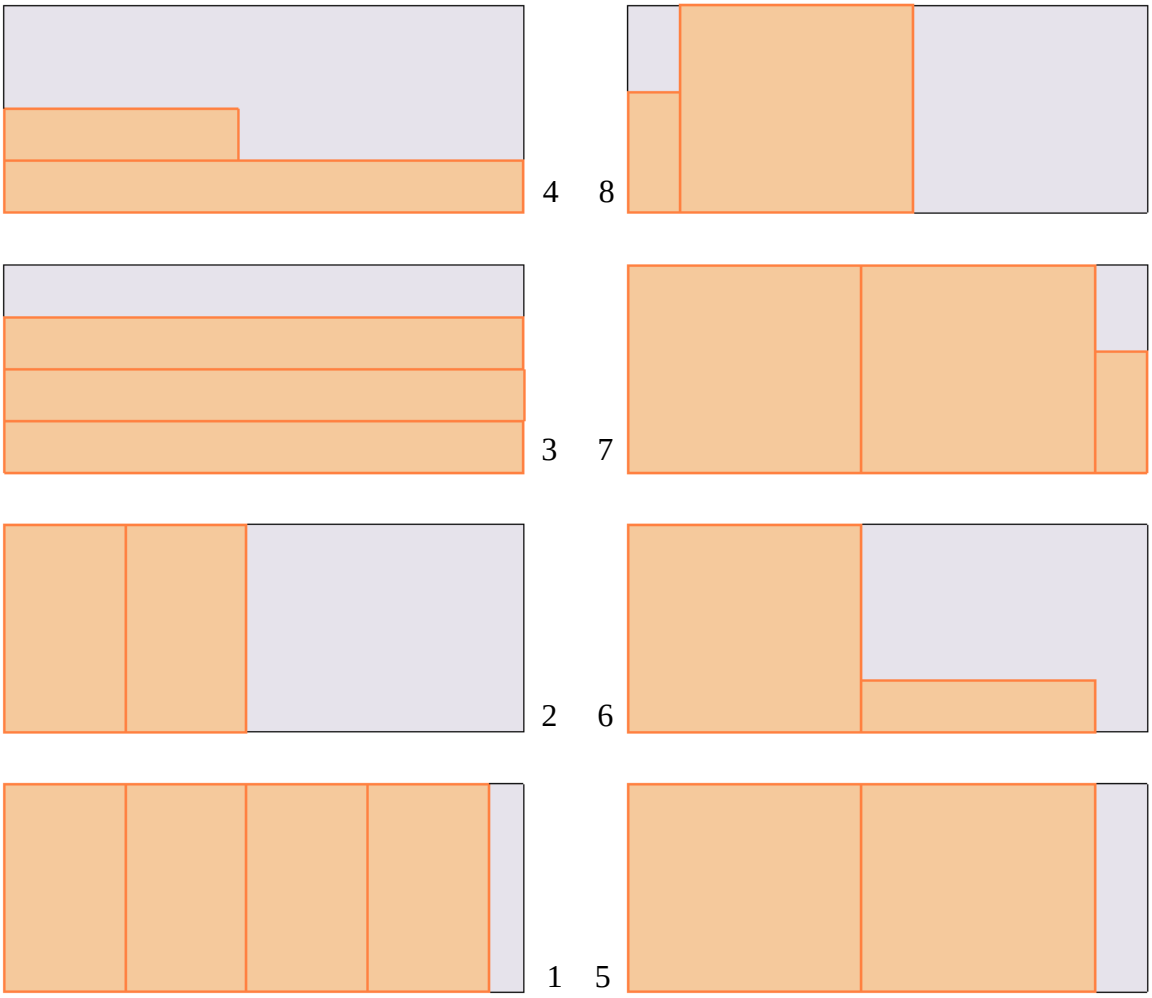


Figure 22: Layouts generated by packing procedure - 180 iterations

4.2. Analysis of the First Phase of Testing

The layouts generated by packing procedure demonstrate that increasing the number of iterations DE runs result in a noticeable improvement of layouts. As an example, in the layouts generated in the first round (100 iterations), items are not very well packed, there are vacant spaces between the items and more importantly, some items are not totally inside the bin. As the number of iterations increase the layouts become more packed and items are completely inside the bin with little vacant space between them. The layouts generated in the final round (180 iterations) are very well packed with no overlaps nor any vacant space between the items.

In these tests, only the performance of the packing procedure is evaluated. The algorithm does not optimize the sequence of the items and they are packed by a randomly generated sequence. Therefore, the increase in the number of iterations does not have any effect on the number of panels used or material wastage. Although in the last run the items are packed into eight panels, it is more desirable than the ones with seven panels.

Figure 23 demonstrates how the process time depends on the number of iterations.

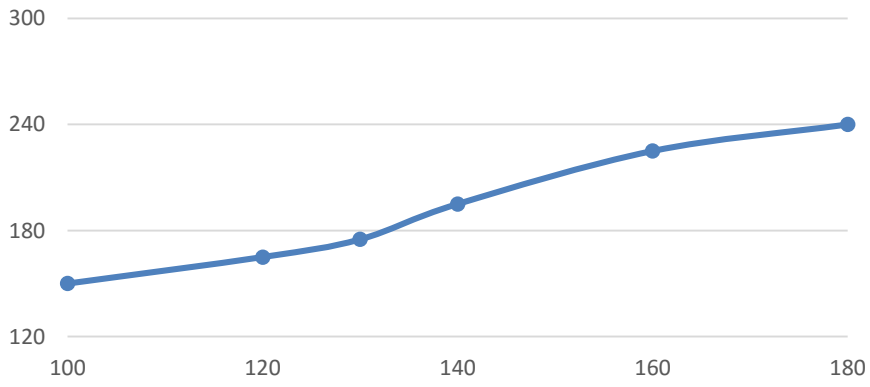


Figure 23: Processing time plotted by number of iterations DE runs

4.3. Optimized Cutting Layouts Generated by Add-in

The following are optimized layouts generated by the add-in (Figures 24 – 27).

Number of Iterations: 10

Processing Time: 20 min

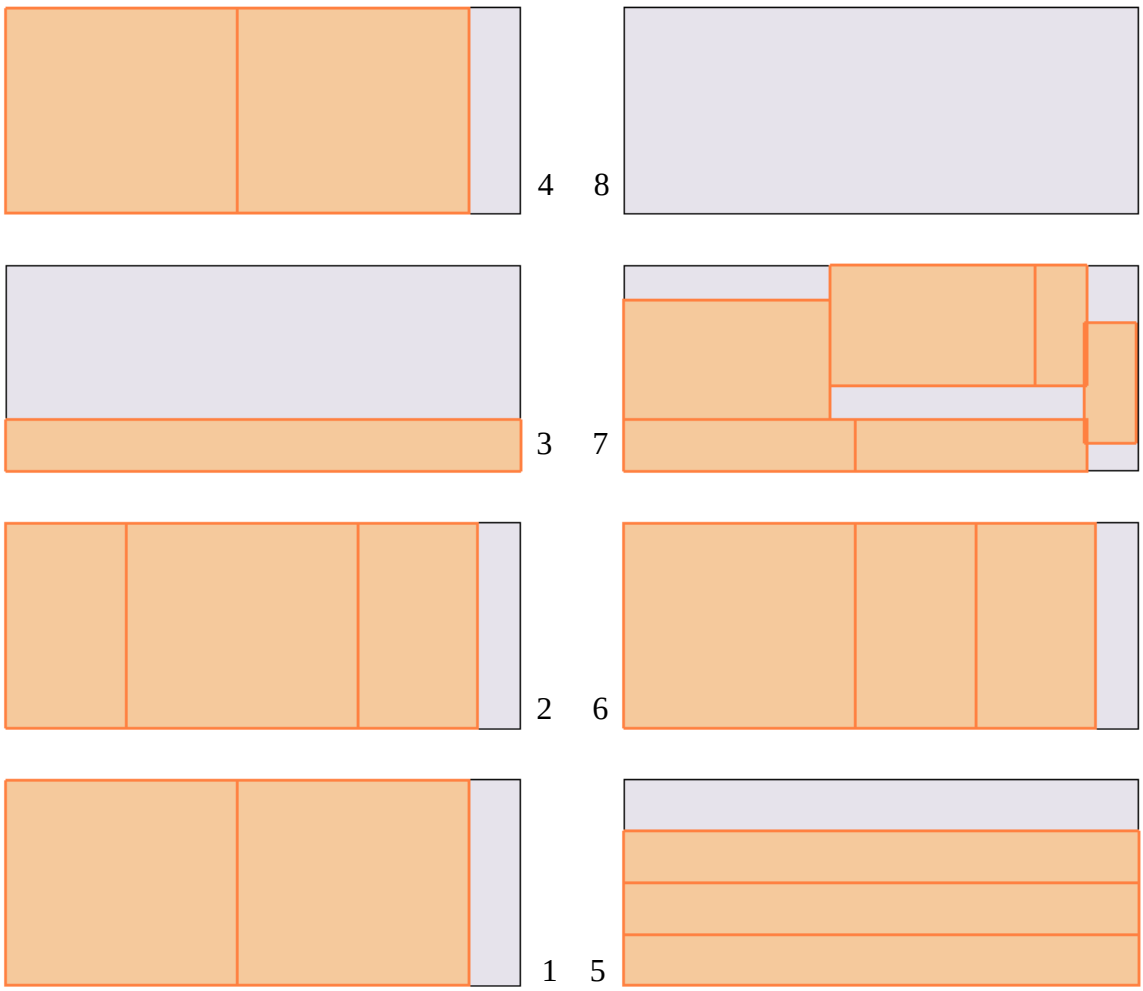


Figure 24: Optimized cutting layouts generated by add-in - 10 iterations

Number of Iterations: 20

Processing Time: 40 min

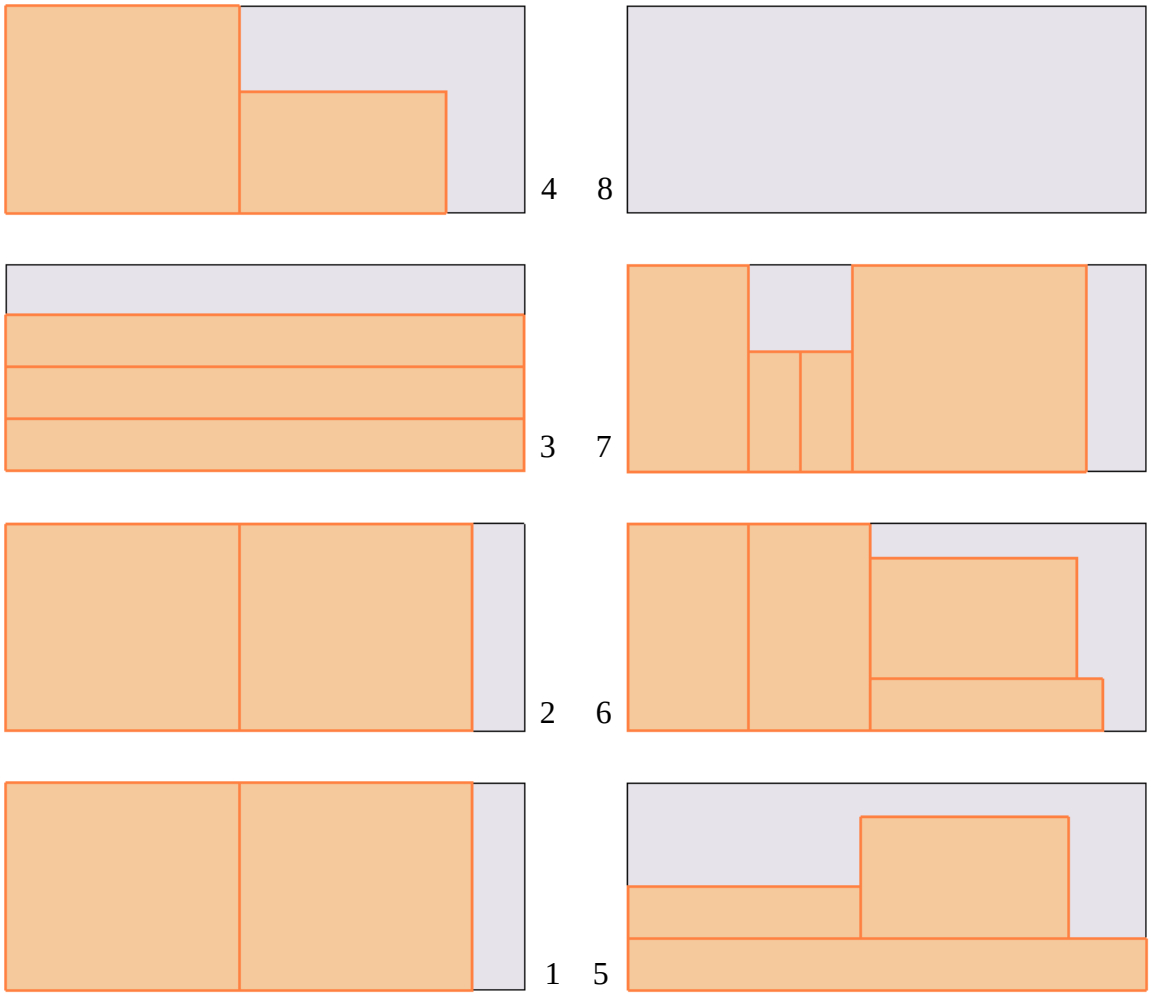


Figure 25: Optimized cutting layouts generated by add-in - 20 iterations

Number of Iterations: 30

Processing Time: 60 min

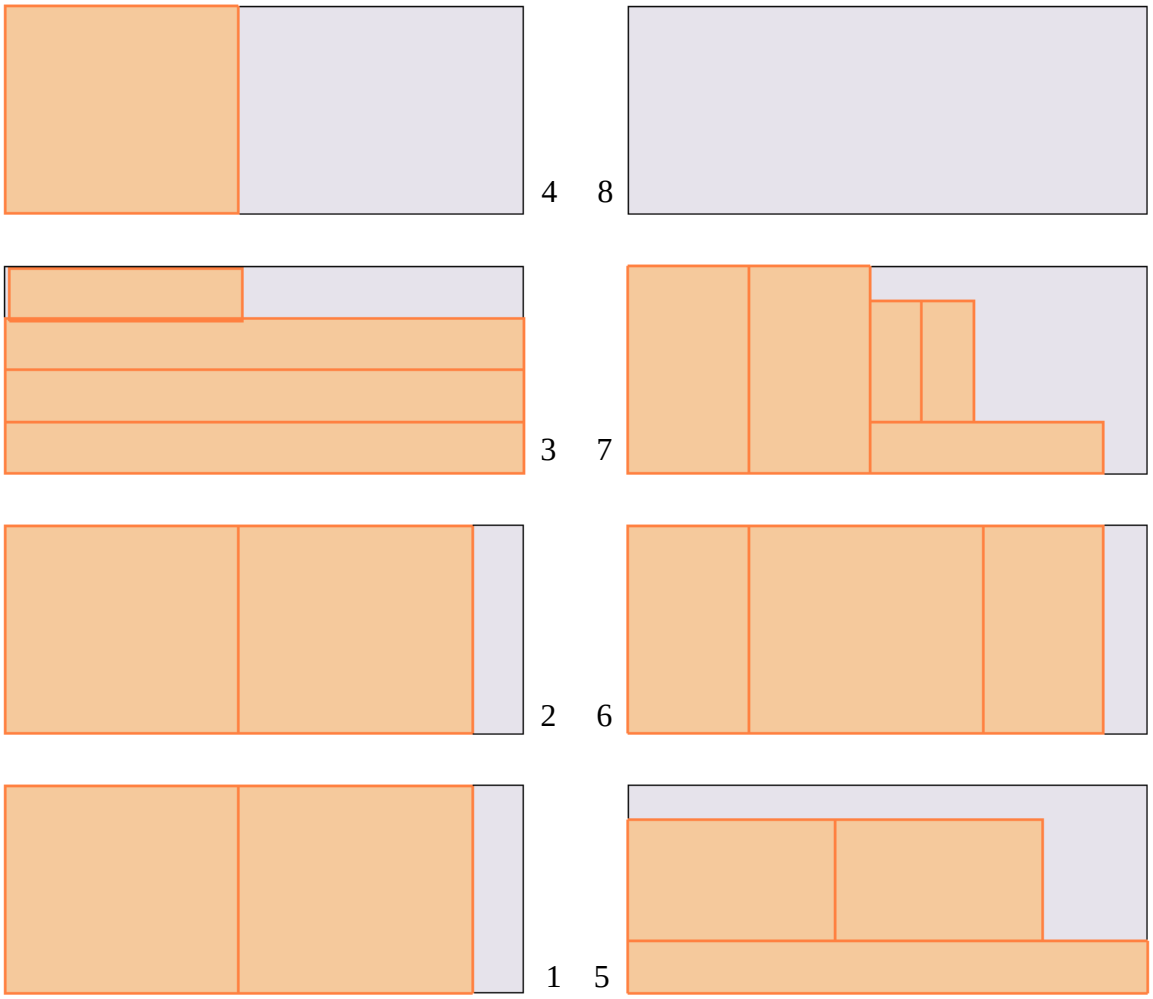


Figure 26: Optimized cutting layouts generated by add-in - 30 iterations

Number of Iterations: 50

Processing Time: 100 min

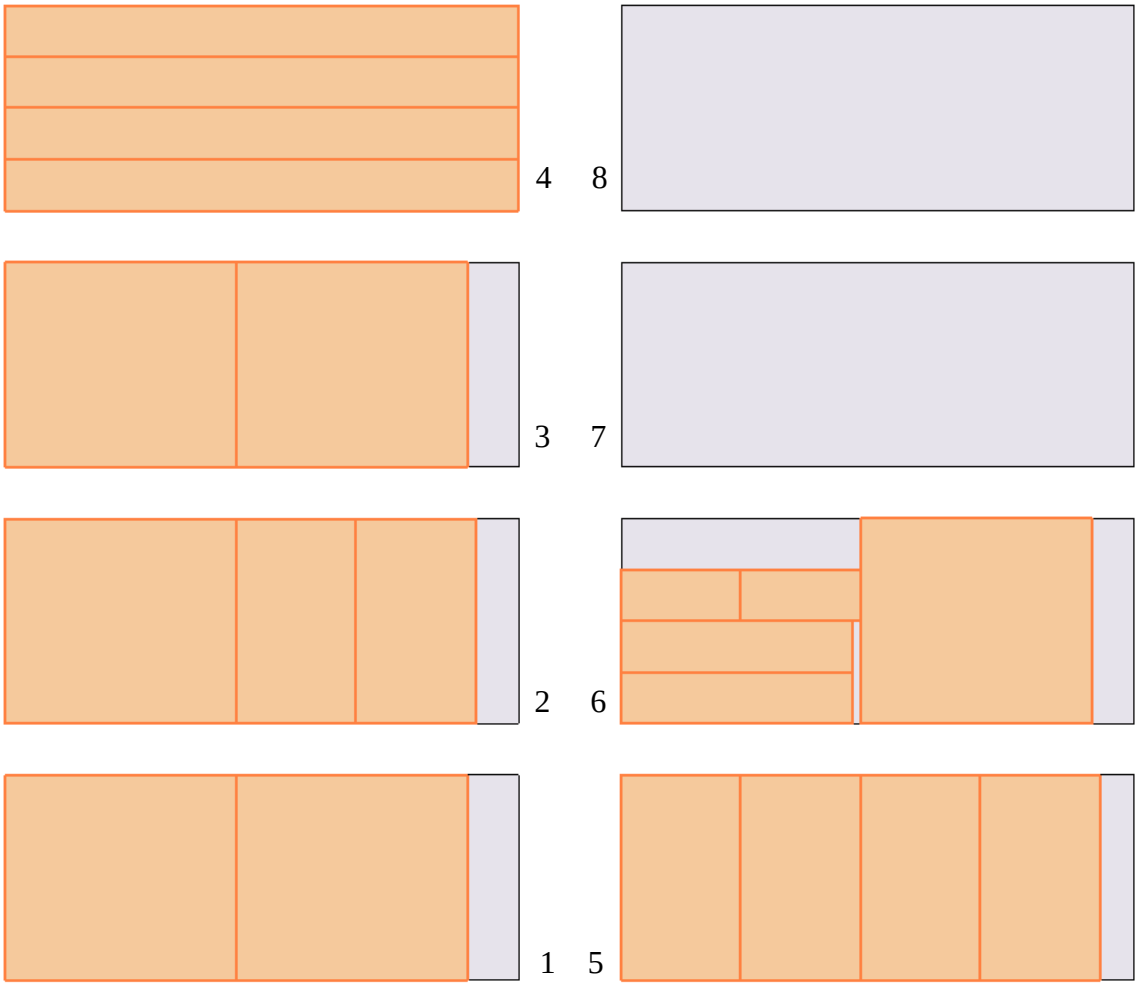


Figure 27: Optimized cutting layouts generated by add-in – 50 iterations

4.4. Analysis of the Second Phase of Testing

In the layouts generated, there is minimum overlaps and there is no vacant spaces between items. All the items are totally inside the bins and the packings are all desirable. It is empirical evidence that the performance of the packing procedure is optimal.

Now, the layouts generated should be compared in regards to the main objective, which is to minimize the leftover. In the layouts generated in the first three rounds (10, 20, 30 iterations), the items are packed into seven bins. Looking at these arrangements, however, it is noticed that they could be improved. For example, bin number 3 in the first round (10 iterations) and bin number 4 in the third round (30 iterations) contain only one item and both have a large vacant space that can be used to fit other items.

In the last round (50 iterations), the sequence is evidently more optimal because the items are packed into 6 bins (less than all the other arrangements). Moreover, no swapping or replacement can be recommended.

In the arrangements generated in the first three rounds and in all the random arrangements, one full panel could have been saved. Therefore, total wastage in those arrangements, is clearly more than 4'×10' = 40 sf of drywall, which means more than $\frac{40}{280} \times 100 \cong 14\ \%$ of stock material is wasted.

Based on calculations, total area of the required items were 218 sf. Therefore, in the last round, $\frac{240-218}{240} \times 100 \cong 9\ \%$ of stock material is wasted. This means that the algorithm could reduce the waste of drywall by about $\frac{14\%-9\%}{14\%} \cong 36\%$

5. CONCLUSION

This study was carried out to investigate if an optimization application coupled with Evolutionary Algorithms can be implemented on a Building Information Modeling platform to generate optimized drywall cutting layouts automatically.

The objectives of this study were: 1) developing an add-in drywall cutting optimization application on a BIM platform using EA-based optimization techniques, and 2) proving if this application can actually suggest a drywall cutting plan that leaves minimum waste.

The add-in application was developed through Autodesk Revit API based on the optimization methods proposed for similar problems in other industries. Then, it was tested on a simple Revit model containing four walls. Tests proved that the proposed algorithm is able to generate optimized cutting layouts. It generated layouts that, compared to the layouts generated in initial iterations has 36% less leftover.

This study proves that it is possible to develop an EA-based optimization application coupled with Building Information Modeling for the purpose of generating drywall cutting layouts automatically. It demonstrates that utilizing such optimization algorithm on a BIM platform should be considered as an effective way to reduce the material waste.

Considering the large number of construction activity in the United States, contractors and consequently owners can save a considerable amount of money utilizing

this method. Also, there is an opportunity to utilize this automated optimization tool to reduce the waste of other building materials.

5.1. Future Research

Considering the scope of this study and the generated results, future research can scrutinize the proposed algorithm. Clearly, this algorithm can be improved in terms of both the generated layouts and processing time.

More tests could be carried out on a larger number of benchmark models to evaluate the performance of the optimization method. It can be tested on different building types – residential and commercial – to compare its efficiency based on the size of the model.

Also, there is an opportunity to customize this tool to apply on other building materials such as façade panels.

REFERENCES

Azhar, S. (2011). Building Information Modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and Management in Engineering*, 11, 241-252.

Blum, C., & Schmid, V. (2013). Solving the 2D bin packing problem by means of a hybrid evolutionary algorithm. *Procedia Computer Science*, 18, 899-908.

Bossink, B. A. G., & Brouwers, H . J. H. (1996). Construction waste: quantification and source evaluation. *Journal of Construction Engineering and Management*, 122, 55-60.

Castelo Branco, C. R. (2007). *An effective way to reduce residential construction waste: a case study in Texas* (Master's Thesis). Texas A&M University, College Station, Texas.

Cheng, C. H., Feiring, B. R., & Cheng, T. C. E. (1994). The cutting stock problem – A survey. *International Journal of Production Economics*, 36, 291-305.

Dasgupta, D., & Michalewicz, Z. (1997). Evolutionary Algorithms in engineering applications. Springer, New York, U. S.

Falk, B., & McKeever, D. (2012). Generation and recovery of solid wood waste in the U.S. *BioCycle*, 30-32.

Feoktistov, V. (2006). Differential Evolution: In search of solutions. Springer, New York, U.S.

Gavilan, R. M., & Bernold, L. E. (1994). Source evaluation of solid waste in building construction. *Journal of Construction Engineering and Management*, 120(3), 536-552.

Koskela, L. (1992). Application of the new production philosophy to construction. *CIFE Technical Report #72*, Stanford University, CA, USA.

Koskela, L. (1997). Lean production in construction, in Alarcon, L. (ed.) Lean Construction, Balkema, Rotterdam, pp. 1–10.

Li, H., Huang, T., Kong, C. W., Guo, H. L., Baldwin, A., Chan, N., & Wong, J. (2008). Integrating design and construction through virtual prototyping. *Automation in Construction*, 17, 915-922.

Manrique, J. D., Al-Hussein, M., Bouferguene, A., & Nasseri, R. (2007). Shop drawing automation and material waste minimization in the construction of wood houses utilizing 3D-CAD and optimization techniques. *Innovations in Structural Engineering and Construction*. London, U.K.: Taylor & Francis.

Manrique, J. D. (2009). *Automation of design and drafting for wood frame structures and construction waste minimization* (Doctoral Dissertation). University of Alberta, Edmonton, Canada.

Manrique, J. D., Al-Hussein, M., Bouferguene, A., Safouhi, H., & Nasseri, R. (2011). Combinatorial algorithm for optimizing wood waste in framing designs. *Journal of Construction Engineering and Management*, 137, 188-197.

McGeorge, D., & Zou, P. (with Palmer, A.). (2012). *Construction management: new directions.* Pondicherry, India: John Wiley & Sons.

McKeever, D. B. (2004). Inventories of woody residues and solid wood waste in the United States, 2002. Retrieved from http://www.fpl.fs.fed.us/documnts/pdf2004/fpl_2004_mckeever002.pdf

National Association of Home Builders, NAHB. (1995). Residential construction waste: from disposal to management. Retrieved from http://www.toolbase.org/PDF/CaseStudies/resi_constr_waste_manage_demo_eval.pdf

Poon, C.S. (2007). Reducing construction waste. *Waste Management*, 27, 1715-1716.

Sacks, R., Koskela, L., Dave, B.A., & Owen, R. (2010). Interaction of Lean and Building Information Modeling in construction. *Journal of Construction Engineering and Management*, 968-980.

Sabahi, P. (2010). *Speeding up the process of modeling temporary structures in a building information model using predefined families* (Master's thesis). Texas A&M University, College Station, Texas.

Sandler, K. (2003). Analyzing what's recyclable in C&D debris. *BioCycle*, 44(11), 51-54.

Shahin, A. A., & Salem, O. (2004). Using genetic algorithms in solving the one-dimensional cutting stock problem in the construction industry. *Canadian Journal of Civil Engineering*, 31(2), 321-332.

Spears, W. (2000). Evolutionary Algorithms: the role of mutation and recombination. Springer, New York, U. S.

Timmerman, M. (2013). *Optimization methods for nesting problems* (Master's Thesis). University West, Trollhattan, Sweden.

Tommelein, I. (2015). Journey toward Lean Construction: pursuing a paradigm shift in the AEC industry. *Journal of Construction and Engineering Management*, 141(6)

U.S. Environmental Protection Agency. (1995). Residential construction waste management demonstration and evaluation. Retrieved from http://www.toolbase.org/PDF/CaseStudies/resi_constr_waste_manage_demo_eval.pdf

U.S. Environmental Protection Agency. (1998). Characterization of building-related construction and demolition debris in the United States. Retrieved from http://www.epa.gov/osw/hazard/generation/sqg/cd-rpt.pdf

U.S. Environmental Protection Agency. (2003). Building-related construction and demolition materials amounts. Retrieved from http://www.epa.gov/osw/conserve/imr/cdm/pubs/cd-meas.pdf

Womack, J. (1999). Manufacturing has moved to Lean - It's time construction does too. *Proceedings 10th Winter Conference in Archistruction*, Neenan, Denver, Colorado, USA.

Wong, L., & Lee, L.S. (2009). Heuristic placement routines for two-dimensional bin packing problem. *Journal of Mathematics and Statistics*, 5(4), 334-341.

Zhang, P., Jiang, J., Xu, H., & Han, X. (2011). Design of multi-dimensional bin-packing heuristic based on genetic algorithm. *International Journal of Computer Science and Artificial Intelligence*, 1(1), 13-17.