Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Economics and Management
Dept. of Industrial Economics and Technology Management

Ingrid Emilie Hermanrud
Carl Fredrik Lystad
Petter Jørgensen Narvhus

A Hybrid Optimization Approach for the School Layout Problem

Combining a Memetic Algorithm, a Mathematical Model, and a Local Search to solve an unparalleled Layout Problem

Master's thesis in Managerial Economics and Operations Research
Supervisor: Henrik Andersson
June 2020

NTNU
Norwegian University of
Science and Technology

Ingrid Emilie Hermanrud
Carl Fredrik Lystad
Petter Jørgensen Narvhus

A Hybrid Optimization Approach for the School Layout Problem

Combining a Memetic Algorithm, a Mathematical Model, and a Local Search to solve an unparalleled Layout Problem

Master's thesis in Managerial Economics and Operations Research
Supervisor: Henrik Andersson
June 2020

Norwegian University of Science and Technology
Faculty of Economics and Management
Dept. of Industrial Economics and Technology Management

Preface

This thesis is written in the course TIØ4905 and concludes our Master of Science degree in Applied Economics and Operations Research at the Department of Industrial Economics and Technology Management at the Norwegian University of Science and Technology (NTNU). This thesis continues the work conducted in the specialization project (TIØ4500) during the fall of 2019.

The thesis is written in collaboration with Spacemaker AI, who has developed an artificial intelligence technology that helps users discover smarter ways to maximize the potential of a building site. The product lets the user generate and explore a multitude of site proposals, sort out the best ones, and provides detailed analyses for each of them. The collaboration comprises the development of school layouts.

We want to thank our supervisor, Prof. Henrik Andersson, for his enthusiasm and valuable input throughout the project. We would also like to express our gratitude to Espen Wold and Simen Braaten at Spacemaker AI for their input on exploiting optimization techniques in the architectural domain, and Nikolai Alfsen at Lille Frøen AS for providing insight into the architectural approach of developing school layouts.

Ingrid E. Hermanrud, Carl F. Lystad and Petter J. Narvhus

Trondheim, June 12, 2020

Abstract

The process of designing a school layout is complex, requiring architectural firms to spend hundreds of hours developing a layout suggestion. The complexity arises from the number of rooms, the diverse composition of rooms, the set of requirements, and both qualitative and quantitative objectives. This motivates the use of optimization techniques to map out the solution space by suggesting layouts with desirable properties. This thesis considers the problem of generating school layouts with low building costs, referred to as the *School Layout Problem* (SLP). The goal is to develop an algorithm that generates school layout designs where building cost, in terms of building area and exterior corners, is minimized.

A comprehensive literature search reveals that there exists no previous research works on using optimization techniques in school layout design. Thus, a review is conducted on comparable problems, mainly packing problems and other layout problems. Based on the findings, a three-stage algorithm is implemented. The algorithm considers the multi-objective optimization problem of allocating rooms and hallways to a building site, forming a single floor in two dimensions. The algorithm consists of a memetic algorithm (MA), a mathematical model, and a local search (LS). The MA consists of a genetic algorithm (GA) and an LS, which jointly generate a first draft of the school layout. Stage two is a mathematical model formulated as a single-objective integer linear program. The model is applied to subareas of the layout, seeking to minimize the number of corners locally. Finally, a local search is employed in stage three, aiming to minimize the number of exterior corners and the total building area.

The SLP takes a *room specification document* (RSD) as input - a document listing all rooms and room requirements for a particular school. These are size, shape, proximity, and natural lighting requirements. The RSD does not specify hallways. Instead, this thesis introduces a sophisticated algorithm for dynamically generating hallways. The three-stage algorithm generates school layouts that satisfy the requirements in the RSD, along with additional constraints such as hallway connections between rooms.

Extensive tests of the algorithm are conducted to assess various implementation alternatives, find suitable parameter settings, and improve the compatibility of the three stages. Six RSDs with different complexity and characteristics are used to test the performance of the algorithm. The RSDs are subsets of the RSD for Levanger Middle School, which was built in 2015. The results are satisfying, as the algorithm generates desirable layouts in terms of cost, while meeting the requirements in the RSD.

This thesis illustrates the suitability of applying optimization techniques in the development process of school layouts. The results show that a multi-stage algorithm which exploits the strengths of several solution methods, has strong potential to serve as decision support for architects when designing school layouts. The algorithm generates a wide range of different, desirable layouts. These can be used both as inspiration and starting points for architects in the planning phase, streamlining the process of developing school layout designs. This study adds to the literature by exploring the absent branch of layout problems that is school layout problems. Additionally, the implemented algorithm extends existing research on layout problems by having to consider qualitative and quantitative objectives of the SLP that differ from comparable research works.

Sammendrag

Å designe planløsningen til en skole er en kompleks oppgave, og krever at arkitektfirmaer bruker hundrevis av timer på å utvikle et planløsningsforslag. Kompleksiteten kommer av antall rom, ulike romstørrelser, spesifikke krav til hvert enkelt rom, samt kvalitative og kvantitative mål. Dette motiverer bruken av optimeringsteknikker til å kartlegge løsningsområdet ved å foreslå ulike planløsninger med ønskede egenskaper. Denne masteroppgaven utforsker problemet med å generere planløsninger til skolebygg som medfører lave byggekostnader, referert til som *Planløsningsproblem for Skoler* (SLP). Målet er å utvikle en algoritme som er i stand til å generere en planløsning der byggekostnadene, med tanke på bygningsareal og utvendige hjørner, minimeres.

Et omfattende litteratursøk viser at det ikke eksisterer tidligere forskning på bruk av optimeringsteknikker for å utforme planløsninger i skolebygg. Dermed gjennomføres et studie på sammenlignbare problemer, hovedsakelig pakkeproblemer og andre planløsningsproblemer. Basert på funnene implementeres en trestegsalgoritme. Algoritmen tar for seg det multi-objektive optimeringsproblemet å fordele rom og ganger på en tomt, som dermed danner en planløsning for en etasje i to dimensjoner. Algoritmen består av en memetisk algoritme (MA), en matematisk modell og et lokalsøk (LS). MA består av en genetisk algoritme (GA) og et LS, som sammen genererer et første utkast til en planløsning. Steg to er en matematisk modell som er formulert som et singel-objektiv linært heltallsprogram. Modellen tar for seg delområder i planløsningen, og forsøker å minimere antall hjørner lokalt i disse områdene. Til slutt benyttes et lokalsøk i steg tre med sikte på å minimere antall utvendige hjørner og det totale bygningsarealet.

SLP tar et *romprogram* (RSD) som input - et dokument som lister alle rom og romkrav for en bestemt skole. Kravene omhandler størrelse, form, nærhet og naturlig belysning. Korridorer er ikke spesifisert i et RSD. I stedet implementeres en sofistikert algoritme for dynamisk generering av korridorer. Trestegsalgoritmen genererer planløsninger som tilfredsstiller kravene i et RSD, samt andre krav som korridorforbindelser mellom rom.

Omfattende tester av algoritmen er gjennomført for å vurdere ulike implementeringsalternativer, finne passende parameterinnstillinger og forbedre kompatibiliteten mellom de tre stegene. Ytelsen til algoritmen testes på seks RSDer av ulik kompleksitet og med forskjellige egenskaper. RSDene er delmengder av romprogrammet til

Levanger ungdomsskole som ble bygget i 2015. Resultatene er tilfredsstillende, da algoritmen er i stand til å generere planløsninger som medfører lave byggekostnader, samtidig som de oppfyller kravene i RSDene.

Denne masteroppgaven illustrerer hvordan optimeringsteknikker kan brukes i utviklingen av planløsninger for skoler. Resultatene viser at en flertrinns algoritme som utnytter styrkene til flere løsningsmetoder har et sterkt potensiale til å fungere som beslutningsstøtte for arkitekter når de utformer planløsningen til en skole. Algoritmen genererer et bredt spekter av forskjellige, ønskelige planløsninger. Disse kan brukes både som inspirasjon og utgangspunkt for arkitekter i planleggingsfasen, og effektivisere prosessen med å utvikle planløsninger. Denne studien beriker litteraturen ved å utforske planløsningsproblemer for skoler, som er en fraværende gren i studier av planløsningsproblemer. I tillegg utvider den implementerte algoritmen eksisterende forskning på planløsningsproblemer ved å vurdere kvalitative og kvantitative mål for SLP som skiller seg fra sammenlignbare studier.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

According to the Norwegian Education Act of 2003, all children have a statutory right to a positive physical school environment. To uphold this, the 3500 school buildings in Norway (Udir 2019) are subject to strict and constantly renewed regulations and recommendations by the Norwegian authorities. A well-designed school contributes to a positive learning environment for the students and an enjoyable working environment for teachers and other employees (Schanke 2008). Additionally, schools act as a cultural arena for the local community, and must be designed to serve this purpose as well. It is the responsibility of each municipality to ensure sufficient capacity and quality of its educational services. While the numbers greatly vary, the average cost of building a new school amounts to $330,000$ NOK per student, according to Statistics Norway (SSB). Clearly, a school building is a significant investment for a municipality.

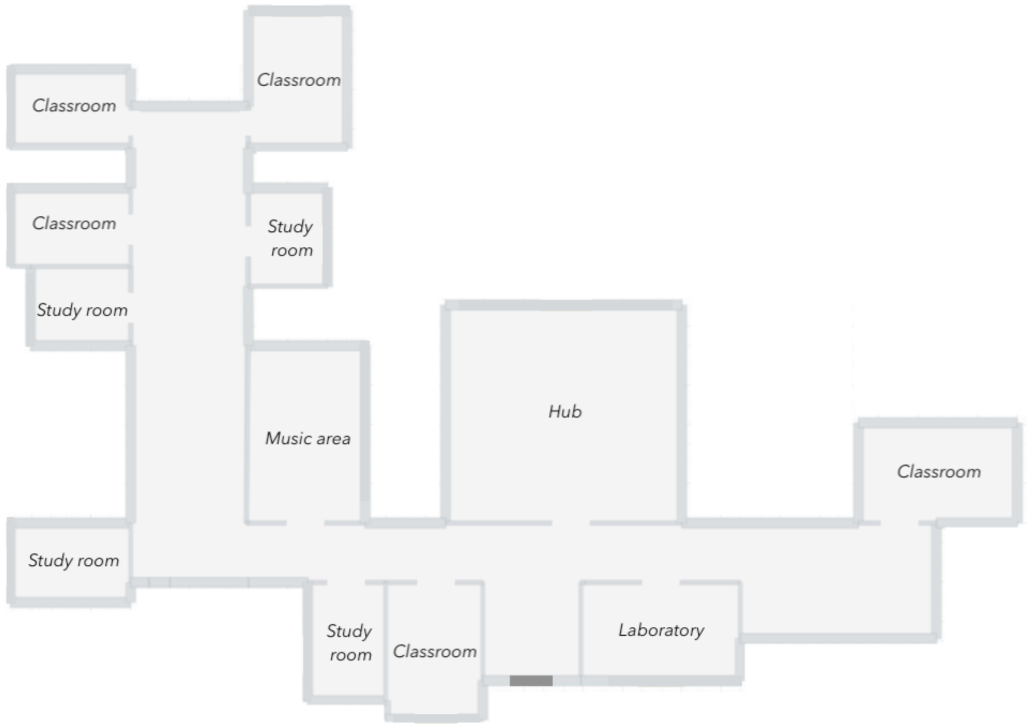
When a municipality decides to build a new school, it must first choose a site and determine specifications such as the number of students and the necessary facilities. Municipalities write a detailed description of requirements in a document called the *room specification document* (RSD). The RSD consists of a list of rooms, typically over a hundred, along with general guidelines for the complete school design. When the specifications are determined, the municipality initiates a bidding process for architectural firms. A bid is a complete design proposal for the school, meaning the exterior and interior of the buildings, as well as the outdoor area.

To create a successful bid, there are a great number of rules, regulations, and objectives an architect must consider. From the architect's point of view, the design of a school can be divided into three components; determining the location of the building on the site, creating a layout for the school, and forming the outdoor area. Arguably, the most demanding design component is the school layout. The architect must adhere to the requirements in the RSD, while the design must satisfy national and regional regulations, e.g., emergency regulations, universal design, and flow capacity in hallways. There are also several important objectives, which are both qualitative, e.g., usability and aesthetics, and quantitative, e.g., building cost. As school buildings are a large investment for the municipality, the cost is often the most decisive criterion in the bidding process. The number of requirements and objectives, in addition to the conflict of interest between them, make developing a complete school design a strenuous task. To exemplify, a layout with large hallways will satisfy flow capacity requirements, but extra square meters add to the building cost. The process of designing a school layout is a multi-objective task that requires balancing conflicting goals and complying with complex regulations.

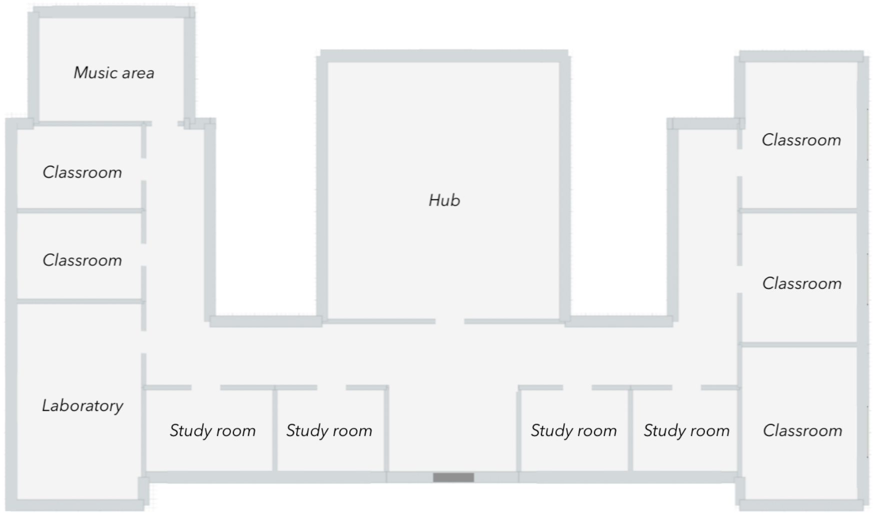
In an RSD, each room comes with a function description, a suggested size, and sometimes an *aspect ratio bound*. The rooms are usually listed in groups, further referred to as *neighbourhoods*. For example, in a middle schools' RSD, a neighbourhood can be classrooms, study rooms and common rooms used by the eight grade students. These neighbourhoods determine proximity requirements, as closeness between rooms within a neighbourhood is desirable. Table 1.1 shows a simplified example of a list of rooms from an RSD. The table displays the name of the rooms, their suggested sizes and aspect ratio bounds. The last column specifies how many of these rooms the school should have. Figure 1.1 illustrates two very different layouts that both follow the specifications in Table 1.1. Figure 1.1 (a) is clearly an inefficient layout as it fails to consider many qualitative and quantitative objectives. Figure 1.1 (b) shows one of many viable layouts that can follow from the simplified RSD, and has a simple geometric shape.

Table 1.1: Simplified RSD example

Room	**Suggested size** (m^2)	**Aspect ratio bound**	#
Classroom	50	1.5	5
Study room	40	2	4
Music area	70	1.5	1
Biological laboratory	80	2.5	1
Hub	300	3	1



(a) Inefficient layout



(b) Efficient layout

Figure 1.1: Two possible layouts that adheres to the specifications in the simplified RSD from Table 1.1. (*a*) is clearly an inefficient layout, whereas (*b*) is a more promising layout.

Today, the development of a bid is predominantly a manual process. An architect starts by looking at key information in the RSD, such as the type of school and the number of students, and performs a site inspection. Architects then rely on traditional methods using their knowledge and experience to form ideas of design solutions. Hence, the process of developing the initial layout design is a resource-demanding task. Discussions with the experienced architect firm Lille Frøen reveal that it often takes 800-900 hours to create a complete bid, and that they spend the majority of this time developing the layout.

Architect firms widely use software tools for visualization and analysis of layouts. A typical analysis is total square meters of usable area or a room's lighting condition. However, the tools require the architect to input a suggested layout, and then perform the analysis. If the layout does not meet regulations or requirements, the architect must manually assess changes to improve the solution, alter the layout and reperform the analysis. This process is repeated until a desirable, feasible design is obtained. Conflicts of interest between requirements and objectives make it likely that fixing one problem creates another. Besides, it is not evident whether or not a design meets the necessary regulations. It is difficult for an architect to keep track of all these challenges simultaneously, and iteratively changing the layout is a tedious and resource-demanding task. Another problem is that the first layout suggestion a firm comes up with often has a large impact on the final design. Since creating an initial layout proposal is time-consuming, subsequent design work tends to lean on the first draft. Consequently, different, potentially more desirable design solutions can be overlooked.

Modelling the problem as an optimization problem enables generating multiple solutions that are guaranteed to meet quantitative regulations, while optimizing for desirable objectives. Using optimization methods can allow the architect to work with feasible solutions and exposes her/him to many possible designs. Historically, optimization models have been used for comparable layout problems. Research has mainly focused on industrial applications and housing, while the research on applying optimization methods to generate school layouts is non-existent.

This thesis seeks to introduce an optimization algorithm that architects can use as a decision support tool when developing school layouts. The algorithm creates layout suggestions which comply with the requirements in the RSD while minimizing building cost. The task of developing layouts consistent with the RSD is referred to as the *School Layout Problem* (SLP). This problem was first introduced in Hermanrud et al. (2019). The SLP is defined in two dimensions and considers the development of a single floor layout with no preset building footprint. Hermanrud et al. (2019) conducted a feasibility study assessing the suitability of applying optimization methods to the SLP. The study showed promising results for using optimization methods as a support tool when solving the SLP. Hermanrud et al. (2019) developed a basic optimization algorithm which, together with the findings, lay the foundation for the work in this thesis. The solution method presented in Hermanrud et al. (2019) was a memetic algorithm (MA). This thesis extends the algorithm to a three-stage algorithm. The first stage is an improved version of the MA from Hermanrud et al. (2019), which chooses the topology of the neighbourhoods and allocates the rooms. The second stage is a mathematical model which takes the layout generated by the first stage as input and improves the neighbourhoods separately in terms of corners. The final stage is a local search (LS) using the gradient descent technique to decrease building costs by minimizing the building area and the number of exterior corners.

Chapter 2 provides insight into relevant literature, and discusses the transferability of reviewed approaches to the SLP. Next, Chapter 3 gives a thorough explanation of the school layout problem. Chapter 4 provides an overview of the three stages in the solution method, as well as a discussion on why they, together, are suited to solve the SLP. Chapters 5 - 7 give a detailed explanation of how the different stages are implemented. The various instances used to test the three-stage algorithm are presented in Chapter 8. To optimize performance, each stage is tested individually in Chapter 9, Chapter 10 and Chapter 11. After completing these tests, a performance study of the full solution method is presented in Chapter 12. Finally, Chapter 13 presents the concluding remarks, and outlines possible future research areas and extensions to the problem considered in this thesis.

Chapter 2

Literature review

This chapter intends to provide insight into relevant literature to the school layout problem (SLP). To the best of our knowledge, there exists no published studies on optimal layouts for schools. Thus, problems similar to the SLP are examined. The SLP can partly be classified as a layout problem and partly as a packing problem. Layout problems concern the allocation of space elements on a given area with regard to a set of constraints and objectives. Similarly, the SLP considers allocating rooms and hallways to a building site. The constraints in layout problems often concern interrelations between objects, which is highly relevant to the SLP. Packing problems attempt to pack small objects into one or several large objects to minimize the unused area. This relates to the SLP as the rooms (small objects) are packed densely, and the excess amount of hallways (unused area) is minimized within the layout (large object). Some packing problems do not have a fixed large object, but instead attempt to minimize the packing area. In the SLP, the footprint of the school building is not predefined. Thus, these types of packing problems are relatable.

In this review, the search engine ”Scopus” is used. For each problem examined, a set of relevant keywords are determined to search for literature. In the beginning of each section, a more detailed description is given on how the literature search is conducted. In Section 2.1, different packing problems are explored and compared to the SLP. Section 2.2 examines layout problems of similar nature to the SLP, mainly focusing on facility layout problems (FLP) and architectural space planning (ASP). Lastly, Section 2.3 discusses this thesis's contribution to the literature.

2.1 Packing problems

Packing problems are a class of optimization problems that attempt to pack smaller objects into one or several larger objects. The objects are polygons defined in multiple geometric dimensions. Certain geometric conditions must hold for the assignment of objects - the small objects have to lie entirely within the large object, and they cannot overlap (Wäscher et al. 2007). The goal is to either pack a single large object as densely as possible or pack all small objects using as few large objects as possible (Di Pieri 2013).

As demonstrated in the typology by Dyckhoff (1990) and later Wäscher et al. (2007) there exists a large variation of packing problems. Wäscher et al. (2007) introduce categorisation criteria to define problem types. These include dimensionality, type of assignment, the assortment of small and large items, and the shape of small items. Based on these criteria, six basic problem types are defined; identical item packing problem, placement problem, knapsack problem, open dimension problem, cutting stock problem and bin packing problem. The SLP is considered an extension of the open dimension problem (ODP). In ODPs, one or several dimensions of the large object are considered decision variables. Additionally, the small objects are rectangles and have fixed dimensions and area. There are clear similarities between the traditional ODP and the SLP. In the SLP, the small objects (rooms) are rectangular, while the large object (school) has variable dimensions and is rectilinear, as it consists of a set of axis-aligned, rectangular rooms. Additionally, the geometric conditions are the same, as rooms cannot overlap.

Besides the similarities, the SLP contains many aspects not present in ODPs. Rooms in the SLP can have variable dimensions as long as their aspect ratio is within a given bound. These are often referred to as *soft rectangles* in literature. The area of the rooms can also vary within a given range, which is uncommon for soft rectangle packing (SRP) and the ODP. Additionally, as the SLP does not consider a set building footprint, the small objects are not packed into a predefined rectilinear object. Still, the final footprint will be of rectilinear shape. Because of this property, the SLP has some resemblance to minimal area packing (MAP). MAP problems intend to minimize the packing area of small objects. The packing area is required to be rectangular, and thus the objective is to minimize the rectangular *envelope* covering all the small objects. The SLP is in some way a rectilinear MAP problem where the packing area is not necessarily rectangular, but rather rectilinear. Other aspects that differentiate the SLP from MAP and ODPs is the interrelation between rooms and the location of hallways. This is rarely present, or non-existing, in packing problems.

2.1.1 Material collection

To systematically explore the literature that examine packing problems with similarities to the SLP, two groups of keywords are developed. Packing problems appear under various names in the literature. Thus, the keywords are determined based on the typologies by Dyckhoff (1990) and Wäscher et al. (2007). The first group contains words equivalent to ”packing” in the literature on packing problems. These words are ”packing”, ”loading”, ”placement” and ”allocation”. The second group consists of words that limit the search to packing problems relevant to the SLP. These are: ”rectangular”, ”polygon”, ”soft rectangle”, ”open dimension”, ”variable dimension”, ”discrete”, ”regular” and ”two-dimensional”.

For a paper to be considered further in the literature review, one of the words from the first group must be part of the title, while at least one word from the second group must appear in either the title, abstract or keywords. This search resulted in 2248 papers. From these, only English journal articles are considered, resulting in 1272 papers. Further, only literature regarding the subject areas mathematics, computer science, engineering and decision sciences are included. By limiting the search to these research fields, 643 papers remain. To further narrow down the search, papers with irrelevant keywords are excluded. These keywords are, for instance, ”particles”, ”molecular dynamics” and ”friction”, many from the fields of chemistry and biology. Considering these criteria, 98 papers remain. The abstracts of these papers are read to determine their relevance, and papers regarding traditional bin- and strip- packing problems are excluded. Some of these papers are read for modelling inspiration, such as how to model overlap constraints. Reviewing the abstracts left nine relevant papers. Lastly, publications citing and cited by the nine papers are reviewed, resulting in three additional relevant papers. Thus, 12 papers are left for full review. Figure 2.1 illustrates the material collection process.

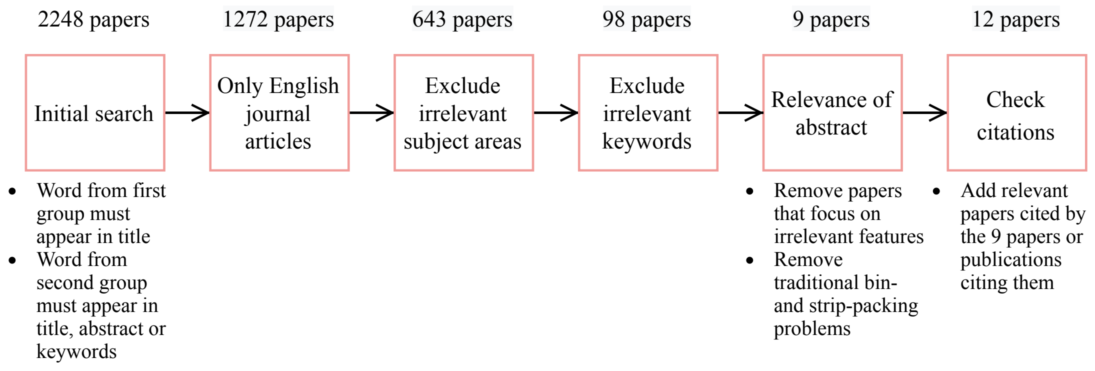

2248 papers
1272 papers
643 papers
98 papers
9 papers
12 papers
Initial search
Only English journal articles
Exclude irrelevant subject areas
Exclude irrelevant keywords
Relevance of abstract
Check citations
Word from first group must appear in title
Word from second group must appear in title, abstract or keywords
Remove papers that focus on irrelevant features
Remove traditional bin- and strip-packing problems
Add relevant papers cited by the 9 papers or publications citing them

Figure 2.1: Material collection process on packing problems similar to the SLP

2.1.2 Full review

Table 2.1 presents the papers left for full review. The numbers in the first column refer to the papers in Table 2.2. The additional columns compare relevant characteristics of the SLP to the various packing problems reviewed. The properties of the SLP is specified in the bottom line of the table for comparison.

In the columns defining the shape of the small and large item(s), the terms *irregular* and *rectilinear* are used. An irregular polygon can have sides of any length, and each interior angle can be any measure. A rectilinear polygon is made up of straight lines, where all sides meet at right angles. In all articles reviewed, the rectilinear objects are arranged in an axis-aligned way, which is also the case in the SLP.

The column ”Problem type” defines the type of packing problem considered in the paper. In this column, Soft MAP and 2D-OP are abbreviations for soft minimal area packing and two-dimensional orthogonal packing, respectively. Soft MAP refers to the packing of soft rectangles, and is an extension of the traditional MAP. 2D-

OP involves determining whether a set of rectangles can be allocated into a single rectangle of fixed size. Soft 2D-OP is a 2D-OP problem with soft rectangles. The last column specifies the solution method presented in the paper. The abbreviations of this column are explained in Table 2.3. Note that a genetic algorithm (GA) hybridized with a local search (LS) procedure is called a memetic algorithm (MA). A heuristic is a technique for finding an approximate solution to a problem. Heuristics are helpful when classic methods fail to find any exact solution. The solution methods of papers using less known heuristics, as opposed to GA, MA, SA and LS, are referred to as heuristic in the solution method column.

Table 2.1: Comparison between the SLP and the papers left for full review. *The area can vary as well as the dimensions.

Instance	**Shape of large object(s)**	**Shape of small object(s)**	**Variable large object dimensions**	**Variable small object dimensions**	**Problem type**	**Solution method**
1	Rectangular	Irregular	Yes	No	MAP	GA
2	Rectangular	Rectangular	Yes	Yes	Soft MAP	LP + SA
3	Rectangular	Rectangular	Yes	Yes	Soft MAP	Lagrange
4	Rectangular	Rectilinear	Yes	Yes	Soft MAP	Lagrange
5	Rectangular	Rectangular	Yes	Yes*	Soft MAP	LS + LP
6	Rectangular	Rectangular	Yes	No	MAP	NLP
7	Cuboid	Cuboid	Yes	No	MAP	LP
8	Rectilinear	Rectangular	Yes	Yes	Soft MAP	LP
9	Rectangular	Rectangular	Yes	No	MAP	Heuristic
10	Rectangular	Rectangular	No	Yes	Soft 2D-OP	Heuristic
11	Square	Rectangular	Yes	Yes	Soft MAP	-
12	Rectilinear	Rectangular	Yes	No	MAP	GA
SLP	Rectilinear	Rectangular	Yes	Yes*	Rectilinear MAP	MA + LP + LS

Table 2.2: Paper overview for Table 2.1

1	Jain and Gea (1998)
2	Murata and Kuh (1998)
3	F. Young et al. (2001)
4	Chu and E. Young (2004)
5	Ibaraki and Nakamura (2006)
6	Maag et al. (2010)
7	Hu et al. (2012)
8	Fügenschuh et al. (2014)
9	He et al. (2015)
10	Ji et al. (2017)
11	Brenner (2018)
12	Erozan and Çalışkan (2020)

Shape of small and large object(s)

The second and third column of Table 2.1 specify the shape of the large and small object(s) of each problem. In the SLP, the large object (school) is rectilinear, and the small objects (rooms) are rectangular. Erozan and Çalışkan (2020) and Fügenschuh et al. (2014) consider problems which allow for rectilinear large objects, like in the

Table 2.3: Abbrevations for solution method column in Table 2.1

MA	Memetic algorithm
GA	Genetic algorithm
SA	Simulated annealing
LS	Local search
LP	Linear program solved using a commercial solver
NLP	Non-linear program solved using a commercial solver

SLP. The model by Erozan and Çalışkan (2020) includes no predefined large object. In contrast to traditional MAPs, which minimizes the envelope covering the small objects, Erozan and Çalışkan (2020) minimize the distances between the centres of the rectangular items using a genetic algorithm. Thus, the problem has a resemblance to rectilinear MAPs, as the packing area is not required to be rectangular. Fügenschuh et al. (2014) present a model to solve an extended bi-objective MAP problem with rectangular small objects and a final rectilinear layout. In addition to minimizing the envelope covering the small objects, the objective minimizes the length of all inner borderlines. This length is proportional to the amount of ink one would use when drawing the list of rectangles. The second objective facilitate rectilinear layouts as much as rectangular ones. Thus, by removing the envelope objective, the problem becomes more similar to the SLP.

Four papers suggest different solution methods to the traditional MAP problem with rectangular small and large objects; Ibaraki and Nakamura (2006), Maag et al. (2010), F. Young et al. (2001), and He et al. (2015). Even though these papers deal with rectangular shaped large objects, they present approaches on how to densely pack rectangles, which is an element of the SLP. Ibaraki and Nakamura (2006) propose a hybrid of local search and mathematical programming to solve the MAP problem. Using local search to find relative positions of the rectangles, the final mathematical model determines the exact locations of rectangles. The resulting algorithm solves problem instances with up to 50 rectangles. This complexity, at least in terms of the number of rooms, resembles the SLP. Maag et al. (2010) and F. Young et al. (2001) present different mathematical models to solve the MAP. F. Young et al. (2001) define an integer program (IP) which is approximated with Lagrangian relaxation, while Maag et al. (2010) model a non-linear program (NLP). Lastly, He et al. (2015) present a dynamic reduction algorithm by solving the MAP problem as a series of 2D-OP problems.

Chu and E. Young (2004) propose a mathematical model which allows for rectilinear small objects. Initially, they solve a soft MAP problem. Then the unused area of the envelope is divided among the rectangles. Thus, if a rectangle is assigned parts of the unused area, its entirety may contain a subset of rectangles, and can, therefore, be of rectilinear shape. The problem solved by Ji et al. (2017) has resemblance to the one presented in Chu and E. Young (2004). They start by solving a soft 2D-OP problem, and thereafter assign unused area to the small objects. The small objects must maintain a rectangular shape after this assignment. Jain and Gea (1998) solve a

MAP problem in which both rectilinear and irregular shapes of the small objects are allowed. The authors use a grid representation where each small object is discretized into a finite number of equisized cells. This allows for irregular shapes as the objects are just a bunch of cells lying next to each other with their relative position defined by a linked list. Also, Hermanrud et al. (2019) exploit a grid representation to model the phenotype of the school layout.

Variable dimensions of small objects

One characteristic that distinguishes the SLP from the reviewed literature is the variable dimensions and area of the small objects. Both the dimensions and area of a room can vary within a given range. The fifth column of Table 2.2 shows that only one paper has both these features in common with the SLP. Ibaraki and Nakamura (2006) include boundaries on the length and width of a rectangle as well as the aspect ratio. In this way, both the dimensions and the area of a rectangle can vary within a given range. Along with this similarity, the paper presents constraints relevant to the SLP, such as how to lock rooms within a particular region and how to avoid overlap.

Six additional papers in Table 2.2 consider soft rectangle packing problems. These problems contain small objects with given areas while their aspect ratios are chosen from a given interval. Hence, the dimensions of the small objects can vary. Ji et al. (2017) propose an iterative merging algorithm to solve a soft 2D-OP problem. The algorithm iteratively merges the two rectangles with the smallest areas, and places the merged rectangle into the large object by recursively determining its position and dimensions. Murata and Kuh (1998) present a *sequence-pair* based placement method for soft rectangles. The sequence-pairs specify the placement topology of objects. The placement method fixes the horizontal and vertical relationships among rectangles. Consequently, avoiding overlap. The solution method is a hybrid of simulated annealing (SA) and mathematical programming, intending to minimize the envelope under a placement topology. Besides the similarity of having variable dimensions, these two papers as well as the four papers by Brenner (2018), Fügenschuh et al. (2014), F. Young et al. (2001), and Chu and E. Young (2004) differ from the SLP as the areas of the rooms are fixed.

Interrelations between objects

An essential consideration of the SLP is the interrelations between objects, which distinguishes it from traditional packing problems. Certain rooms are required to lie next to each other, and the distances within each neighbourhood are subject to minimization. Interrelations is often part of layout problems, but is not commonly present in packing problems. Erozan and Çalışkan (2020) use GA to solve an extended MAP problem and consider the location of related objects. Some rectangular items have a *defensive radius*, in which only certain other rectangles can be placed. The main purpose of the radius is to place rectangular items that are related close to one other. F. Young et al. (2001) consider the packing of floorplans and use constraint graphs to represent horizontal and vertical placement relationships between rooms. The graphs are given as input, such that the relative positions of rooms are predetermined. The placement restrictions in the SLP are stricter, as some rooms are required to lie next to each other instead of only constraining their relative positions. By tightening some of the constraints in the model by F. Young et al. (2001), their problem becomes more similar to the SLP.

2.2 Layout problems

Layout problems are concerned with the allocation of space elements in a given area, with regards to a set of constraints and objectives. The term layout problems cover a vast number of applications ranging from placing transistors on a chip, known as very-large-scale integration (VLSI) design, to residential building design. These problems are also subject to several geometric requirements, e.g. the space elements cannot overlap, and they must be placed within the given area. The objectives in a layout problem varies across the different applications, but usually consider the relationships and interaction between the space elements.

A familiar layout problem studied for the past 60 years is the facility layout problem (FLP). The FLP is defined as the placement of facilities in a plant area, with the aim of determining the most effective arrangement in accordance with some objectives and constraints (Hosseini nasab et al. 2018). Typically are these constraints the shape, size and orientation of the facilities. The general objectives of FLPs are to allocate the facilities to maximize throughput, productivity and efficiency.

The SLP and FLP share the concerns of a typical layout problem, i.e. placing space elements in feasible locations, having non-overlapping constraints and geometric restrictions. On the other hand, the objectives when developing an efficient layout in an industrial environment differ from the objectives of a beneficial educational facility. They do, however, share some of the core objectives of layout problems such as proximity and accessibility between the allocated space elements. An important feature that FLPs share with SLPs is the high complexity. The size of the problem instances, as well as the variety of facility sizes and shapes, correspond well with the variety and number of rooms in the SLP. Hence, solution approaches that can handle the complexity of FLPs are likely to do so also on SLPs.

Architectural space planning (ASP) is defined by Dutta and Sarthak (2011) as finding feasible locations for a set of interrelated objects. ASPs usually consider residential building layouts. The layout should not only meet design requirements and preferences, but also satisfy aesthetics and usability. Naturally, ASP shares the aforementioned characteristics of a canonical layout problem. Furthermore, ASP includes a strong focus on subjective criteria such as aesthetics and design preferences, which are essential aspects of the SLP. Thus, the representation of the problem and the solution approach for ASP problems must facilitate the possibility of including subjective and preferential objectives. A solution approach performing well on ASP problems is likely to capture these aspects also in the SLP. Most of the studies on ASP focus on private housing, such as apartments. The problem instances usually consist of fewer and more homogeneous space elements than SLPs. Therefore, it is crucial to assess the suggested solution methods' ability to scale, as the increased complexity of modelling a school layout might make them inappropriate.

2.2.1 Material collection

To efficiently explore the literature on layout problems, a similar search strategy as for the bin packing problems is conducted, where two groups of keywords are developed. The first group contains words equivalent to "layout". These words are "layout", "floor plan", "space planning" and "spatial allocation". The second group contains words to limit the search to layout problems relevant to the SLP. These words are: "rectangular", "rectilinear", "building", "variable dimension" and "soft rectangle".

For a paper to be considered further in the literature review, one of the words from the first group must be part of the title, while at least one words from the second group must appear in either the title, abstract or keywords. This search resulted in 2180 papers. Limiting the language to English and the type of paper to journal decreases the results to 1009 papers. Next, the search is limited to the subject areas Engineering, Computer Science, Mathematics and Decision Sciences, which yields 634 papers. To further constrict the literature search, papers containing irrelevant keywords are excluded. These keywords are, for instance, "VLSI", "Integrated Circuits" and "Fluid Dynamics". 189 papers remain when these keywords are excluded. To further narrow down the search, all papers dated before 1995 are excluded, which results in 121 remaining papers. The titles and if necessary, the abstract of the remaining papers are read to determine their relevance. Several papers considering basic layout problems such as quadratic assignment problems (QAP) are then excluded. QAP are problems where the site is partitioned into equally sized segments where the number of segments equals the number of space elements to allocate. The problem is then to assign each space element to a segment. Formulating the SLP as a QAP is impracticable, and QAPs are therefore considered irrelevant. Additionally, papers considering domains incomparable to SLP are removed, e.g. sewer system design. Finally, 14 papers remain.

Additionally, the literature search discovered three comprehensive surveys on layout problems. Drira et al. (2007) and Hosseini nasab et al. (2018) conducts a survey on a magnitude of extensions of FLP and Dutta and Sarthak (2011) conducts a review on using evolutionary solution approaches on ASP. From the papers discussed in the surveys, an additional nine papers are considered interesting for further review. Consequently, 25 papers are chosen for full review. Figure 2.2 illustrates the material collection process.

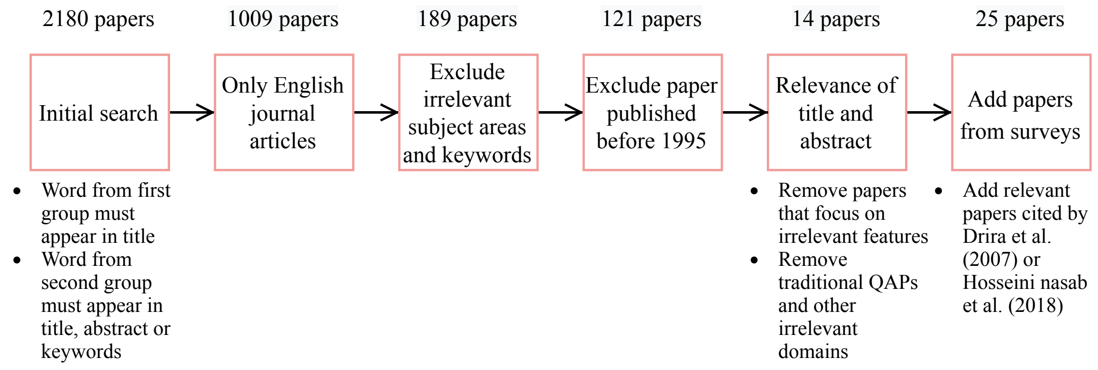

2180 papers
Initial search
Word from first group must appear in title
Word from second group must appear in title, abstract or keywords
1009 papers
Only English journal articles
189 papers
Exclude irrelevant subject areas and keywords
121 papers
Exclude paper published before 1995
14 papers
Relevance of title and abstract
Remove papers that focus on irrelevant features
Remove traditional QAPs and other irrelevant domains
25 papers
Add papers from surveys
Add relevant papers cited by Drira et al. (2007) or Hosseini nasab et al. (2018)

Figure 2.2: Material collection process on layout problems similar to the SLP

2.2.2 Full review

Table 2.4 presents the papers left for full review. The numbers in the first column refer to the papers in Table 2.5. The additional columns compare relevant characteristics of the SLP to the various layout problems reviewed. The properties of the SLP is specified in the bottom line of the table for comparison.

The column ”Shape of space elements” specifies the geometrical class of the space objects, e.g. rooms or facilities, which are to be allocated. Next, The column ”Fixed space elements” states whether or not the objects are subject to changes in shape, dimension and size. Similarly, the column ”fixed footprint” states whether or not the resulting footprint of the allocated elements is predetermined or not. Essentially, if the footprint is fixed, the space elements must completely cover the area they are allowed to be placed. If the problem does not define a fixed footprint, the footprint is generated when the problem is solved, as the area covered by the allocated space elements. The abbreviations used in the column ”Solution method” are explained in Table 2.6.

Table 2.4: Comparison between the SLP and the papers left for full review

Article	**Shape of space elements**	**Fixed space elements**	**Fixed footprint**	**Problem type**	**Solution method**
1	Rectangular	No	No	ASP	SA
2	Rectangular	No	Yes	FLP	B&B
3	Rectangular	Yes	No	FLP	GA + TS/SA
4	Rectilinear	No	Yes	ASP	GA + GDS
5	Rectangular	Yes	Yes	FLP	SA
6	Rectangular	Yes	No	FLP	B&B
7	Rectilinear	No	Yes	ASP	GA
8	Rectangular	No	Yes	FLP	ACO
9	Rectilinear	No	Yes	ASP	GA
10	Rectangular	No	No	FLP	LP
11	Rectilinear	Both	Yes	ASP	EAs
12	Rectangular	Yes	Yes	Interior Design	NSGA-II
13	Rectilinear	No	No	ASP	DFS + NN
14	Irregular	No	N/A	Construction Site	GA
15	Rectilinear	No	Yes	ASP	EA
16	Rectangular	Yes	N/A	Construction Site	PSO
17	Cuboid	No	No	ASP	NSGA-II
18	Rectangular	No	No	FLP	CPM
19	Rectilinear	No	No	ASP	SBRP
20	Rectangular	Yes	Yes	FLP	LP
21	Rectangular	Yes	No	FLP	LP
22	Rectangular	No	Yes	ASP	GADG
23	Rectangular	No	Yes	FLP	FA
24	Rectilinear	No	Yes	ASP	CNN
25	Rectangular	Yes	No	FLP	GA + A*-Search
SLP	Rectilinear	No	No	SLP	MA + LP + LS

Objectives & fitness evaluation

Most FLPs seek to minimize material handling cost between facilities. This is incorporated by some determined cost of adjacency or the amount of flow between the facilities. Additionally, the flow or adjacency cost is multiplied with the distance between the facilities. The fitness of a solution for the FLP is most often comprised only of this objective which is commonly handled by a weighted sum. The only approach out of the 11 FLP papers considered which deviates from this is (Tari and Neghabi 2018). The authors instead implement the material handling cost more implicitly by seeking to maximize boundary lengths between facilities. Thus, prioritizing direct adjacency above minimizing distance between the facilities. Proximity and adjacency are essential also for the SLP, but they are merely a small part of the relevant objectives. Thus, FLP are insufficiently relatable to SLP with regards to objectives.

ASP considers a much wider range of objectives, which in turn requires a more sophisticated fitness evaluation. Wu et al. (2019) suggest an approach which avoids having to define objectives and compute their corresponding objective values explicitly. The authors implemented a neural network and trained it on large data-sets, over $120k$ instances, of existing residential buildings. This approach is, however, highly

Table 2.5: Paper overview for Table 2.4

1	Chwif et al. (1998)
2	J.-G. Kim and Y.-D. Kim (1999)
3	Y. H. Lee and M. H. Lee (2002)
4	Michalek et al. (2002)
5	McKendall et al. (2006)
6	Xie and Sahinidis (2008)
7	S. S. Y. Wong and Chan (2009)
8	Komarudin and K. Y. Wong (2010)
9	Verma and Thakur (2010)
10	Dutta and Sarthak (2011)
11	Flack and Ross (2011)
12	Chatzikonstantinou and Bengisu (2016)
13	Zawidzki (2016)
14	Abotaleb et al. (2016)
15	Dino (2016)
16	Bazaati (2017)
17	Gürsel and Göktürk (2017)
18	Hammad et al. (2017)
19	Shekhawat and Duarte (2017)
20	Feng and Che (2018)
21	Tari and Neghabi (2018)
22	Wang et al. (2018)
23	Scalia et al. (2019)
24	Wu et al. (2019)
25	Besbes et al. (2020)

Table 2.6: Abbrevations for the solution method column in Table 2.4

MA	Memetic algorithm
GA	Genetic algorithm
CNN	Convolutional neural network
FA	Firefly algorithm
GADG	Graph approach to design generation
LP	Linear program solved using a commercial solver
SBRP	Spiral based rectangular placement
PSO	Particle swarm optimization
NSGA-II	Non dominated sorting genetic algorithm
EA	Evolutionary algorithm
DFS	Depth-first search
B&B	Branch & bound
CPM	Cutting plane method
SA	Simulated annealing
TS	Tabu search
ACO	Ant colony optimization
GDS	Gradient descent search
LS	Local search

restricted by the size of the problem instances. Wu et al. (2019) consider apartment layouts consisting of five to eight rooms. Considering the small size of the instances the NN is able to handle, similar solution approaches are most likely unsuitable for the SLP.

A decisive objective of the SLP is the area of the layout as it is directly connected to cost. Total area as an objective subject to minimization is common for ASPs in general (Shekhawat and Duarte 2017; Flack and Ross 2011; Zawidzki 2016). More specifically, Zawidzki (2016) seeks to minimize area by minimizing the size of hallways which are used to connect the rooms. As the suggested size of the rooms in the SLP are defined in the RSD, minimizing area is accomplished through efficient hallways. The purpose of the hallways is to enable to move from A to B in a layout. In the literature this is referred to as *connectivity*, which is a common objective for the ASP as well as the SLP (Zawidzki 2016; Flack and Ross 2011). In addition to connectivity is *reachability* introduced in multiple papers. Reachability is the number of rooms passed through when moving from A to B, and is an objective subject to minimization. Flack and Ross (2011) handle reachability by implementing an adjacency graph with the objective of minimizing its depth.

Some layout problems are essentially constraint satisfaction problems, where the goal simply is to allocate the rooms to fulfil a number of requirements. However, the requirements can either be considered as absolute or implemented as soft constraints. Dino (2016) considers rooms wit a given desired size. The author implements a single objective evolutionary algorithm where the goal is to minimize the rooms deviation from these sizes. Hence, Dino (2016) models the size ”requirements” as soft constraints in the fitness function. It is not necessarily beneficial or possible to completely adhere to the suggested room sizes and shapes in the SLP. Hence, similar techniques as the one proposed by Dino (2016) must be considered when solving the SLP.

Some of the papers considered include most of the aforementioned objectives as well as additional problem-specific objectives; geometric simplicity of the allocated space elements (Zawidzki 2016) and maximizing rooms with window access (Zawidzki 2016). As the composition of objectives determining the fitness of a solution becomes more complex, it demands more sophisticated fitness evaluations. The SLP is inherently multi-objective, and efficiently prioritizing the objectives is crucial to obtain desirable results. Flack and Ross (2011) explore alternatives to the weighted-sum method, such as Pareto ranking and the sum of ranks. Gürsel and Göktürk (2017) and Chatzikonstantinou and Bengisu (2016) utilized NSGA-II, a specific GA implementation to handle multi-objective problems.

Multi-stage models

The complexity of layout problems can make it necessary or beneficial to divide the problem into sub-problems. In practice, this can be done by considering a subset of objectives or space elements at the same time. Additionally, not having to consider all aspects simultaneously simplifies formulating and representing the problem, as well as choosing the solution approach. Solving the problem step-wise facilitates applying more suitable models in each step. A number of multi-stage hybrid models are proposed in the literature (Michalek et al. 2002; Wu et al. 2019; Zawidzki 2016; Y. H. Lee and M. H. Lee 2002; Besbes et al. 2020; Gürsel and Göktürk 2017).

Michalek et al. (2002) and Wu et al. (2019) consider ASPs. Both choose to divide the problem into two stages. First, the topology of the rooms are determined, and then a corresponding geometric solution is formed. A topology represents the relative positioning of the rooms, while a geometry specifies the exact position and dimensions of the rooms. Wu et al. (2019) apply a convolution neural net in both stages, while Michalek et al. (2002) employ an evolutionary algorithm in the first stage and both a genetic algorithm and simulated annealing in the second stage. The model suggested by Michalek et al. (2002) iterates between stage one and two until a desirable layout is obtained.

Zawidzki (2016) suggest a three-stage model to solve the ASP. In the first stage, Zawidzki (2016) formulates the problem as a CSP and exploits a depth-first search to find feasible layouts. The solutions are then classified, as the authors define it, as ”proper” or ”improper” using a feed-forward neural network. The last stage ranks the proper layouts based on several objectives using a weighted-sum fitness function. Applying a DFS unguided by the objectives is not an efficient way to explore a large solution space. Thus, an evident drawback of the model is its ability to handle large problem instances, making it less comparable to the SLP.

To efficiently explore the solution space, Y. H. Lee and M. H. Lee (2002) suggest a hybrid genetic algorithm (HGA). The authors combine GA, TS and SA to make up for a weakness in the GA by employing the strong points of TS and SA that find local solutions rapidly (Y. H. Lee and M. H. Lee 2002). The model locally optimizes the solutions in the initial population using the TS and SA before they are propagated to a standard GA procedure. Essentially, the first step is a sophisticated initialization which provides the GA with locally optimized solutions for it to assess and improve at a global level.

2.3 Our contribution

Various layout problems are well studied, but the research conducted on School Layout Problems is non-existing. This thesis sheds light on the unexplored challenges the SLP offers. The SLP considers large problem instances with heterogeneous rooms which are subject to variable dimensions and sizes. In addition, numerous objectives ranging from proximity between the rooms to natural lighting conditions are vital when generating the layouts. Most research works on ASPs are concerned with a given footprint and allocating the space elements to fit within this given area. The practical problem of building a school is also concerned with new constructions. As a result, this thesis considers allocating rooms on a given building site, and creating the footprint based on the room locations. Collectively, these aspects comprise a highly complex problem, which sets demanding requirements for problem formulation, modelling decisions and solution approaches.

The solution method suggested in this thesis is a carefully assembled multi-stage approach. The purpose is to successively consider aspects of the SLP, as assessing all aspects of the problem simultaneously is considered too complex to obtain desirable results. Additionally, dividing into stages allows exploiting the strengths of multiple solution approaches. Specifically, this thesis presents an algorithm combining an MA, a mathematical model and an LS. The robustness and flexibility of this approach makes it likely to be efficient also on a wide range of multi-objective layout problems.

Chapter 3

Problem description

This chapter discusses the school layout problem studied in this thesis. Consider an architect firm who wants to enter into a bidding process for a new school building. The main input available to the firm is the information in the room specification document. Given the RSD, it is the architects task to design an efficient layout. Several factors, such as the number of rooms and objectives, make this a highly complex problem. In essence, the number of possible layouts are endless and developing a single layout suggestion requires lots of work. Increasing the efficiency of the development process will help architects create better school layouts.

To develop a school layout, an architect must place rooms, doors, windows, hallways and vertical transportation. The primary task is to place rooms with varying sizes and functions on the building site. Their shapes are not preset, and thus architects must ensure they are efficient. In this thesis, rooms are required to be rectangular, which is common for rooms in school buildings. A rooms' door(s) must be placed so that the room has a natural access point(s). It is also essential to recognize the relations between the rooms. The RSD explicitly states many of these relations. For instance, every room is part of a neighbourhood. There are also implicit interdependencies between rooms, such as noise pollution, that an architect must take into account.

Another challenging task is allocating hallways. In a feasible solution to the SLP, it is required that every room is connected through hallways. Connectivity should be met without creating an excessive amount of hallways, because extra square meters are costly. Architects must also consider vertical transportation and outdoor area. The location of vertical transportation impacts the validity and effectiveness of the school layout, while the outdoor area adds complexity to the SLP due to the interaction between the outdoors and indoors.

This thesis makes some simplifications to reduce the SLP's complexity. The building site used as input is a flat square which is large enough to fit the rooms and hallways comfortably. Since school sites usually have room for a large outdoor area, this is an insignificant simplification. The SLP studied in this thesis considers the construction of a single building with no preset footprint. Thus, the exterior walls are not set, but are placed after the room and hallway locations have been decided. Both exterior and interior walls need to be horizontal or vertical in relation to the site. Additionally, this thesis focuses on a single, ground floor layout. The assumption of a single floor reduces the problem to two dimensions, and removes vertical transportation from the problem.

The constraints of the SLP are mostly related to the room specifications in the RSD. These are size, shape, proximity and natural lighting requirements. Some constraints, such as noise pollution and emergency exit access, apply to the real world SLP, but are not taken into account in this thesis. These aspects are left for the architect to consider when modifying the layouts this thesis produces.

A challenging part of the SLP is to determine the effectiveness of a layout. A great number of objectives measures effectiveness and the weighting of these objectives vary between architects. Aesthetics, planning for people flow, even creating a layout that suppresses bullying, are all examples of criteria architects reflect on when designing a school building. While they are undoubtedly important, they are also difficult to quantify. This thesis selects objectives based on importance and their ability to be measured quantitatively and objectively. As discussed in Chapter 1, cost is a critical criteria. Architects consider two main cost drivers, the total building area and the number of exterior building corners. Thus, these are the two key objectives of this thesis. The suggested size of the rooms in the SLP are defined in the room specification document. Building area is therefore minimized by minimizing the hallway area. In addition to cost, this thesis focuses on aesthetics through obtaining geometric simplicity in the layouts.

Chapter 4

Solution method

This chapter presents the overall architecture of the solution method developed in this thesis. The solution method consists of three stages; a memetic algorithm, a mathematical model and a local search. The stages are briefly explained in this chapter and further elaborated in Chapter 5, 6 and 7. First, Section 4.1 provides a general overview of the different stages of the solution method. The chosen solution method and the interaction between the stages are discussed in Section 4.2. Section 4.3 provides a pipeline example illustrating the behaviour of the three components of the solution method. Lastly, Section 4.4 presents assumptions and clarifications to the SLP solved in this thesis.

4.1 Overall architecture

The first stage is a memetic algorithm, consisting of a genetic algorithm and a local search, which creates a first draft of the school layout. This stage aims to create layouts which fulfill the requirements in the RSD while minimizing building area. The second stage is a mathematical model which takes the layout from stage one as input and attempts to minimize the number of corners in each neighbourhood separately. In this stage, adjustments to the sizes and locations of the rooms are allowed to create a more geometric simplistic layout with fewer corners. Finally, in stage three, a local search is conducted to minimize the number of exterior corners and total building area. Figure 4.1 illustrates the overall architecture.

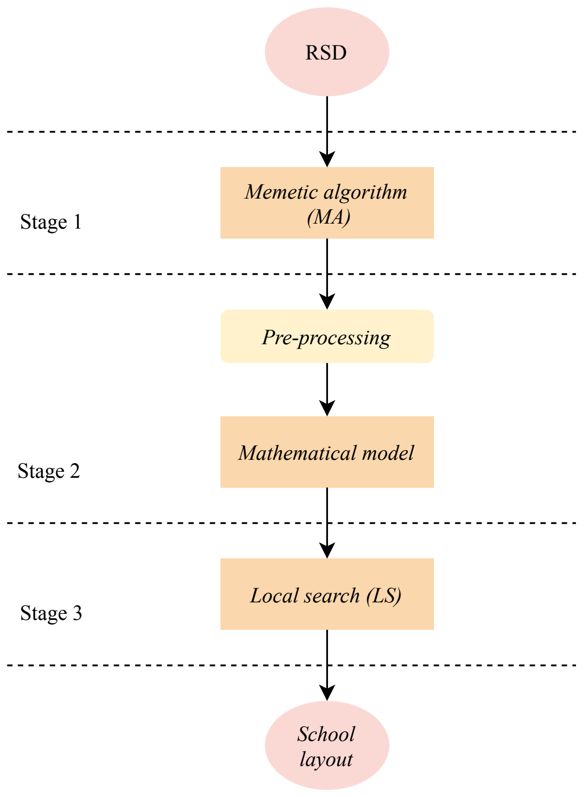


Figure 4.1: Overall architecture of the solution method

4.1.1 Stage one - memetic algorithm

The memetic algorithm allocates rooms and hallways to the building site, jointly forming a school layout. The genetic algorithm initializes the solutions by using the list of rooms, with their corresponding size suggestions and proximity requirements, in the RSD as input. Since the RSD does not specify hallways, this thesis implements a sophisticated algorithm for dynamically generating hallways. This algorithm is explained in Section 5.8.1. The allocation of rooms is done by mutation and crossover operators, which are standard operations in genetic algorithms. The local search exploits a gradient descent technique to make small, improving changes to the layout. The greediness of the local search complements the inherent randomness of the GA. Collectively, they balance exploration and exploitation and search the solution space efficiently.

The goal of the memetic algorithm is to produce a school layout which satisfies the requirements in the RSD, along with additional constraints such as hallway connections between rooms. Additionally, it aims to minimize the total building area. A weighted-sum fitness function is used to handle the multi-objective MA. The output propagated to the next stage is a complete layout suggestion where rooms and hallways are placed on the site.

4.1.2 Stage two - mathematical model

This stage is a mathematical model formulated as an integer linear program (ILP) and solved using a commercial mathematical optimization solver. The model is applied to each neighbourhood separately, aiming to minimize the number of corners by adjusting the size and location of the rooms. This stage aids the three-stage algorithm with creating geometric simplistic solutions, as this is directly tied to the number of corners, both interior and exterior.

The model is preceded by a processing step to connect phase one and two. Additionally, the pre-processing step exploits information in the layout generated by the MA to decrease run time and increase solution quality of phase two. Stage one sets the relative position of the neighbourhoods. To maintain this topology, stage two is only allowed to place rooms in a subarea of the site. Based on the neighbourhood's position in the output from stage one, the algorithm determines a subarea for each neighbourhood. Additionally, the layout produced in stage one includes hallways which connect the neighbourhoods. The location of these hallways are used to prevent stage two from placing rooms that breaks connectivity.

Once a neighbourhood is optimized, it is inserted back into the layout. Next, the hallways are regenerated using the same procedure as in stage one. The output from stage two is a layout where the neighbourhoods are locally optimized with respect to number of corners.

4.1.3 Stage three - local search

When the two first stages are completed, a layout with each neighbourhood optimized separately, is designed. Stage three performs a local search to enhance the layout in terms of cost by minimizing the building area and the number of exterior corners. As fewer exterior corners often yields more natural-looking layouts, this stage also enhances geometric simplicity. The previous stage minimizes the number of corners within a neighbourhood, while the local search seeks to minimize the number of exterior corners in the layout as a whole. The LS moves a neighbourhood to the best position within a given radius. All rooms within the neighbourhood are moved the same distance in the chosen direction. This preserves the relative position of the rooms within the neighbourhoods. The output of stage three is the final school layout.

4.2 Algorithm composition

Layout problems are known to be complex and are generally NP-Hard. Consequently, it is not likely to find an optimal solution within an acceptable amount of time. Furthermore, given the subjective nature of the SLP, it does not necessarily make sense to seek an "optimal" solution. Solving the SLP is rather to find an adequate solution, adhering to the requirements in the RSD and satisfying both subjective and objective criteria, within an acceptable amount of time. This fact is vital to consider when choosing the solution approach.

Memetic algorithms have the advantage of producing reasonably good solutions in an acceptable amount of time, for problems with large search spaces. MAs achieve this through their ability to balance exploration and exploitation. Additionally, MA is flexible to domain specifics, has robust performance and supports multi-objective optimization (Zhao and Sannomiya 2001). Exploiting the strengths of MA in the initial phase allows to efficiently produce desirable layout suggestions. On the other hand, genetic algorithms perform successive operations such that the resulting changes collectively improve the solution. The GA chooses operations in a random manner. Therefore, as the number of consecutive changes necessary to improve the solution increases, the probability of the algorithm succeeding decreases. Some objectives require a great number of consecutive operations, e.g., the number of corners where multiple rooms must be placed and fit together. As a result, MA is inadequate as a stand-alone solution method.

Applying an exact method allows exploiting its guarantee of optimality, not having to rely on many desirable successive changes to the solution. However, exact methods are restricted by the complexity of the SLP, and applying them to the SLP as a whole is practically impossible. Hence, the algorithm in this thesis utilizes the memetic algorithm to determine the relative position of the neighbourhoods and rooms, substantially narrowing down the search space. This facilitates for optimizing each neighbourhood using the mathematical model locally. Another drawback of a mathematical model is the challenge of modelling complex objectives such as placing hallways to ensure connectivity between neighbourhoods. The output of the MA is optimized for such objectives. Therefore, the task of the mathematical model is to preserve these objectives, rather than explicitly incorporating them in the ILP as objectives subject to optimization. Heuristics are implemented to the model both to preserve the objectives considered in stage one, and to reduce the run time.

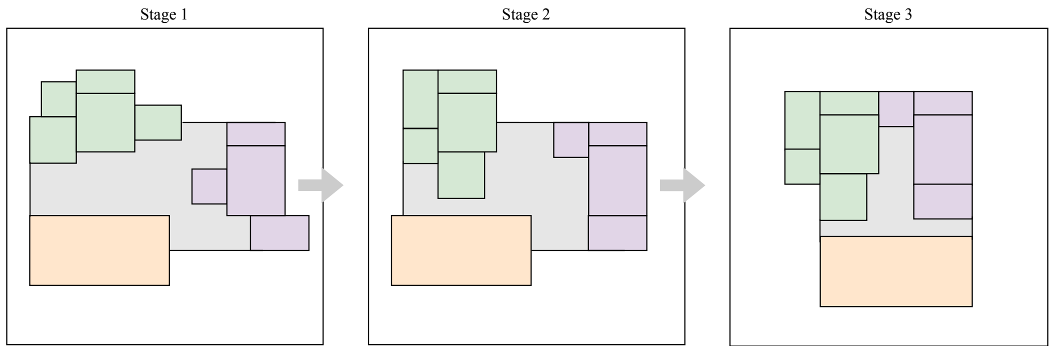
As the memetic algorithm is unsuitable to consider exterior corners and the mathematical model is unable to consider the SLP at the global level, the third stage is implemented to fit all neighbourhoods together. A local search moves from solution to solution by applying local changes, and the resulting jumps in the solution space are too small to explore the vast solution space of the SLP sufficiently. However, after stage two, the topology of the neighbourhoods and the position of rooms within a neighbourhood are determined. Hence, the successive, greedy and local changes performed in the local search is suited to solve the remaining problem. In addition to minimizing the number of exterior corners, the local search also considers a subset of the complex objectives included in the MA. By doing so, the third stage can fix requirement violations at the global level.

Given the variety of challenges faced with when solving the SLP, this thesis implements a multi-stage algorithm exploiting the strengths of multiple solution approaches. This hybridization and division of responsibility collectively forms a robust and flexible algorithm.

4.3 Pipeline example

Figure 4.2 illustrates the evolution of a simple school layout through the three stages. This layout contains three neighbourhoods coloured in green, purple, and orange. Each coloured rectangle represents a room, where the rooms belonging to the same neighbourhood have the same colour. The light grey area is a hallway connecting the neighbourhoods while the white area is the outdoor area.

Figure 4.2 (*a*) shows a possible output from the memetic algorithm. In this layout, the given proximity requirements within each neighbourhood is fulfilled, and a direct path connects all neighbourhoods through hallways. Furthermore, the rooms which requires natural lighting have window access. The green and purple neighbourhood have 14 and 12 corners, respectively, while the school as a whole has 16 exterior corners. The layout proceeds to stage two, and the result is shown in (*b*). In the green neighbourhood, both the size and location of several rooms has changed, decreasing the number of corners from 14 to 8. For the purple neighbourhood, all rooms maintained their suggested size, but by relocating several rooms, the number of corners decreased from 12 to 6. The layout still fulfils the requirements mentioned above regarding connectivity, proximity and natural lighting. Finally, stage three performs a local search. As seen from Figure 4.2 (*b*), the locations of all three neighbourhoods have changed, decreasing the number of exterior corners to six. Note that the relative positions of the rooms within a neighbourhood remain unchanged from stage two to three.



(a) Layout after the memetic algorithm (b) Layout after the mathematical model (c) Final layout after the local search

Figure 4.2: Possible evolution of a simple school layout through the three stages

4.4 Assumptions and simplifications

To implement the presented three-stage algorithm, some assumptions and simplifications are made. The following sections present the assumptions and simplifications defining the SLP examined in this thesis.

4.4.1 Single floor layout and discrete building site

As this thesis considers a 2-dimensional single floor SLP, staircases, elevators and the height of rooms are not considered, and the building site is assumed to be level. Furthermore, the site is a rectangle of $a \times b$ meters, where a and b are chosen such that the site fits all the rooms comfortably. The site is split up in cells of a square meter to form a grid, shown in Figure 4.3. As this is a discrete representation, the number of possible room locations is finite.

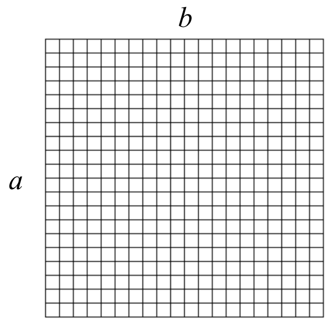


Figure 4.3: A building site of $a \times b$ metres with a rows and b columns. Each cell is a possible room location.

4.4.2 Rooms and hallways

The SLP only considers rectangular rooms. The required rooms and their suggested sizes are given in the RSD and is used as input in the algorithm. Completely adhering to the given size suggestions are neither achievable nor desirable as it would adversely affect the quality of the solutions. Therefore, a small deviation from these sizes is allowed in the algorithm. Furthermore, the dimension of the room is subject to change within its aspect ratio bound. The RSD specifies the aspect ratio bound of a subset of rooms, for example, classrooms should be close to quadratic. For the rooms without a specified bound, a suitable one is chosen. See Chapter 8 for further elaboration on the chosen aspect ratio bounds.

Hallways are not an input to the algorithm and are handled differently than rooms. Space that is within the building, but not part of a room, is defined as a hallway. Consequently, hallways are rectilinear polygons. Figure 4.4 shows a small school layout consisting of six rooms, r_1 to r_6, where the solid black line marks the exterior walls of the building. The grey area represents hallways, as the area is part of the building, but not part of a particular room.

Each room is part of a larger area, called a neighbourhood. A neighbourhood contains rooms that have to be grouped together according to the RSD. For instance, all *8th grade* classrooms should be in close proximity to each other. In Figure 4.4,

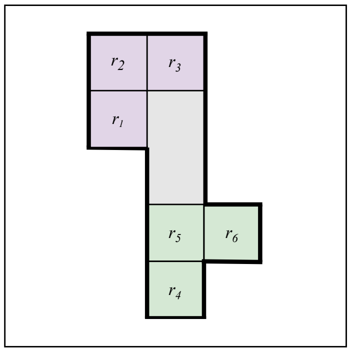


Figure 4.4: School layout with six rooms, r_1 - r_6. The rey area is considered a hallway.

rooms in the same neighbourhood are illustrated with the same colour. In RSDs, neighbourhoods usually contain a room which, in addition to its own purpose, serve as an access point to the other rooms in that neighbourhood. The RSD used as input in this thesis specify such a room for each neighbourhood. This room will be referred to as the *main room* throughout this thesis.

Schools usually have a multi-functional hub which contains several essential functions, for example, a cafeteria and an assembly hall. Most RSDs do not specify the size of the components in the hub, but specify its total area. Based on this, the multi-functional hub is modelled as one large room, and is further referred to as the *hub*.

A room can have two types of neighbours. *Door-neighbours* must share a wall, such as r_1 and r_2 in Figure 4.4. The shared wall must be at least two meters wide, such that it can fit a door. When two neighbours satisfy this constraint, the rooms are *attached*. In this thesis, all rooms need to be attached to the main room of its neighbourhood. The other type of neighbour is a *hallway-neighbour*. Two hallway-neighbours are required to be attached or to be connected by a direct path through hallways. This pathway cannot lead through other rooms. For instance, there is a direct pathway between r_3 and r_5, but not r_3 and r_4. All main rooms are required to be hallway-neighbour with the hub. Combining door- and hallway-neighbour requirements ensures connectivity between all rooms in the school.

Chapter 5

Memetic algorithm

In this chapter, the first stage of the solution method is described. Section 5.1 provides a general overview of the different phases in a memetic algorithm consisting of a genetic algorithm and a local search. The objectives of the MA are presented and elaborated in Section 5.2. Section 5.3 describes the representation of the chromosomes, including both the genotype and the phenotype. Finally, sections 5.4 - 5.9 describe the additional steps of the algorithm. These steps are the population initialization, crossover, mutations, local search, fitness evaluation and selection. This chapter revisits excerpts from Hermanrud et al. (2019), as the MA is an improved version of the MA developed in this specialization project.

5.1 Memetic algorithm procedure

The memetic algorithm consists of a genetic algorithm, complemented with a local search. Genetic algorithms work on a pool of individuals, called the population. The number of individuals in the population is called the population size. An individual is a solution - in this case, a school layout. Each individual has a chromosome which represents the features of the individual. The main idea of GA is to make small changes to the individuals over time, steering the development of the population in the desired direction. This development is encouraged by keeping the individuals that have a desired evolution and eliminating those who do not. The flow chart of the genetic algorithm, complemented with a local search, is illustrated in Figure 5.1.

First, the initial population is generated, and each individual is given a fitness score based on the features of the chromosome. The next phase is to generate children, or offspring, of the current individuals in the population. Children are generated by performing crossover and mutations. During crossover, two individuals, called parents, are chosen from the population through ”natural selection”. The selection considers the fitness of the individuals, giving the fittest individuals a higher probability of being chosen. Every pair of parents generate two offspring by combining the genetic information of the parents.

Mutation is the operation of producing children by making changes to a single parent. Mutation occurs to maintain and introduce diversity in the population. The crossover- and mutation-phase continue until the number of children is equal to the initial population size.

Finally, the fitness of the children is evaluated. The individuals for the next population is chosen from the pool of the newly generated children and the previous population, maintaining the same population size. These individuals are selected based on their fitness. This is commonly called the selection-phase. Each population created is called a generation. The algorithm terminates after a fixed number of generations, or when a solution is considered sufficiently good.

The steps mentioned above are the classical GA-steps. In addition to these steps, the algorithm performs a local search of the population and the generated children before evaluating the individuals, composing the memetic algorithm.

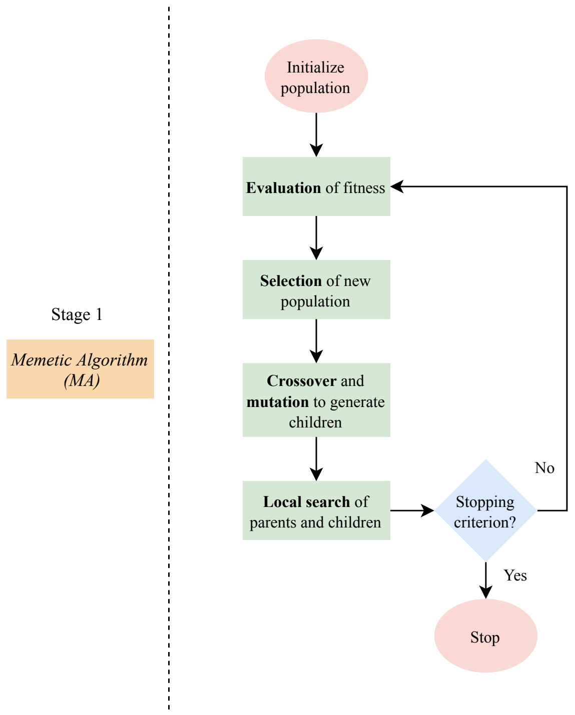
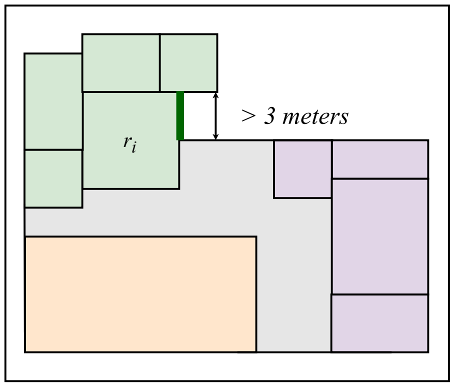

Stage 1
Memetic Algorithm (MA)
Initialize population
Evaluation of fitness
Selection of new population
Crossover and mutation to generate children
Local search of parents and children
Stopping criterion?
No
Yes
Stop

Figure 5.1: Memetic algorithm flow chart

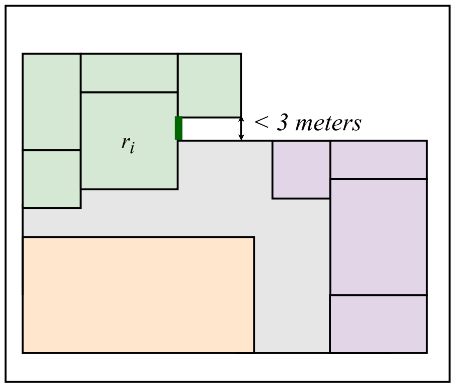
5.2 Objectives

The memetic algorithm considers a multi-objective optimization problem with seven variables using the weighted-sum method. All the objectives are to be minimized, and therefore all objectives f_i have a corresponding positive weight, w_i. The MA includes the following objective variables:

- f_1 **- Overlap**: The total overlap between all rooms in terms of square metres. Two rooms overlap when they cover the same space.
- f_2 **- Connectivity**: The number of hallway-neighbour pairs which do not have a direct pathway through a hallway connecting them.
- f_3 **- Narrow hallways**: The amount of narrow hallways in terms of square meters. A narrow hallway is a hallway no wider than 3 meters. This considered unusable area.
- f_4 **- Door-neighbour distance**: The sum of the Manhattan distance between all door-neighbours. If two door-neighbours are attached, their Manhattan distance is zero.
- f_5 **- Window access**: The number of rooms required to have windows which do not have access. A room has window access if one of its walls is an exterior wall, and has at least nine square meters of outdoor area next to it. Each side must be at least three meters wide (3×3) to allow for daylight. Figure 5.2 shows an example where room r_i has window access in (a), but not in (b). r_i has an exterior wall, marked in dark green, but in case (b) the wall is shorter than three meters, preventing the room from having window access.
- f_6 **- Hallway area**: The amount of hallway area in the layout. The hallway area is given as the percentage of the total building area.
- f_7 **- Excess neighbourhood area**: The excess neighbourhood area is defined as the area not occupied by rooms within the envelope of a neighbourhood. This objective is the sum of the excess neighbourhood areas. Its purpose is to reward compact neighbourhoods. In Figure 5.3, the excess neighbourhood area of two neighbourhoods are coloured in yellow.



(a) The exterior wall is longer than 3 meters, providing the room with more than 3x3 meters of outdoor area next to it. Thus, it has window access.



(b) The exterior wall is shorter than 3 meters, and the room does not fulfill the criteria of having window access.

Figure 5.2: Examples where a room r_i has window access in (a), but not in (b). Its exterior wall is marked with a dark green solid line.

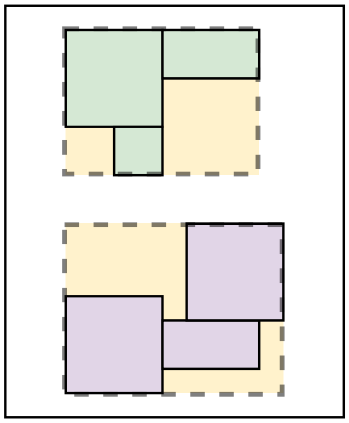


Figure 5.3: The excess neighbourhood area of a purple and green neighbourhood is illustrated in yellow. The envelope (dashed line) minus the area of the rooms within the neighbourhood make up the excess neighbourhood area.

For a solution to be *feasible*, objective f_1 and f_2 must be fulfilled: a layout cannot contain overlapping rooms, and for a school to be functional, it must be possible to get from a neighbourhood to another through a hallway. An objective is fulfilled if its corresponding value is zero. These constraints are implemented as objectives in the fitness function as soft constraints, rather than hard constraints excluding such solutions from the search space. Further, a solution is *satisfactory* if objective $f_1 - f_5$ is fulfilled. Note that these are the five objectives where an objective value of zero is highly achievable, contrary to objective f_6 and f_7.

It is important to note that the magnitude of the objective values differs. For instance, the upper bound of f_5 is equal to the number of rooms requiring window access, while the size of the site bounds the objective value of f_7. Hence, all objectives are scaled to obtain comparable magnitudes. This also allows the weights to take on similar values, making it more intuitive to understand the effects of the objectives and tune the weights accordingly.

An objectives importance can vary for different stages of a run. To sufficiently explore the search space, infeasible solutions should be tolerated as they might be a necessary step towards improved feasible solutions. Hence, it is likely beneficial to moderate the penalty of violating connectivity and overlap in the first generations. The importance of obtaining feasible solutions increases as the run progresses, and the weights w_1 and w_2 should be adjusted accordingly. The MA facilitates this, and similar type of behaviour, by using dynamic weights.

5.3 Representation

Each individual has a set of properties referred to as its chromosome. Choosing a beneficial chromosome representation is a crucial part of developing GAs, as it determines the limits and possibilities for making changes to and scoring the individuals. The representation consists of a genotype G, a phenotype P and an invertible mapping $f\colon G \to P$.

5.3.1 Genotype

A genotype is typically a low-level representation of the solution, which is easily understood and manipulated. In this case, the chromosomes genotype is represented as a hash map with the room IDs as keys and a room object as the corresponding value. A room object contains information specific to the chromosome, which is its current position on the site defined by the (x, y)-coordinate of the upper left corner, along with the room's length l and width w. A room object r is illustrated in Figure 5.4.

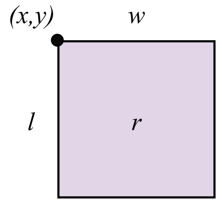


Figure 5.4: Room r with (x, y) coordinate, width w and length l

5.3.2 Phenotype

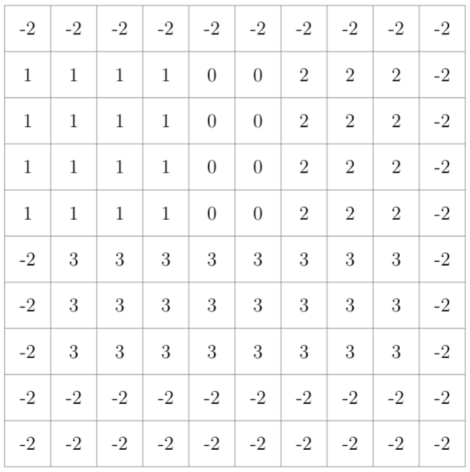
The phenotype is the physical representation of the chromosome, which in this case is the school layout design. In this thesis, the phenotype is the site matrix where a cell represents a square meter of the site, as mentioned in Section 4.4.1. Each cell can be assigned these different values; room ID, overlap ID, hallway ID, narrow hallway ID or outdoors ID.

If a room in the genotype covers cell (i, j), the room's ID is assigned to the cell value a_{ij}. All room IDs are positive integers. If two or more rooms occupy the same cell, the cell is given a negative overlap ID of -1. Overlap is visualized with the colour red in the school layout. If a cell is considered a hallway, the cell is assigned a hallway ID of 0, illustrated as grey in the school layout. Narrow hallways are represented in the matrix as -3 and are illustrated with the colour magenta, while cells that are not part of the building have an outdoors ID of -2. This area is illustrated in white. The cell IDs are shown in Table 5.1

A simple school layout is shown in Figure 5.5. Figure 5.5 (a) shows its site matrix, while 5.5 (b) illustrate the corresponding school layout. The different cell IDs are drawn in different colours. The white area is part of the site, but not part of the building. The three rooms with ID 1, 2 and 3 are illustrated as three coloured rectangles, r_1, r_2 and r_3. Lastly, the light grey area with ID 0 is a hallway connecting the three rooms.

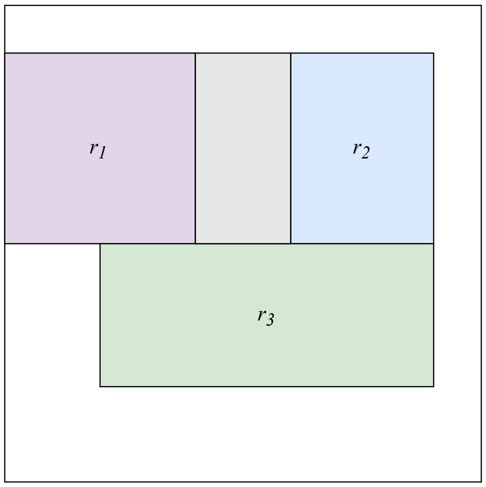
Table 5.1: Cell IDs

Cell IDs	
Hallways	0
Overlap	-1
Outdoors	-2
Narrow hallways	-3
Rooms	$\mathbb{Z}$



-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
1	1	1	1	0	0	2	2	2	-2
1	1	1	1	0	0	2	2	2	-2
1	1	1	1	0	0	2	2	2	-2
1	1	1	1	0	0	2	2	2	-2
-2	3	3	3	3	3	3	3	3	-2
-2	3	3	3	3	3	3	3	3	-2
-2	3	3	3	3	3	3	3	3	-2
-2	-2	-2	-2	-2	-2	-2	-2	-2	-2
-2	-2	-2	-2	-2	-2	-2	-2	-2	-2

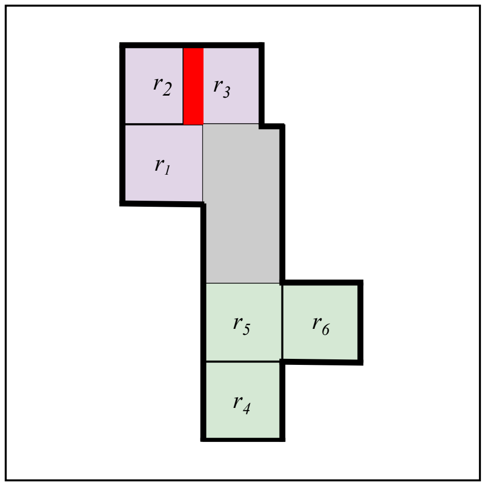
(a) Phenotype, represented by a site matrix



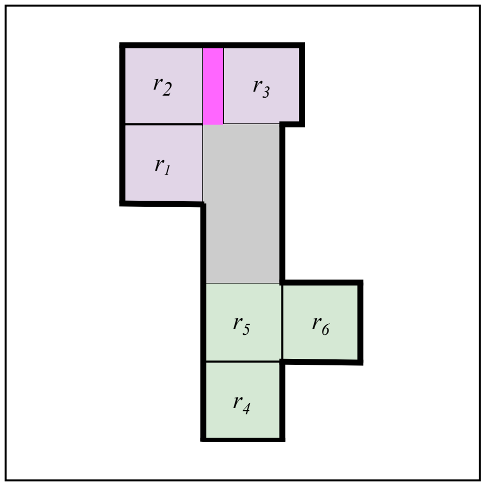
(b) Corresponding school layout

Figure 5.5: A phenotype matrix and its corresponding school layout

More extensive examples are given in Figure 5.6. Figure 5.6 (a) shows the case with overlapping rooms, where room r_2 and r_3 overlap. Thus their shared space is coloured red. Figure 5.6 (b) shows a case containing a narrow hallway, where the hallway between room r_2 and r_3 is narrower than 3 meters. The area is therefore coloured magenta.



(a) School layout with overlap. Room r_2 and r_3 overlap, and is illustrated in red.



(b) School layout with narrow hallways, which is illustrated in magenta

Figure 5.6: School layouts with overlap and narrow hallways

5.4 Population initialization

When initializing the population, two primary methods are considered - random or using a heuristic. With a random initialization, each rooms initial coordinate is chosen at random. This often creates individuals with a poor initial fitness score. By using a heuristic, problem-specific information can be exploited to improve the initial solutions.

The initialization of the population should be given significant consideration. Completely random initialization may prevent the algorithm from finding promising solutions, but has the positive effect of maintaining diversity in the population. Using a heuristic may result in an initial population with fit individuals, but less diversity, preventing exploration of the search space. Research works on GA have found that a combination of randomness and a heuristic has shown to be effective, as it exploits the good properties of both.

The MA initializes the population combining random and heuristic initialization. The heuristic initialization works as follows. First, the building site is divided into n sub-sites, where n is equal to the number of neighbourhoods. In Figure 5.7 there are 4 sub-sites named from A to D. For each sub-site, one of the neighbourhoods is chosen at random. Then, all rooms within that neighbourhood are given random locations within the sub-site. This results in an initial positioning where all rooms in a neighbourhood are close to each other. Hence, it exploits information about the problem when initializing an individual. The random initialization randomly allocates rooms within the bounds of the site. Whether the heuristic or random approach is chosen to initialize an individual is determined using a probability parameter, p_h.

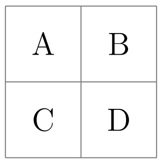


Figure 5.7: Building site divided into four sub-sites, A-D

5.5 Crossover

Crossover, also called recombination, is a genetic operator combining genes from two parents to generate new offspring. The two parents are chosen through parent selection, where fitter chromosomes have a higher probability of being selected. The idea is that mating two fit parents is more likely to generate children with improved features. The simplest form of crossover is a single-point crossover, as illustrated in Figure 5.8. The crossover operator implemented swaps neighbourhoods between parent chromosomes.

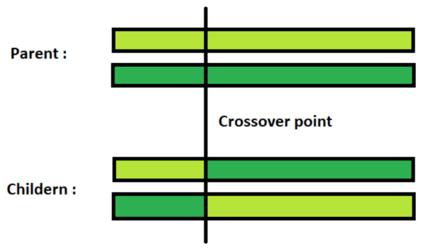


Figure 5.8: Single-point crossover

For parent selection, several alternatives exist. The most common are roulette wheel selection, tournament selection and rank selection. Tournament selection is implemented as the parent selection mechanism.

5.5.1 Tournament selection

To select parents for crossover, tournament selection is implemented and is illustrated in Figure 5.9. Tournament selection chooses a percentage p_t of the population at random as competitors, this is called the tournament size. The fittest chromosome of the competitors is chosen. Tournament selection occurs twice, resulting in two distinct parents chosen for crossover.

The purpose of tournament selection is to explore the state space by allowing different individuals to mate, but also exploit the fittest individuals by giving them a higher probability of reproducing. A large tournament size can cause one, fit solution to dominate the population since it prevents a diverse set of chromosomes from mating and producing offspring. This results in a lack of exploration.

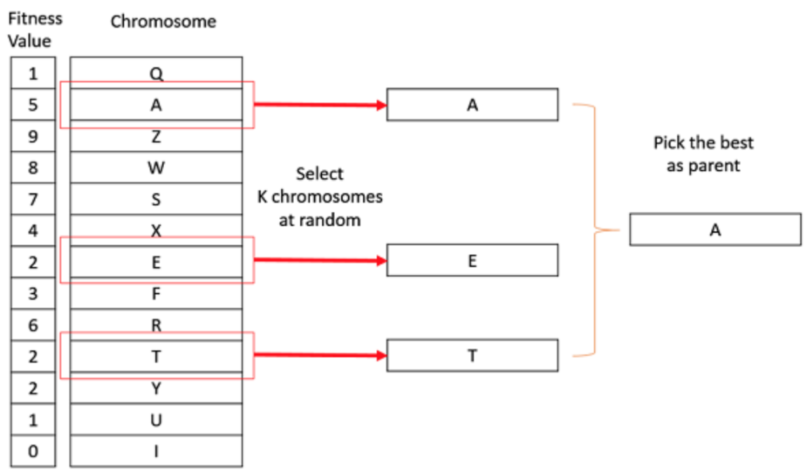


Figure 5.9: Tournament selection. Chromosome A, E and T are chosen as competitors. A is the fittest chromosome and wins the tournament.

5.5.2 Swap neighbourhoods crossover

In the implemented crossover operator, a neighbourhood n is chosen at random. Then, neighbourhood n in the two parent chromosomes are swapped - the relative position of the rooms within neighbourhood n in chromosome c_1 is swapped with the relative position of the rooms within the same neighbourhood in chromosome c_2. This happens twice, such that two different neighbourhoods are swapped. Crossover occurs with a probability p_c for every generation.

In Figure 5.10, each neighbourhood has its own colour. Two neighbourhoods are chosen at random for swapping - the purple and orange in the figure. The result is two children with many of the same features as its parents, but with two neighbourhoods swapped as shown by the circles in *Offspring 1* and *Offspring 2*.

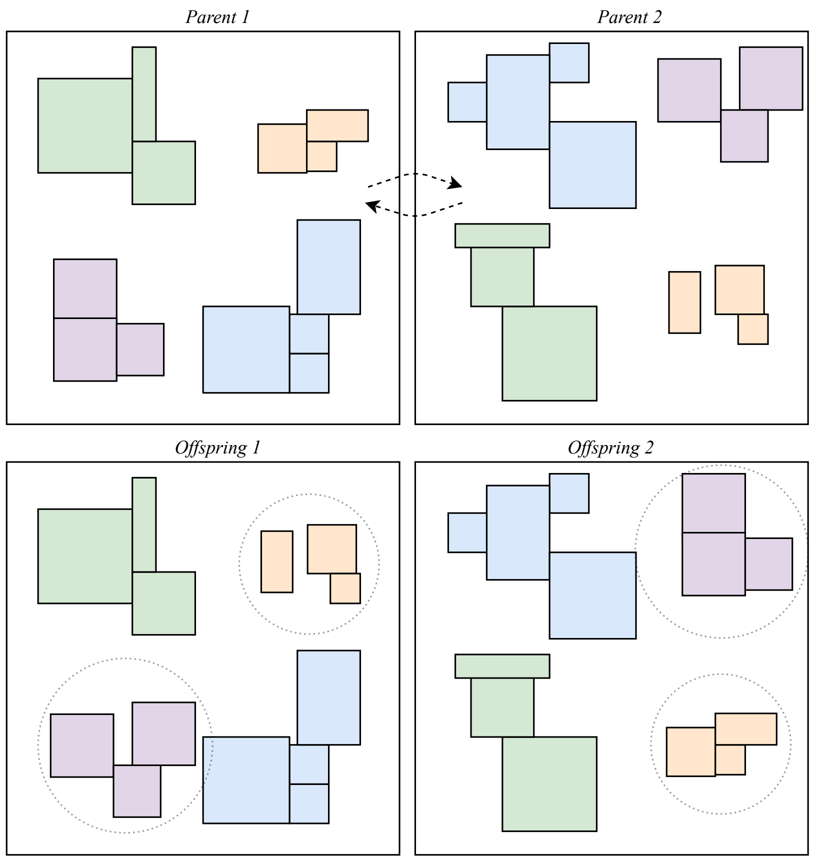


Figure 5.10: Swap neighbourhood crossover. The purple and orange neighbourhoods are swapped between the two parents, generating two offspring. The change of neighbourhoods within each offspring is illustrated by a circle.

Crossover improves both exploration and exploitation. It creates a chromosome with unique features, which explores the state space. Additionally, it exploits desirable properties, as the parents are selected based on fitness.

5.6 Mutation

Mutation is a genetic operator to maintain diversity from one generation to the next. Thus, mutation is used to explore the state space in new regions. Due to the complexity of the SLP, mutations exploring specific problem knowledge is implemented. It has been shown that mutations are essential for the convergence of GAs while crossover is not (R. L. Haupt and S. E. Haupt 1998). Thus, the GA mainly relies on mutation operators.

Mutation happens with probability p_m, referred to as the mutation rate. If mutation occurs, each of the different mutation operators is applied with a probability of q_m for each mutation operator m. This allows for several mutations to happen in a single generation. Another approach would be to make mutation combinations manually and give these a probability of occurring. Since eight mutation operators are implemented, explicitly defining efficient combinations is a challenging task avoided by making the probabilities q_m conditionally independent.

5.6.1 Move room random

This mutation operator moves a room to a random location on the site and is illustrated in Figure 5.11. The operator is implemented to create randomness and explore the search space.

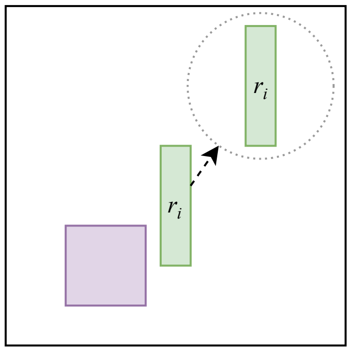


Figure 5.11: Move room random operator. Room r_i is moved to a random location.

5.6.2 Move neighbourhood random

Similar to the move room random-mutation, this operator moves a neighbourhood to a random location on the site. All rooms are shifted the same distance in x and y-direction, maintaining the relative position of the rooms within the neighbourhood. Figure 5.12 shows the operator.

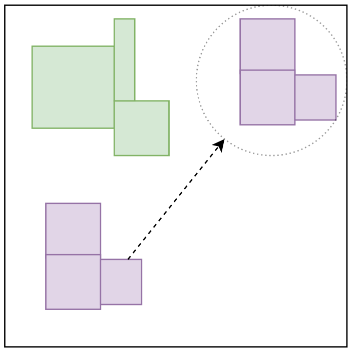


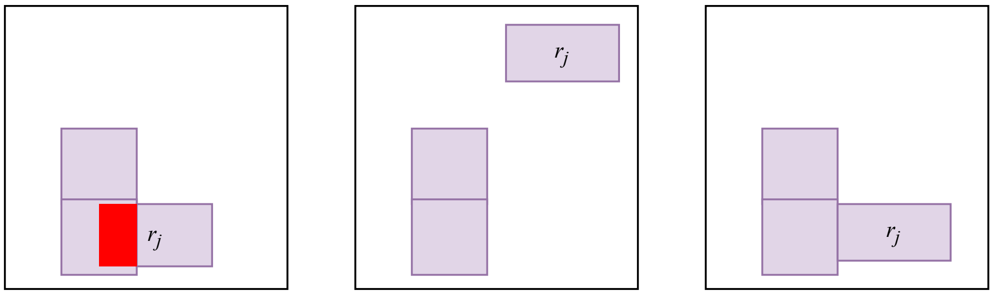
Figure 5.12: Move neighbourhood random operator. The purple neighbourhood is moved to a random location.

5.6.3 Swap rooms

The swap rooms-mutation chooses two random rooms and swaps their position.

5.6.4 Move overlapping room

In this mutation, both a random and smarter way to move an overlapping room is implemented. With a small probability p_o, a room that is overlapped is moved to a new random location. Else, the algorithm performs a local search of the room, trying to escape the overlap. The operator is illustrated in Figure 5.13. Room r_j in (a) overlaps another room, and is moved in one of two ways. (b) shows a possible result if the room is moved random, while (c) shows the result of a local search.



(a) Pre mutation. Room r_j overlaps another room.

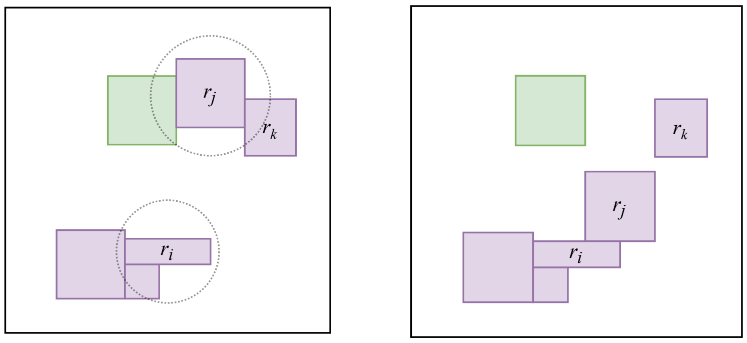
(b) Alternative one, post mutation. Room r_j is moved random.

(c) Alternative two, post mutation. Room r_j is moved by a local searc.

Figure 5.13: Move overlapping room operator

5.6.5 Move to unattached door-neighbour

This mutator is implemented to attach door-neighbours. The operator finds an unattached door-neighbour r_j of room r_i, and moves r_j to the closest side of r_i. The mutation is illustrated in Figure 5.14, where (a) is the pre mutation state and (b) shows the result of the operator where room r_j is attached to its door-neighbour r_i.



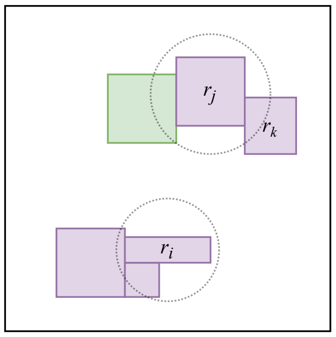
(a) Pre mutation. Room r_j is a door-neighbour of r_i, but the rooms are not attached

(b) Post mutation. Room r_j attached to r_i

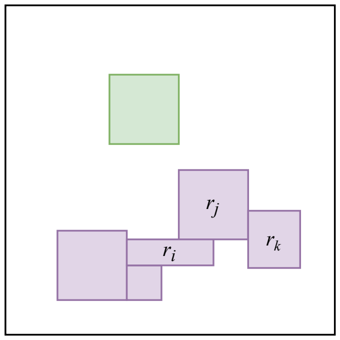
Figure 5.14: Move to unattached door-neighbour operator

5.6.6 Move attached door-neighbours

This operator is an extension of the operator in Section 5.6.5. The purpose of this mutation is to move unattached door-neighbours closer to each other, while not breaking with a room's already attached door-neighbours. This operator is illustrated in 5.15. A room r_i and one of its unattached door-neighbours r_j is picked for mutation, as shown in (a). Then, room r_j and its attached door-neighbours, here r_k, are moved to the closest side of r_i. The result of the operator is shown (b).



(a) Pre mutation. Room r_j is a door-neighbour of r_i.



(b) Post mutation. Room r_j attached to r_i while r_k still attached to r_j.

Figure 5.15: Move attached door-neighbours operator

5.6.7 Change room dimension

As mentioned, each room has an aspect ratio bound. This mutation changes the dimensions of a room to a random width and length without extending the aspect ratio bound.

5.6.8 Swap wall-sharing side

This mutation finds two rooms that are attached, r_i and r_j, and move room r_i to another side of room r_j. Whether it moves to the north, east, south or west side is chosen at random. The operator is illustrated in Figure 5.16, where r_i is moved from the east to the north side of r_j.

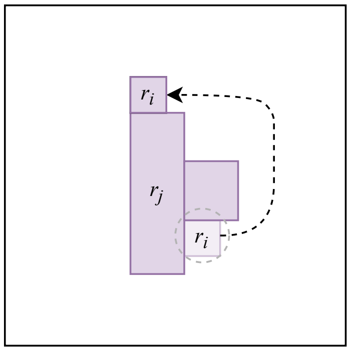


Figure 5.16: Swap wall-sharing side of attached door-neighbours. Room r_i is swapped from the east side to the north side of room r_j.

5.7 Local search

As mentioned, a local search is added as an additional step to the classical genetic algorithm, making the first stage of the solution method a memetic algorithm. The complexity of the SLP motivates this extension, which provides a guided movement of rooms. Local search happens with a probability p_l and for one room only in each generation.

The implemented local search is a gradient descent procedure which works as follows - pick a room r_i and try to move it one unit (one meter) in each direction. Then, if one of the new positions improves the fitness of the chromosome, move the room to that position and proceed with the local search. The number of moves is bounded to mitigate time complexity issues. The procedure is described in Algorithm 1.

Algorithm 1: Local search of room

```
Data: Phenotype $P$, Set of rooms $R$, maximum meters of movement $m$
Result: Updated layout with room possibly moved
LocalSearch ($P, R, m$)
    $Improvement \longleftarrow True$;
    $Bound \longleftarrow 0$;
    /* Pick room at random                                  */
    $r \longleftarrow Random(R)$;
    while $Improvement$ and $Bound < m$ do
        $Improvement \longleftarrow False$;
        /* Local search room                                */
        $LocalSearch(P, r)$;
        if $r.hasMoved$ then
            $Improvement \longleftarrow True$;
            $Bound \pm 1$;
        end
    end
    return $P$
end
```

5.8 Fitness evaluation

For some of the objectives defined in Section 5.2, pre-scoring processing or extensive calculation is needed. The effectiveness of these processes highly impacts the run time of the MA. Thus, sophisticated methods are developed to perform these calculations. The objectives are repeated for convenience in Table 5.2. Obtaining the value of f_1, f_4 and f_7 require cheap and straightforward calculations. The value of f_2, f_3, f_5 and f_6 are all dependent on hallways. As mentioned, hallways are not explicitly defined in the genotype - they must be set based on the location of the rooms. This procedure is explained in Section 5.8.1. Once the hallways are located, the value f_5, window access, and f_6, hallway area, are trivial to obtain. Computing the values for f_2, connectivity, and f_3, narrow hallways, require further complex procedures, explained below.

Table 5.2: Objective variables of the MA

	Objective variables, MA
f_1	Overlap
f_2	Connectivity
f_3	Narrow hallways
f_4	Door-neighbour distance
f_5	Window access
f_6	Hallway area
f_7	Excess neighbourhood area

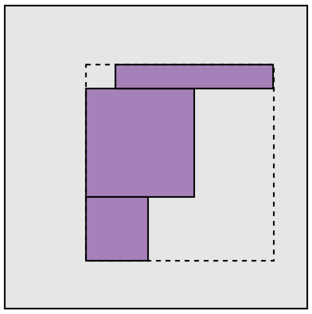
5.8.1 Locating hallways

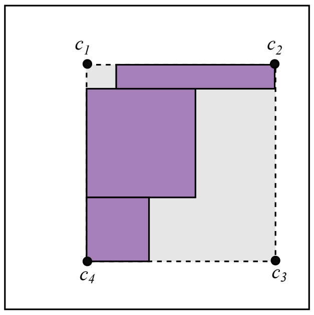
When rooms have been given a location, the hallways have to be placed. Hallway location also sets the exterior walls of the building, as they are generated by drawing a line surrounding the rooms and hallways. The process of establishing the location of hallways is divided into three phases.

Phase one is explained in Figure 5.17, where hallways are coloured grey. The starting point is illustrated in (a), where the whole site, except for the three rooms, is considered hallway. First, the hallway area is reduced to the rectangle given by the extreme points of the school layout c_1 to c_4, as shown in Figure 5.17 (b). To further reduce the hallway area, the largest vacated rectangle that can be found, starting from each corner c_i, is cut off. 5.17 (c) shows the largest and only vacated rectangle $r_{1,1}$ corresponding to corner c_1. Figure 5.17 (d) and (e) show the two rectangles $r_{3,1}$ and $r_{3,2}$ found by searching from c_3. As the area of $r_{3,2}$ is larger than $r_{3,1}$, $r_{3,2}$ is cut off. Note that there are no vacated rectangles found from c_2 and c_4. The result is shown in Figure 5.17 (f), where $r_{1,1}$ and $r_{3,2}$ are no longer part of the building. The exterior walls of the building are sketched with a solid black line.

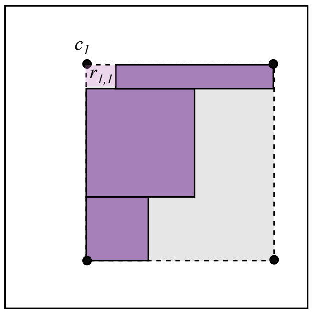
In phase two, the hallways traversed when finding a connecting path between two hallway-neighbours is fixed, such that it cannot be cut off in phase three. This ensures that connectivity is still fulfilled after cutting hallways in the next phase.

To explain phase three, a more extensive example is given in Figure 5.18. The example consists of two neighbourhoods (purple and yellow), where r_i and r_j are hallway-neighbours. Figure 5.18 (b) shows the building after phase one, where vacated rectangles are cut off from each corner. The cutting results in a narrow hallway, which is not possible to traverse. The narrow hallway is illustrated in magenta. The stapled line shows the pathway connecting r_i and r_j, which is found in phase two. The hallway area containing this line cannot be cut off in the next phase. Hallways that do not connect neighbourhoods do not help satisfy the connectivity objective and are therefore superficial. Removing them allow for more constructive hallways to appear elsewhere or reduces building costs as the building area decreases. This is the purpose of phase three.

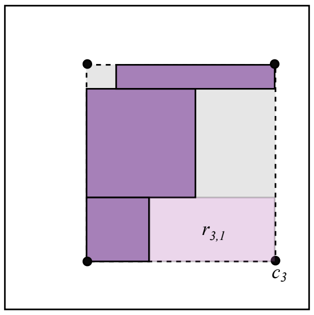

(a) Initial layout. The whole site is considered hallway.



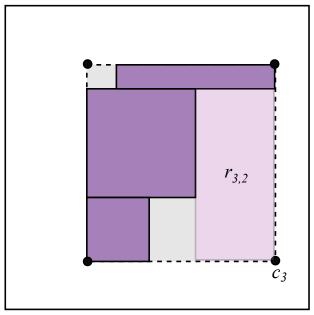
(b) The building defined by the rectangle given by the extreme points of the layout



(c) Largest (and only) vacated rectangle $r_{1,1}$ starting from c_1



(d) The first of the vacated rectangles $r_{3,1}$ starting from c_3



(e) The second of the vacated rectangles $r_{3,2}$ starting from c_3

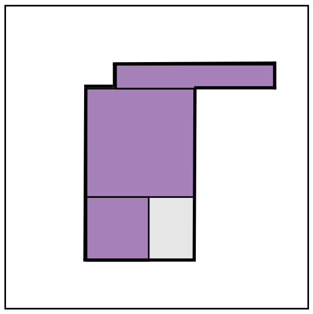
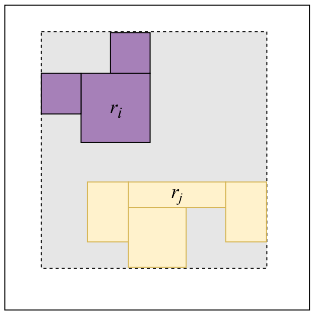

(f) The building after phase one is complete. Rectangles $r_{1,1}$ and $r_{3,2}$ are removed.

Figure 5.17: Phase one of locating hallways. Rectangular hallways are removed from four starting points, c_1 to c_4.

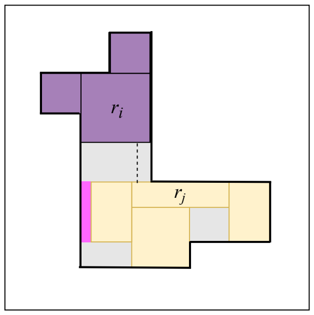
In phase three, all hallways which are not part of the hallways ensuring connectivity are cut off. Additionally, all narrow hallways with an exterior wall are removed. The final hallway area and exterior walls are shown in Figure 5.18 (*c*). As a result of phase three, connectivity is maintained, as there still exists a connecting pathway between r_i and r_j. Additionally, more rooms have window access. Thus, hallway cutting may affect the window access objective.

5.8.2 Connectivity

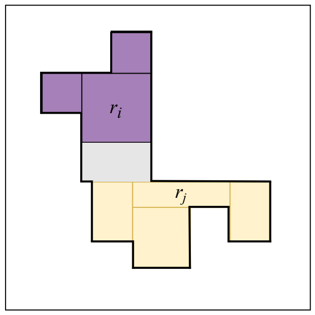
An extended Breadth-First Search (BFS) is implemented to measure the connectivity of a solution. The algorithm searches for pathways through hallways connecting hallway-neighbour pairs (r_i, r_j), where r_i and r_j are the IDs of the rooms. When performing this search, r_i is considered the source room. The search starts from the upper left corner of r_i. The algorithm only traverses cells with the value of r_i and zero, which is the ID for cells defined as hallways. If the algorithm finds a cell with the ID of r_j, it successfully found a pathway through a hallway connecting the two



(a) Layout where the building is defined by the rectangle given by the layouts extreme points



(b) Phase one complete, resulting in a narrow hallway. Dashed lines illustrate path between r_i and r_j and is found in phase two.



(c) Phase three complete. Hallways not connecting r_i and r_j and narrow hallways are cut off.

Figure 5.18: Phase three of locating hallways. Hallways not connecting hallway-neighbours are removed, in addition to narrow hallways having exterior walls.

hallway-neighbours. Furthermore, the algorithm searches for all hallway-neighbour pairs containing r_i simultaneously to improve run time. The procedure is described in Algorithm 2.

Algorithm 2: Procedure for calculating the value of the connectivity objective

```
Data: Phenotype $P$ and a list of hallway-neighbour pairs $L$
Result: Number of hallway-neighbours not connected through hallways $(n - d)$
CalculateConnectivity $(P, L)$
    $n \longleftarrow |L|$;
    $d \longleftarrow 0$;
    while $L \neq \emptyset$ do
        /* Initialize visited list                                  */
        $V \longleftarrow \emptyset$;
        $(R_1, R_2) \longleftarrow Pop(L)$;
        /* Initialize search queue                                  */
        $Q \longleftarrow \{GetCoordinate(R_1)\}$;
        while $Q \neq \emptyset$ do
            $c \longleftarrow Pop(Q)$;
            $V \longleftarrow V \cup \{c\}$;
            foreach $n \in GetNeigbhours(P, c)$ do
                if $n \notin V$ then
                    if $Value(n) == 0$ or $Value(n) == R_1$ then
                        $Q \longleftarrow Q \cup \{n\}$;
                    else if $Value(n) > 0$ then
                        if $L.contains((R_1, Value(n))$ then
                            $L.remove((R_1, Value(n)))$;
                            $d = d + 1$;
                        end
                    end
                end
            end
        end
    end
    return $n - d$
end
```

5.8.3 Narrow hallways

Narrow hallways are found by doing a check for cells defined as a hallway, that is, $a_{ij} = 0$. For these cells, the algorithm checks whether or not the cell is part of at least three consecutive hallway cells in both dimension i and j. The search strategy is equal for dimension i and j, and for simplicity, only elaborated for dimension j, which means traversing rows. A cell (i, j) defined as a hallway is changed to a narrow hallway if the following holds:

$$\{\nexists k \mid (\sum_{k=\max(j-2,0)}^{k+2} a_{i,k} == 0) \geq 3\} \quad \text{for} \quad k \in [\max(j-3,0), j] \tag{5.1}$$

The search is only performed on the extreme rows and columns of the allocated rooms, as this is the only part of the site that can contain hallways. This is done to increase the efficiency of the search. Furthermore, if the previous coordinate $(i, j-1)$ is defined as a hallway, it follows that the current coordinate (i, j) cannot be a narrow hallway. This property is exploited to determine if (i, j) is part of a narrow hallway without having to check its two closest neighbours in both directions.

5.9 Selection

Selection is the phase of choosing chromosomes for the next generation. Selection is crucial as it should ensure that the fitter individuals are kept to the next generation, while at the same time maintain the diversity in the population. This algorithm uses a combination of elitism and fitness proportionate selection (FPS). With elitism, a percentage p_e of the fittest individuals in the population always propagate to the next population. Besides the elite, the rest of the next generation is chosen based on fitness proportionate selection. In this selection method, every individual can become a parent with a probability which is proportional to its fitness. Thus, the fitter chromosomes have a higher chance of being propagated to the next generation.

Chapter 6

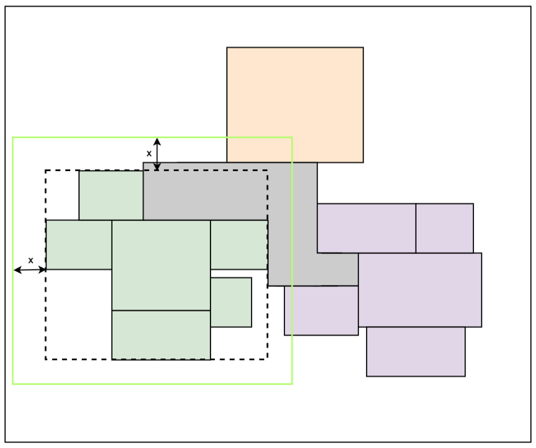
Mathematical model

In this chapter, the second stage of the algorithm is presented. Section 6.1 explains the procedure of the mathematical model, including how the output of the memetic algorithm is used to generate input for the mathematical model. The notation used for the mathematical formulation is presented in Section 6.2. Next, a basic integer linear program (ILP) is presented in Sections 6.3 - 6.4. Preliminary tests reveal that although this model finds optimal solutions in terms of minimizing corners, the computational complexity for neighbourhoods of relevant size is too high. Also, the model does not consider connectivity or window access for rooms in its output. To reduce the solution space without impacting the optimal solution, several inequalities and symmetry breaking constraints are introduced and presented in Sections 6.5 and 6.6. Additionally, problem-specific heuristics taking advantage of traits from the solution produced in stage one, are added to the mathematical formulation. These heuristics intends to reduce the complexity and let the model produce better solutions by taking advantage of the output from stage one. Section 6.7 presents the motivation behind each heuristic, as well as how they are implemented.

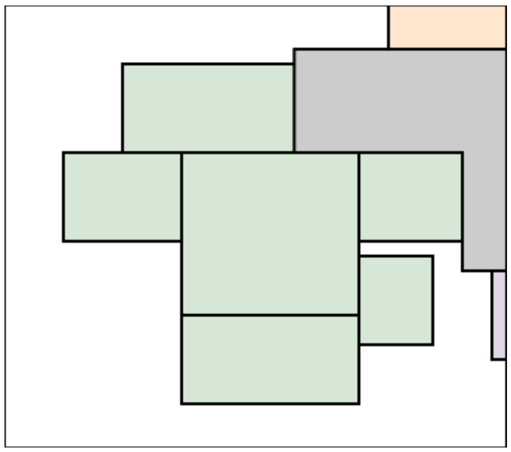
6.1 Procedure

The mathematical model is applied to each neighbourhood separately to minimize the number of corners within the neighbourhood. The neighbourhoods are iteratively optimized by adjusting the size and location of rooms from the first stage. Subsequently, the neighbourhood is inserted back into the solution used as input to the model. Each neighbourhood is enclosed by an *extended envelope* where the mathematical model is allowed to place rooms. The extended envelope is determined by taking the envelope of a neighbourhood from the solution in stage one and adding x meters in each direction. Figure 6.1 (a) illustrates the extended envelope of the neighbourhood consisting of the green rooms. The extended envelope is the rectangle formed by the solid green line. Expanding the extended envelope increases the search space, possibly resulting in a better optimal solution. However, a larger extended envelope increases the run time of the model. The size of the increase to the envelope is chosen to balance quality and complexity.

Figure 6.1 (*b*) shows the area inside the extended envelope. The area contains every room in the chosen neighbourhood and some hallway area. It can also contain rooms from other neighbourhoods, which is the case with the two purple and orange rooms in (*b*). The rooms belonging to the chosen neighbourhood can be placed anywhere inside the extended envelope, except for the areas covered by the rooms from different neighbourhoods.



(a) The light green square shows the extended envelope resulting from an x meter increase, in all directions, to the regular envelope illustrated by a dashed black line. The chosen neighbourhood consists of the green rooms. The purple and orange rooms belong to other neighbourhoods and the hallways are coloured in grey.



(b) Zooming in on the extended envelope. The green rooms, belonging to the chosen neighbourhood can be relocated, and resized, within the extended envelope.

Figure 6.1: Example of an extended envelope

When the model finds the optimal solution for a neighbourhood, the layout is updated with new positions and sizes for the rooms in the solved neighbourhood. Figure 6.2 shows the layout when the neighbourhood is updated. The procedure is repeated for the remaining neighbourhoods, except for the hub which is fixed from stage one. When the model is finished with all the neighbourhoods, hallways are regenerated. The hallways are generated using the same procedure as in stage one. The result is a complete solution which is sent to the last stage of the algorithm.

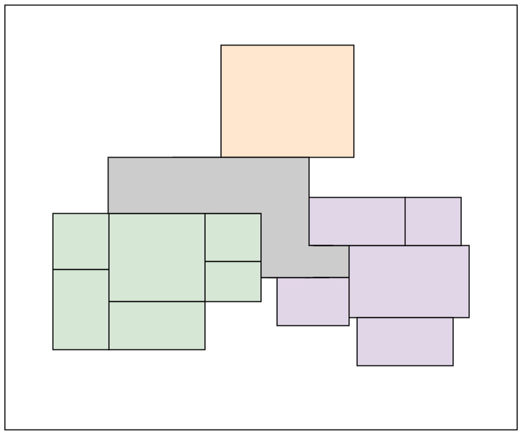


Figure 6.2: The resulting layout when the model is applied to the neighbourhood consisting of the green rooms. The number of corners in the neighbourhood is decreased from 16 to six.

6.2 Notation

Sets	
$\mathcal{E}$	coordinates (i, j) in the extended envelope
$\mathcal{D}$	directions $\{w = \text{west}, e = \text{east}, n = \text{north}, s = \text{south}\}$
$\mathcal{D}^c$	corner directions $\{nw, ne, se, sw\}$
$\mathcal{E}^d_{ij}$	coordinates in $\mathcal{E}$ in direction $d \in \mathcal{D}^c$ of coordinate (i, j)
$\mathcal{R}$	all rooms in the layout
$\mathcal{R}^n$	rooms in the chosen neighbourhood n
$\mathcal{R}^w$	rooms required to have window access, $\mathcal{R}^w \subset \mathcal{R}^n$
$\mathcal{R}^+$	rooms including other (o), $\mathcal{R}^+ = \mathcal{R} \cup \{o\}$
$\mathcal{R}^o$	rooms not included in the chosen neighbourhood, $\mathcal{R}^o = \mathcal{R} \setminus \mathcal{R}^n$

The o in the set $\mathcal{R}^+$ is a placeholder for outdoors or hallways. The hallways and outdoor area are placed at the end of stage two when the model has been applied to every neighbourhood. Thus, the mathematical model in itself does not differentiate between hallways and outdoors. The hallways are generated using the algorithm explained in Section 5.8.1.

The set $\mathcal{E}^d_{ij}$ is illustrated in Figure 6.3. Specifically, the figure considers $\mathcal{E}^{ne}_{(5,6)}$, which contains the coordinate $(5, 6)$ and all coordinates north, east and northeast of coordinate $(5, 6)$ in the extended envelope $\mathcal{E}$. The coordinates in $\mathcal{E}^{ne}_{(5,6)}$ are coloured in blue in Figure 6.3.

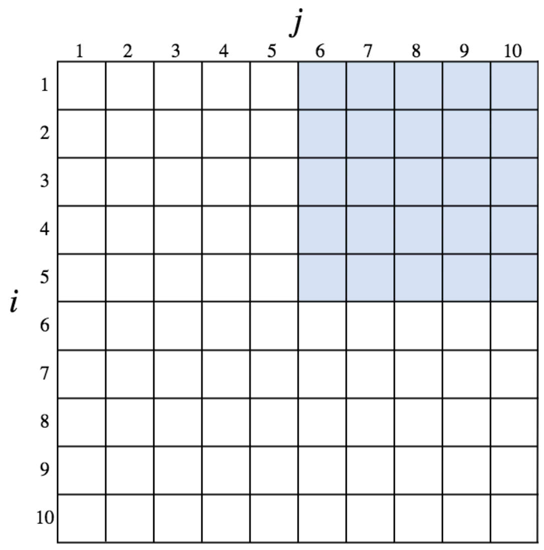


Figure 6.3: The coordinates in the set $\mathcal{E}^{ne}_{(5,6)}$ coloured in blue

Parameters

Δ_r	maximum % size deviation for room $r \in \mathcal{R}^n$
S_r	suggested size of room $r \in \mathcal{R}^n$
R_r	aspect ratio bound of room $r \in \mathcal{R}^n$
A_{rt}	adjacency matrix for $r, t \in \mathcal{R}^n$
K_{ijr}	1 if coordinate (i, j) is covered by room $r \in \mathcal{R}^o$, 0 otherwise
E^L	length of extended envelope
E^W	width of extended envelope

Variables

y_{ijr}^d	$\begin{cases} 1, & \text{if the } d \text{ corner of room } r \text{ is in } (i,j) \\ 0, & \text{otherwise} \end{cases}$	$(i,j) \in \mathcal{E}, r \in \mathcal{R}^n, d \in \mathcal{D}^c$
x_{ijr}	$\begin{cases} 1, & \text{if room } r \text{ covers coordinate } (i,j) \\ 0, & \text{otherwise} \end{cases}$	$(i,j) \in \mathcal{E}, r \in \mathcal{R}^+$
a_{ijrt}^d	$\begin{cases} 1, & \text{if the } d \text{ wall of room } r \text{ is adjacent} \\ & \text{to room } t \text{ in coordinate } (i,j) \\ 0, & \text{otherwise} \end{cases}$	$(i,j) \in \mathcal{E}, \{r,t \in \mathcal{R}^+ \mid r < t\}, d \in \mathcal{D}$
h_{ij}	number of rooms with a corner in (i,j)	$(i,j) \in \mathcal{E}$
θ_{ij}	$\begin{cases} 1, & \text{if } (i,j) \text{ is a corner} \\ 0, & \text{otherwise} \end{cases}$	$(i,j) \in \mathcal{E}$
p_{ij}	integer variable used to count corners	$(i,j) \in \mathcal{E}$

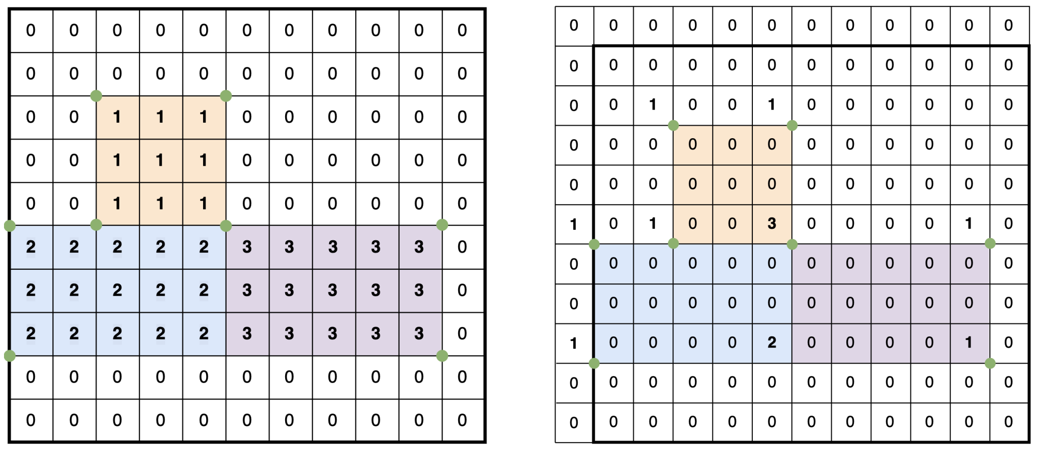
6.3 Objective function

$$\min \sum_{(i,j)\in\mathcal{E}} \theta_{ij} \tag{6.1}$$

The objective function seeks to minimize the number of corners in the neighbourhood.

6.3.1 Corner count

The variables h_{ij}, θ_{ij} and p_{ij} are introduced to count corners within the neighbourhood. θ_{ij} is a binary variable which determines whether or not the coordinate (i,j) contains a corner of the neighbourhood. A value θ_{ij} of 1 means there is a neighbourhood corner in the lower right corner of the coordinate. Integer variable h_{ij} is equal to the number of rooms with a corner adjacent to the lower right corner of the coordinate (i,j). h_{ij} is thus the sum of southeast (se) corners in (i,j), southwest (sw) corners in $(i,j+1)$, northeast (ne) corners in $(i+1,j)$ and northwest (nw) corners in $(i+1,j+1)$. Figure 6.4 (a) shows an example with the three rooms with ID 1, 2 and 3, in orange, blue and purple respectively. The rooms belong to the same neighbourhood but is illustrated with different colours for simplicity. The green dots show the corners of the neighbourhood. Figure 6.4 (b) shows the h_{ij} matrix. Observe that h_{ij} is defined over an additional row and column as h_{ij} is the sum of corners in the four coordinates (i,j), $(i+1,j)$, $(i,j+1)$ and $(i+1,j+1)$. Counting in this manner ensures that when h_{ij} is an odd number there is a corner in the lower right corner of coordinate (i, j).



(a) Grid showing the room value of each cell

(b) Grid showing the h_{ij} matrix. When h_{ij} is odd, there is a corner in the lower right corner of coordinate (i, j).

Figure 6.4: Corner count example, where the green dots display the resulting corners

6.4 Constraints

6.4.1 Corner constraints

Constraints 6.2 sets the value of h_{ij} by summing over the corner variables in coordinate (i, j) for all rooms, while constraints 6.3 ensure that $\theta_{ij} = 1$ when h_{ij} is odd and that $\theta_{ij} = 0$ otherwise.

$$h_{ij} = \sum_{r \in \mathcal{R}^n} y^{se}_{ijr} + y^{nw}_{i+1,j+1,r} + y^{ne}_{i+1,j,r} + y^{sw}_{i,j+1,r} \qquad (i,j) \in \mathcal{E} \qquad (6.2)$$

$$2p_{ij} = h_{ij} + \theta_{ij} \qquad (i,j) \in \mathcal{E} \qquad (6.3)$$

6.4.2 No overlap between rooms

$$\sum_{r \in \mathcal{R}^+} x_{ijr} + \sum_{r \in R^o} K_{ijr} = 1 \qquad (i,j) \in \mathcal{E} \qquad (6.4)$$

Constraints 6.4 ensure that every coordinate (i, j) is either outdoors or covered by a hallway or a single room.

6.4.3 Room and corner consistency

A room can only cover coordinate (i, j) if the coordinate is enclosed by the four corners of the room, this is ensured by constraints 6.5. Additionally, if all four corners of a room enclose a coordinate, the room must cover that coordinate, which is ensured by constraints 6.6.

$$x_{ijr} \leq \sum_{(k,l)\in\mathcal{E}^d_{ij}} y^d_{klr} \qquad (i,j)\in\mathcal{E}, r\in\mathcal{R}^n, d\in\mathcal{D}^c \tag{6.5}$$

$$x_{ijr} \geq \sum_{d\in\mathcal{D}^c} \sum_{(k,l)\in\mathcal{E}^d_{ij}} y^d_{klr} - 3 \qquad (i,j)\in\mathcal{E}, r\in\mathcal{R}^n \tag{6.6}$$

6.4.4 Room shapes and sizes

Separately, constraints 6.7 - 6.10 ensure that walls are horizontal or vertical in relation to the building site. Collectively, they make sure all rooms are rectangular.

$$\sum_{(i,j)\in\mathcal{E}} j y^{nw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} j y^{sw}_{ijr} = 0 \qquad r\in\mathcal{R}^n \tag{6.7}$$

$$\sum_{(i,j)\in\mathcal{E}} j y^{ne}_{ijr} - \sum_{(i,j)\in\mathcal{E}} j y^{se}_{ijr} = 0 \qquad r\in\mathcal{R}^n \tag{6.8}$$

$$\sum_{(i,j)\in\mathcal{E}} i y^{nw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} i y^{ne}_{ijr} = 0 \qquad r\in\mathcal{R}^n \tag{6.9}$$

$$\sum_{(i,j)\in\mathcal{E}} i y^{sw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} i y^{se}_{ijr} = 0 \qquad r\in\mathcal{R}^n \tag{6.10}$$

Constraints 6.11 and 6.12 make sure that the size of room r is between its upper and lower limit, respectively. Constraints 6.13 and 6.14 enforce a room to have an aspect ratio within its bound.

$$\sum_{(i,j)\in\mathcal{E}} x_{ijr} \leq \Big\lfloor (1+\Delta_r)S_r \Big\rfloor \qquad r \in \mathcal{R}^n \tag{6.11}$$

$$\sum_{(i,j)\in\mathcal{E}} x_{ijr} \geq \Big\lceil (1-\Delta_r)S_r \Big\rceil \qquad r \in \mathcal{R}^n \tag{6.12}$$

$$\sum_{(i,j)\in\mathcal{E}} (y^{ne}_{ijr} - y^{nw}_{ijr}) \cdot j \cdot R_r \geq \sum_{(i,j)\in\mathcal{E}} (y^{se}_{ijr} - y^{ne}_{ijr}) \cdot i \qquad r \in \mathcal{R}^n \tag{6.13}$$

$$\sum_{(i,j)\in\mathcal{E}} (y^{se}_{ijr} - y^{ne}_{ijr}) \cdot i \cdot R_r \geq \sum_{(i,j)\in\mathcal{E}} (y^{ne}_{ijr} - y^{nw}_{ijr}) \cdot j \qquad r \in \mathcal{R}^n \tag{6.14}$$

6.4.5 Adjacency

Constraints 6.15 - 6.19 ensure that the rooms r and t are adjacent when $A_{rt} = 1$, namely when they are required to be door-neighbours. Constraints 6.19 ensure that the rooms share a wall of at least two meters, which is a requirement for two rooms to be attached. Given the rectangular shape of the rooms, adjacency is only possible on one side at a time.

$$2a^{n}_{ijrt} \leq x_{ijr} + x_{i-1,jt} \qquad r,t \in \mathcal{R}^+, (i,j) \in \mathcal{E} \tag{6.15}$$

$$2a^{s}_{ijrt} \leq x_{ijr} + x_{i+1,jt} \qquad r,t \in \mathcal{R}^+, (i,j) \in \mathcal{E} \tag{6.16}$$

$$2a^{w}_{ijrt} \leq x_{ijr} + x_{ij-1,t} \qquad r,t \in \mathcal{R}^+, (i,j) \in \mathcal{E} \tag{6.17}$$

$$2a^{e}_{ijrt} \leq x_{ijr} + x_{ij+1,t} \qquad r,t \in \mathcal{R}^+, (i,j) \in \mathcal{E} \tag{6.18}$$

$$2A_{rt} \leq \sum_{(i,j)\in\mathcal{E}} a^{n}_{ijrt} + a^{s}_{ijrt} + a^{e}_{ijrt} + a^{w}_{ijrt} \qquad r,t \in R^+ \tag{6.19}$$

6.4.6 Variable declarations

The following constraints declare the decision variables of the mathematical model.

$$y^{d}_{ijr} \in \{0,1\} \tag{6.20}$$

$$x_{ijr} \in \{0,1\} \tag{6.21}$$

$$a^{d}_{ijrt} \in \{0,1\} \tag{6.22}$$

$$\theta_{ij} \in \{0,1\} \tag{6.23}$$

$$h_{ij} \in \mathbb{Z} \tag{6.24}$$

$$p_{ij} \in \mathbb{Z} \tag{6.25}$$

6.5 Valid inequalities

A valid inequality for an IP (or MILP) is any constraint that does not eliminate any feasible integer solutions. The following inequalities are implemented to improve run time.

Constraints 6.26 make sure a room has one corner in all four directions.

$$\sum_{(i,j)\in\mathcal{E}} y^d_{ijr} = 1 \qquad r \in \mathcal{R}^n, d \in \mathcal{D}^c \tag{6.26}$$

Constraints 6.27 force a room to be located in every coordinate where the room has a corner.

$$x_{ijr} \geq y^d_{ijr} \qquad (i,j) \in \mathcal{E}, r \in \mathcal{R}^n, d \in \mathcal{D}^c \tag{6.27}$$

If a coordinate is enclosed by the four corners of room r, then the corner cannot be covered by a hallway, outdoor or a different room t. This is ensured by constraints 6.28.

$$x_{ijt} + \sum_{d\in\mathcal{D}^c} \sum_{(k,l)\in\mathcal{E}^d_{ij}} y^d_{klr} \leq 4 \qquad (i,j) \in \mathcal{E}, r \in \mathcal{R}^n, t \in \mathcal{R}^+ \setminus r \tag{6.28}$$

The absolute minimum number of corners in a neighbourhood is four, as it contains solely rectangular rooms. Constraint 6.29 sets the lower bound of the number of corners.

$$\sum_{(i,j)\in\mathcal{E}} \theta_{ij} \geq 4 \tag{6.29}$$

Constraints 6.30 - 6.36 make sure the length of the walls is within its lower and upper limit. The limit is given by the size and aspect ratio bound of the room.

$$\sum_{(i,j)\in\mathcal{E}} jy^{nw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} jy^{ne}_{ijr} + \sqrt{S_r(1-\Delta_r)/R_r} \leq 0 \qquad r \in \mathcal{R}^n \tag{6.30}$$

$$\sum_{(i,j)\in\mathcal{E}} iy^{nw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} iy^{sw}_{ijr} + \sqrt{S_r(1-\Delta_r)/R_r} \leq 0 \qquad r \in \mathcal{R}^n \tag{6.31}$$

$$\sum_{(i,j)\in\mathcal{E}} jy^{sw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} jy^{se}_{ijr} + \sqrt{S_r(1-\Delta_r)/R_r} \leq 0 \qquad r \in \mathcal{R}^n \tag{6.32}$$

$$\sum_{(i,j)\in\mathcal{E}} iy^{ne}_{ijr} - \sum_{(i,j)\in\mathcal{E}} iy^{se}_{ijr} + \sqrt{S_r(1-\Delta_r)/R_r} \leq 0 \qquad r \in \mathcal{R}^n \tag{6.33}$$

$$\sum_{(i,j)\in\mathcal{E}} jy^{nw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} jy^{ne}_{ijr} + \sqrt{R_r S_r(1+\Delta_r)} \geq 0 \qquad r \in \mathcal{R}^n \tag{6.34}$$

$$\sum_{(i,j)\in\mathcal{E}} iy^{nw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} iy^{sw}_{ijr} + \sqrt{R_r S_r(1+\Delta_r)} \geq 0 \qquad r \in \mathcal{R}^n \tag{6.35}$$

$$\sum_{(i,j)\in\mathcal{E}} jy^{sw}_{ijr} - \sum_{(i,j)\in\mathcal{E}} jy^{se}_{ijr} + \sqrt{R_r S_r(1+\Delta_r)} \geq 0 \qquad r \in \mathcal{R}^n \tag{6.36}$$

$$\sum_{(i,j)\in\mathcal{E}} iy^{ne}_{ijr} - \sum_{(i,j)\in\mathcal{E}} iy^{se}_{ijr} + \sqrt{R_r S_r(1+\Delta_r)} \geq 0 \qquad r \in \mathcal{R}^n \tag{6.37}$$

6.6 Symmetry breaking constraints

Several neighbourhoods have two or more identical rooms in terms of size and function, thus creating symmetrical solutions. To break these symmetries, the sets $\mathcal{T}$, representing room types, and $\mathcal{R}^t$ ($t \in \mathcal{T}$), which represents rooms with room type t, is introduced. Only rooms that are identical in the RSD are considered the same room type. To break symmetry, a room p, identical to room r, is forced to be placed either below or to the right of r. The two binary variables β^v_{rp} and β^h_{rp} and constraints 6.38 - 6.40 are added to ensure this.

$$\sum_{(i,j)\in\mathcal{E}} y^{nw}_{ijr} i \leq \sum_{(i,j)\in\mathcal{E}} y^{nw}_{ijp} i + E^L \beta^v_{rp} \qquad r, s \in \mathcal{R}^t, t \in \mathcal{T}, r < p \tag{6.38}$$

$$\sum_{(i,j)\in\mathcal{E}} y^{nw}_{ijr} j \leq \sum_{(i,j)\in\mathcal{E}} y^{nw}_{ijp} j + E^W \beta^h_{rp} \qquad r, p \in \mathcal{R}^t, t \in \mathcal{T}, r < p \tag{6.39}$$

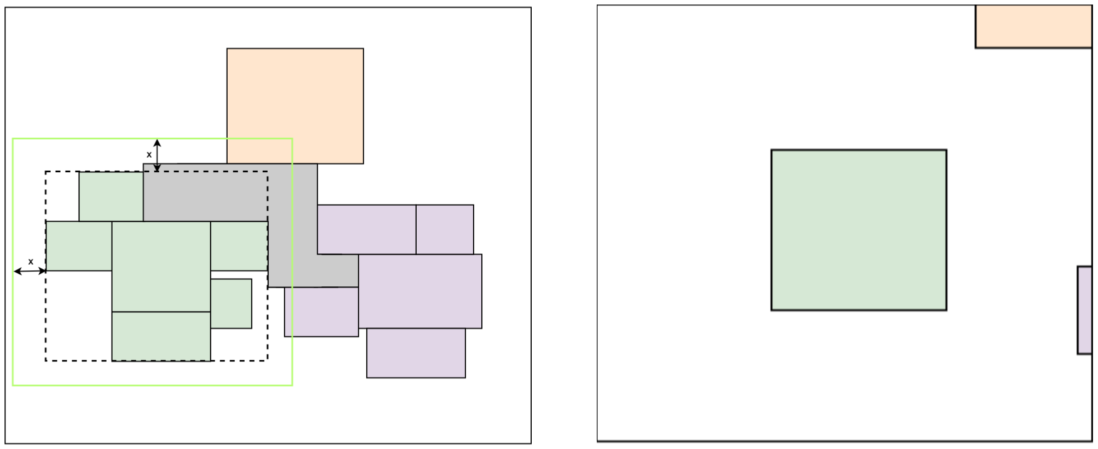
$$\beta^h_{rs} + \beta^v_{rs} \leq 1 \qquad r, p \in \mathcal{R}^t, t \in \mathcal{T}, r < p \tag{6.40}$$

6.7 Modelling heuristics

The model described in Sections 6.2 - 6.6 does not take full advantage of the solution it receives as input from stage one. The problem-specific heuristics presented in Sections 6.7.1 - 6.7.5 attempt to improve the solution quality or run time of the mathematical model.

6.7.1 Lock main room

Every neighbourhood has a main room which is required to be a door-neighbour with all the other rooms. Adjusting the main room therefore affects the feasible positions of all the other rooms in the neighbourhood. Thus, locking the main room to its location in the output from the MA will drastically reduce the run time. A large drawback of this heuristic is that it removes many good, potentially optimal, solutions. Still, as the extended envelope area is created based on the placement of the rooms from stage one, keeping the main room fixed gives it a central location within the area. This allows the other rooms to be adjacent on either side. A very good, or even optimal, solution is therefore likely to be found with this positioning of the main room. Figure 6.5 (a) once again illustrates a green neighbourhood with its extended envelope. The main room of the green neighbourhood is the room located near the centre of the extended envelope. Figure 6.5 (b) shows what is fixed inside the extended envelope of the neighbourhood with this modelling heuristic. This now includes both the main room and the surrounding rooms from other neighbourhoods. The remaining rooms of the green neighbourhood are allowed to cover the white area in the extended envelope.



(a) The extended envelope is determined the same way as before

(b) The area covered by the main room is now fixed. As earlier, the rooms from other neighbourhoods (here purple and orange) are also fixed.

Figure 6.5: The extended envelope is input to the model with locked coordinates for the main room

To lock the main room m in the model, the set $\mathcal{M}$ is introduced. This set contains the grid coordinates the main room covers given its position in the solution from stage one. Constraints 6.41 prevents other rooms, hallways or outdoor to cover these coordinates.

$$x_{ijr} = 0 \qquad (i,j) \in \mathcal{M}, r \in \mathcal{R}^+ \setminus m \tag{6.41}$$

For performance enhancements is x_{ijr} for $(i,j) \in \mathcal{M}$ not defined in the implementation of the model when the main room is locked. Additionally, x_{ijm} for $(i,j) \in \mathcal{E}$ is not defined at all. However, constraints 6.41 is included for simplicity of the mathematical formulation.

Locking the main room allows for significantly more efficient adjacency constraints. The coordinates of the walls of the main room are placed in the sets $\mathcal{M}^d$ where $d \in \mathcal{D}$. Constraints 6.42 ensure that every room shares one of its walls with the main room.

$$\begin{aligned}(\sum_{(i,j)\in\mathcal{M}^n} x_{i-1,jr} + \sum_{(i,j)\in\mathcal{M}^s} x_{i+1,jr} +\\ \sum_{(i,j)\in\mathcal{M}^e} x_{ij+1,r} + \sum_{(i,j)\in\mathcal{M}^w} x_{ij-1,r}) \geq 1 \qquad r \in \mathcal{R}^n\end{aligned} \tag{6.42}$$

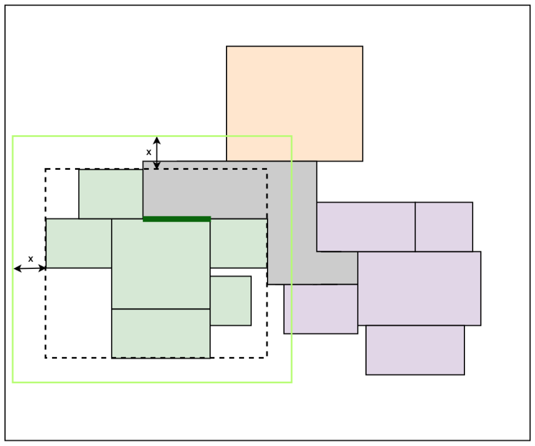
This implementation of adjacency allows for variables a^d_{ijrs} to be excluded from the model, drastically improving run time. Lastly, locking the main room has the additional advantage of allowing the heuristics in Sections 6.7.2 - 6.7.3 to be implemented.

6.7.2 Lock hallways

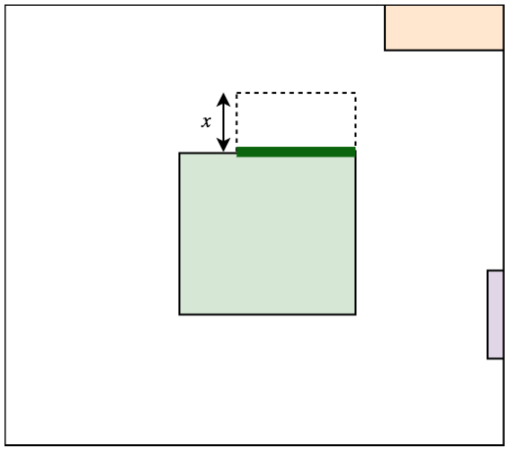
To ensure connectivity for the school as a whole, all main rooms are required to be hallway-neighbour with the hub. Fixing the main room makes it possible to use a heuristic to keep the hallway structure from the solution in stage one. Placing hallways is intricate and requires considering the school layout as a whole, as explained in Section 5.8.1. The model only considers the area inside the extended envelope for one neighbourhood at a time. It is therefore inherently unsuited to locate hallways.

A pre-processing step is added to maintain the connectivity in stage two. The step is illustrated in Figure 6.6. First, a search algorithm finds the longest consecutive stretch of coordinates within the main room that is adjacent to a hallway. A dark green solid line illustrates this stretch in (a). Then a set $\mathcal{H}$ is defined. This set contains the coordinates within the area extending x meters out of the main room from the stretch. The area is illustrated by the dotted black rectangle in 6.6 (b). Constraints 6.43 are added to make sure at least four coordinates in $\mathcal{H}$ are reserved for hallways. This prevents rooms from occupying the coordinates, making it possible for the hallway generator to place hallways in this area.

$$\sum_{(i,j)\in\mathcal{H}} x_{ijo} \geq 4 \qquad\qquad (i,j)\in\mathcal{E} \tag{6.43}$$



(a) The extended envelope is set in the same way as before. The longest consecutive stretch of hallway-adjacent coordinates is illustrated by a dark green solid line.



(b) The main room is fixed, and the neighbourhood is forced to have a minimum amount of reserved area for hallways within the dotted rectangle area, extending x meters out from the main room.

Figure 6.6: Extended envelope with a locked main room and a pre-determined area where there must exist a minimum amount of coordinates reserved for hallways

6.7.3 Window access heuristic

As the hallways are regenerated after the model is applied to all neighbourhoods, the model cannot distinguish between outdoors and hallways. Consequently, it does not consider the window access objective. However, preliminary testing shows that the lack of windows access is a recurring issue in the solutions generated by stage two. Hence, a heuristic approach to handle the window requirements is developed.

Through a change in the objective function, rooms in $\mathcal{R}^w$ are incentivized to be located where they have the largest chance of being adjacent to outdoor area. The solutions generated in stage one show that rooms located on the far side of the main room, in relation to the hub, are most likely to have window access. In the preprocessing phase, calculations are performed to discover which side of the main room is furthest from the hub. By using this as input to the model, the objective function is adjusted to reward solutions which place rooms further from the hub. To implement the heuristic the parameters P^d, where $d \in \mathcal{D}$, are introduced. If the calculations reveal that placing rooms on the d side of the main room is desirable, P^d is set to 1, while P^t for $\forall t \in \mathcal{D}$ where $t \neq d$, is set to 0.

To ensure that the objective of minimizing corners is prioritized, the coefficient C^c is introduced. The objective function, with the window heuristic implemented, is shown in 6.44

$$\begin{aligned} \min \quad & \Big(\sum_{(i,j)\in\mathcal{E}} C^c \theta_{ij} + \\ & \sum_{r\in\mathcal{R}^w} \sum_{(i,j)\in\mathcal{E}} \big(P^n i y^{se}_{ijr} + P^s (E^L - i) y^{ne}_{ijr} + P^w j y^{ne}_{ijr} + P^e (E^W - j) y^{nw}_{ijr} \big) \Big) \end{aligned} \tag{6.44}$$

6.7.4 Concurrent neighbourhood optimization

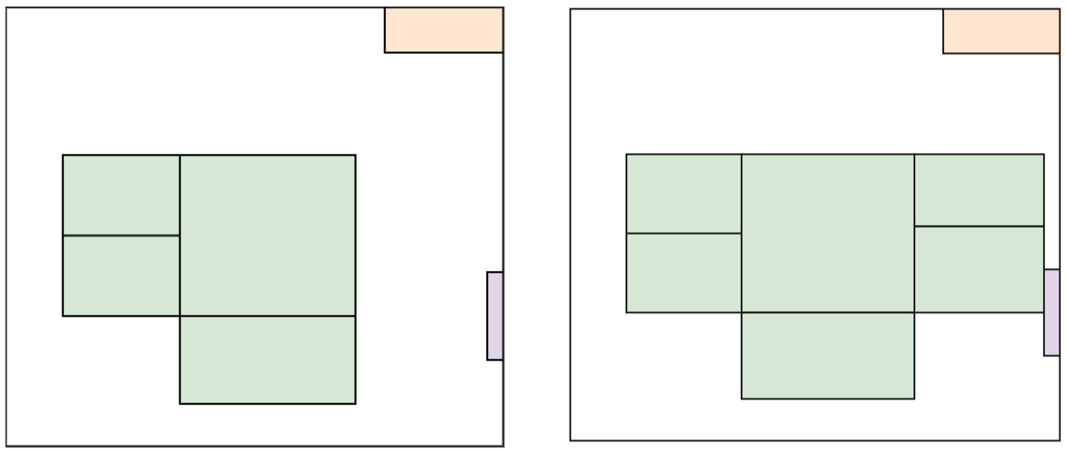
An alternative implementation of the model is to optimize neighbourhoods concurrently. This allows for a more efficient use of computational power, thus reducing the run time. With concurrent neighbourhood optimization, the total run time of the model is decided by the run time of the most complex neighbourhood in the input. Because the area of two extended envelopes sometimes overlap, this can cause rooms from different neighbourhoods to overlap in the output from stage two. This must be handled in stage three to ensure feasible solutions.

6.7.5 Split neighbourhood

The number of rooms in a neighbourhood heavily impacts the run time of the mathematical model. An alternative approach to address this is to run the model for a subset of the rooms, iteratively placing all rooms in the neighbourhood. The procedure works as follows. First, the rooms in the neighbourhood $\mathcal{R}^n$ are partitioned into P subsets $\mathcal{R}^p$ where $p \in \{1, .., P\}$, such that each room is included in only one subset $\mathcal{R}^P$. The model then iterates the subsets and places the rooms. Rooms requiring window access are placed in the first subset. When optimizing for $\mathcal{R}^t$ where $1 < t \leq P$, the rooms in the sets $\mathcal{R}^p$, where $p \in \{1, .., t-1\}$, are locked in place. To lock coordinates that are covered by rooms placed in previous iterations is the set $\mathcal{M}^f$ containing these coordinates introduced. Constraints 6.45 ensure that rooms cannot cover coordinates in $\mathcal{M}^f$.

$$x_{ijr} = 0 \qquad\qquad (i,j) \in \mathcal{M}^f, r \in \mathcal{R}^p \tag{6.45}$$

Figure 6.7 illustrates the procedure considering a neighbourhood with six rooms partitioned into two subsets. The green rooms are subject to optimization, while the purple and orange rooms are locked in place as they belong to different neighbourhoods. Figure 6.7 (a) shows the first iteration where four rooms are placed. (b) shows the final iteration where the remaining rooms are placed while considering the placement of the rooms in the previous iteration.



(a) Four rooms, including the main room (middle) are placed by the model in the first run

(b) The second run places the last two rooms

Figure 6.7: The green neighbourhood with six rooms is optimized in two consecutive model runs

By splitting the neighbourhoods, the model optimizes for the given rooms without considering the placement of the rooms in following iterations. Consequently, there is a trade-off between improving run time and quality of solutions when applying this alternative approach.

Chapter 7

Local search

The third and final stage of the three-stage algorithm is a local search. The local search is performed in turn for each neighbourhood, where the order is chosen based on a *selection approach*. The procedure is described in Section 7.1, while Section 7.2 describes three different alternatives for selecting the order of the neighbourhoods.

7.1 Algorithm

The local search moves a neighbourhood to the best position within a square search area. The square is set by adding l meters in each direction from the upper left corner of the envelope of the neighbourhood. Figure 7.2 (a) illustrates the search area, where the solid black line forms the square. The l meters is further referred to as the length of the search area. Each position within the area is scored using a weighted sum of several objectives. The local search incorporates a subset of the MA objectives; overlap (f_1), connectivity (f_2), narrow hallways (f_3), window access (f_5) and hallway area (f_6). The local search also includes exterior corners as an objective, subject to minimization, denoted f_8. The objective variables of the LS is summarized in Table 7.1. Figure 7.1 shows an example with the three rooms 1, 2 and 3, in orange, blue and purple respectively. The green dots show the exterior corners of the building. Each dot is an intersection of four cells. If an odd number of these cells have an outdoor ID of 0, the intersection is an exterior corner.

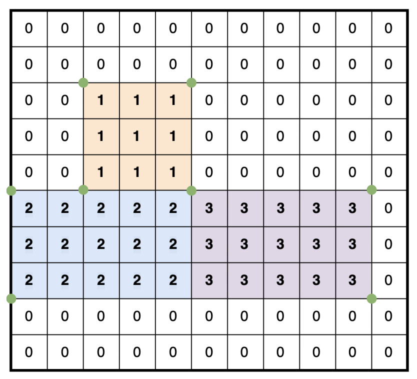


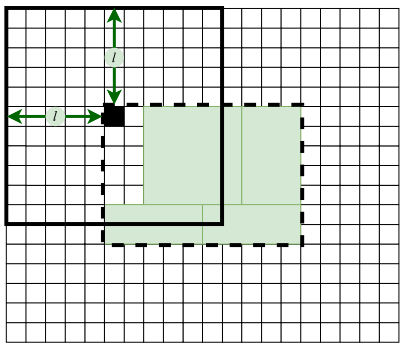
Figure 7.1: Grid showing the room value of each cell for a simple layout with three rooms. The green dots display the resulting exterior corners.

Table 7.1: Objective variables of the LS

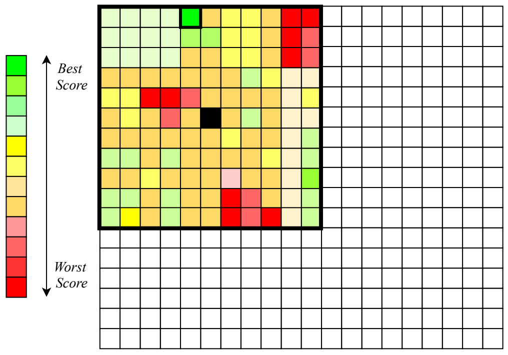
Objective variables, LS	
f_1	Overlap
f_2	Connectivity
f_3	Narrow hallways
f_5	Window access
f_6	Hallway area
f_8	Exterior corners

Locations that result in narrow hallways, overlap, or violate window access or connectivity requirements are strongly penalized during the evaluation. Consequently, the LS attempts to minimize these objectives to the optimal value of zero while seeking to minimize the two remaining objectives; hallway area (f_6) and exterior corners (f_8). Minimizing exterior corners is given the highest weight among the two, as it is the primary goal of the LS.

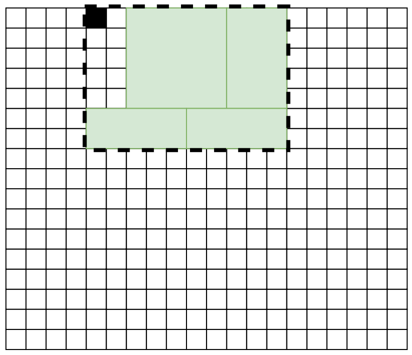
Figure 7.2 illustrates the local search of a neighbourhood. Figure 7.2 (a) shows a neighbourhood with its envelope and search area. The black filled cell in the upper left corner is the reference point of the neighbourhood. The solid black line forms a square which marks the area in which the reference point can be moved; in this case 5 meters, or cells, in each direction, as $l = 5$. Figure 7.2 (b) illustrates the scoring of the possible locations, where red and green are the worst and best score, respectively. The red cells are typically locations resulting in overlap or connectivity violation. The green square marked by a solid black line is the best location based on the scoring function. Thus, the neighbourhood moves to this position in (c).



(a) Initial position of a neighbourhood. The dashed black line is the envelope of the neighbourhood, while the black solid line forming a square is the search area with length l.



(b) Scoring of possible new locations within the search area. The green square marked by a black solid line is scored as the best location.



(c) Result after neighbourhood moved to the new and locally best location

Figure 7.2: The local search of a neighbourhood. The neighbourhoods moves to the best location within a square search area.

The local search is performed in turn for each neighbourhood, where the order is based on a selection approach. Three different approaches are implemented and described in the subsequent section. As the movement of one neighbourhood can affect the outcome of other neighbourhoods, the procedure repeats until no beneficial moves are present. The algorithm is described in Algorithm 3.

Algorithm 3: Local search of neighbourhoods

```
Data: Set of neighbourhoods $N$, Layout $L$, search area length $l$
Result: Updated layout $L$ with new locations of neighbourhoods
LocalSearch $(N, L, l)$
    $Improvement \longleftarrow True$;
    while $Improvement$ do
        $K \longleftarrow N$;
        $Improvement \longleftarrow False$;
        for $i = 1 \longrightarrow |N|$ do
            /* Pick neighbourhood by criteria                    */
            $n \longleftarrow SelectionCriteria(K)$;
            $K.remove(n)$;
            /* Local search neighbourhood                        */
            $LocalSearch(n, l, L)$;
            if $n$ hasMoved then
                $Improvement \longleftarrow True$
            end
        end
    end
    return $L$
end
```

7.2 Selection approaches

Different approaches for choosing the order of neighbourhoods are described in the following. Three approaches are developed, a random order approach, and two heuristics. The first heuristic chooses the largest neighbourhood from the set of remaining neighbourhoods K. The hypothesis is that the largest neighbourhoods are most decisive of the fitness of the solutions and should be considered first.

The second heuristic intends to fix layouts which are either infeasible, meaning f_1 or f_2 is not zero, or fail to fulfil the window access objective (f_5). The algorithm is described in Algorithm 4. This heuristic chooses the neighbourhoods that break feasibility or the window requirements first. It prioritizes feasibility over window access, and overlap over connectivity. Thus, if a layout contains overlap, the neighbourhoods containing the overlapping rooms are considered first, and the order among them are chosen at random. Next up are neighbourhoods violating connectivity and lastly neighbourhoods containing rooms without window access. If the layout is feasible and satisfy the window objective, a random order approach is applied.

Algorithm 4: Selection Approach 2: Prioritize neighbourhoods by objectives

```
Data: Set of neighbourhoods $K$, Layout $L$
Result: Chosen neighbourhoood $n$
PrioritizeObjectives ($K, L$)
    if $f_1 \neq 0$ then
        $N_o \longleftarrow getOverlappingNeighbourhoods(L)$;
        $n \longleftarrow Random(N_o)$;
        return $n$
    end
    else if $f_2 \neq 0$ then
        $N_c \longleftarrow getUnconnectedNeighbourhoods(L)$;
        $n \longleftarrow Random(N_c)$;
        return $n$
    end
    else if $f_5 \neq 0$ then
        $N_w \longleftarrow getNeighbourhoodsWithoutWindowAccess(L)$;
        $n \longleftarrow Random(N_w)$;
        return $n$
    end
    $n \longleftarrow Random(K)$;
    return $n$
end
```

Chapter 8

Case description

This thesis uses the RSD of Levanger Middle School as input to the three-stage algorithm. The RSD is a private document provided by our industry partner, Spacemaker. Since most RSDs consist of the same information and contain similar requirements, a single RSD is chosen as the starting point for this study without much loss of generality. Levanger Middle School was built in 2015 and accommodates 500 students. The school contains approximately 120 rooms divided into 20 neighbourhoods, where the rooms make up 4412 square meters in total. Today, the school constitutes 6200 square meters over two floors. This includes hallways and vertical transportation in addition to rooms. The first floor consists of neighbourhoods such as the administration, sciences and the music area. The second floor mainly consists of the classrooms and their corresponding study rooms. This thesis does not focus on the actual layout of Levanger Middle School, as it is only one out of many possible layout suggestions coherent with the RSD. Instead, parts of the RSD is used as input for generating layouts. This includes the list of neighbourhoods and rooms, and the specifications that come with each room.

Six RSDs are created for testing purposes, where each contains a subset of the neighbourhoods described in the RSD of Levanger Middle School. As this thesis considers a single floor SLP, some of the 20 neighbourhoods are excluded from the different RSDs. The excluded neighbourhoods have less critical functions, such as storage for cleaning equipment. The test RSDs are chosen to reflect the various compositions of neighbourhoods a floor can contain, and with different degrees of complexity to test the abilities of the three-stage algorithm. To preserve the characteristics of the SLP, the neighbourhoods selected for each RSD contain rooms with different sizes and aspect ratio bounds. For the rooms in the RSD of Levanger Middle School without a specified aspect ratio bound, a suitable one is chosen - most rooms are assigned a bound of either 1, 2 or 3. The main rooms, which are typically bigger and have many door-neighbours, are assigned an aspect ratio bound of 3, 4 or 5, depending on its size. All the RSDs contain the hub, as it is considered the centre of the school and contains several essential functions.

Table 8.1 shows the number and total area of the rooms in each RSD. The column "RSD" states the name of each RSD, which is the number of neighbourhoods it contains in addition to the hub. Thus, RSD $2N$ contains two neighbourhoods and the hub. Table A.1 in Appendix A displays the neighbourhoods each RSD contains. Together these six RSDs consist of 15 unique neighbourhoods. Table 10.2 presents the key characteristics of the neighbourhoods. The table shows the number of rooms and their total area, along with a colour used for visualization purposes throughout the technical studies. For a complete specification of the neighbourhoods and their corresponding rooms, see Table A.2 in Appendix A.

Table 8.1: The six RSDs used for testing, with the number and total area of rooms

RSD	Number of rooms	Total area (m^2)
$2N$	16	1610
$3N$	21	1705
$6N$	27	2061
$7N$	21	1835
$9N$	41	2795
$11N$	50	3230

Table 8.2: Number of rooms and total area of each neighbourhood. The colour is used for visualizations of the neighbourhoods.

	Neighbourhoods		
Neighbourhood	Number of rooms	Total area (m^2)	Colour
Music Area	4	231	🟪
9th grade offices	5	140	🟧
Gym	4	200	🟪
Arts & Crafts	3	270	🟦
9th grade	7	540	🟩
Cooking	3	180	🟥
10th grade offices	5	140	🟩
10th grade	7	560	🟧
Science	3	160	🟦
Library	2	144	🟥
Administration	3	200	🟥
Employees wardrobes	2	150	🟩
ICT	6	165	🟨
8th grade	7	500	🟪
Hub	1	500	🟫

Figure 8.1 illustrates a possible layout generated from RSD $6N$. The RSD contains six neighbourhoods in addition to the hub, consisting of 27 rooms in total. The exterior walls of the school building are marked by a solid black line surrounding the rooms and hallways. The coloured rectangles are various rooms, and the light grey areas are hallways. Rooms within the same neighbourhood are illustrated with the same colour. These are the colours specified in Table 10.2 - e.g. rooms in the Arts &

Crafts neighbourhood are coloured in blue. Additionally, all rooms have a dark grey wall colour.

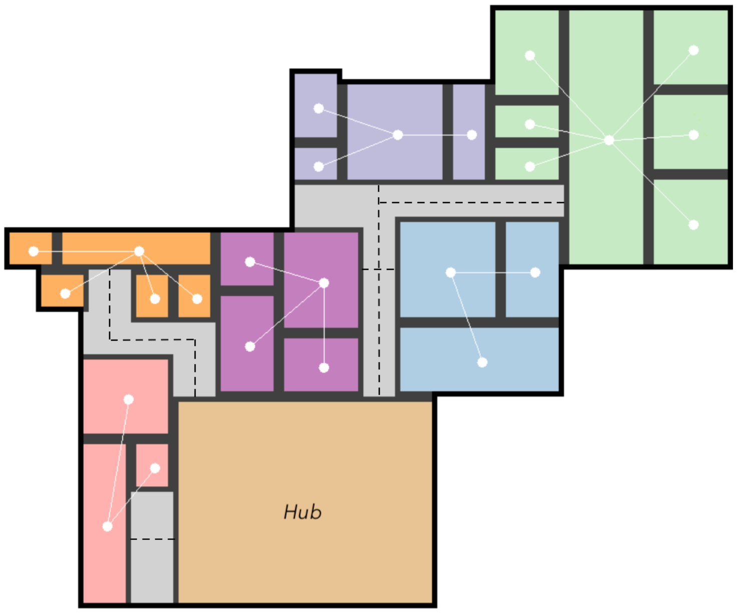


Figure 8.1: A generated school layout using RSD 6N as input, with seven neighbourhoods and 27 rooms

A graph shows the door-neighbour relationships, where each room contains a white node in the centre and white edges connecting the room to its door-neighbours. The hub is a neighbourhood with only one room. Since it has no door-neighbours, it has no white node. As specified in Section 4.4.2, all neighbourhoods have a main room, and all rooms need to be attached to the main room of its neighbourhood. Thus, the main room of each neighbourhood contains the root node of a graph in the layout, while the other rooms contain leaf nodes. The layout in Figure 8.1 fulfils the door-neighbour relationships, as all rooms are attached to the main room of its neighbourhood.

The main room of each neighbourhood is required to be a hallway-neighbour with the hub. Thus, the hub of the layout in Figure 8.1 has six hallway-neighbours, one main room for each neighbourhood. The dashed black lines show the pathways from the hub to its six hallway-neighbours. As there is a direct pathway through a hallway from all main rooms to the hub, the layout fulfils the connectivity objective.

Figure 8.2 shows the blueprint of the layout visualized in Figure 8.1. Possible locations for doors are manually inserted, and the thick solid grey line on the east wall of the hub shows the suggested building entrance. As Figure 8.1 is more informative, this design is used as the visualization tool for the technical studies, while the blueprint design is used when presenting the final layouts in Chapter 12.

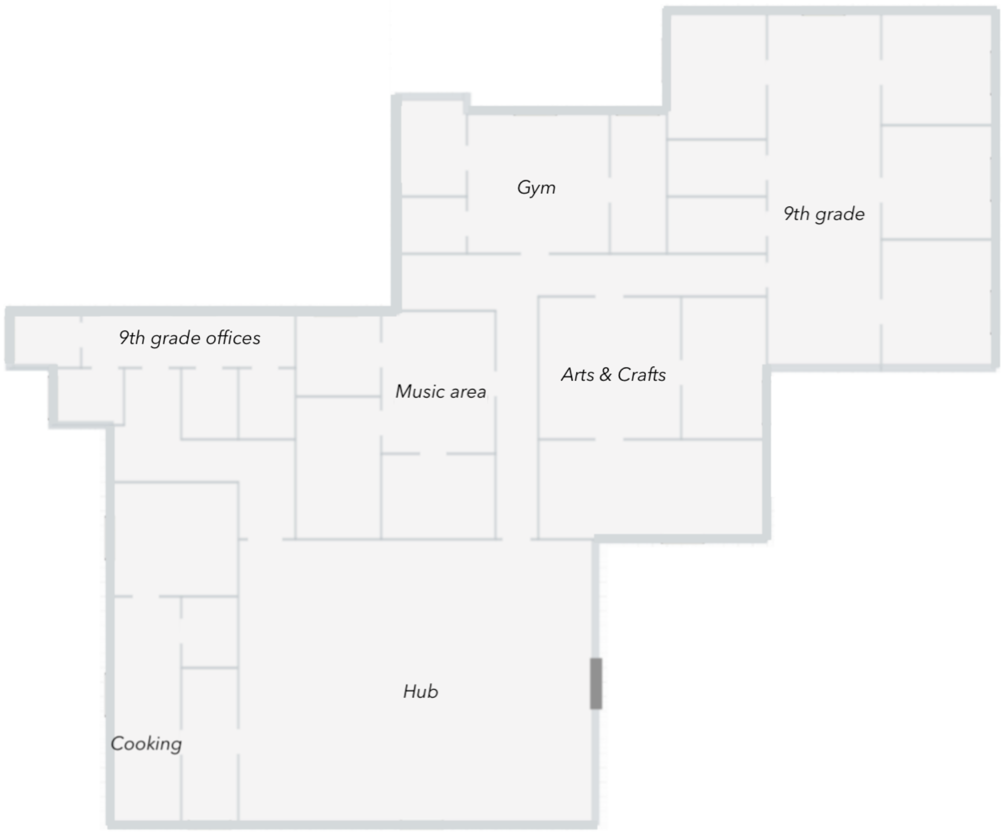


Figure 8.2: Blueprint of the layout in Figure 8.1

Chapter 9

Technical study, memetic algorithm

This chapter studies the performance and capabilities of the memetic algorithm. The performance of an MA is profoundly affected by its parameter settings and objective weights. These are collectively referred to as the *settings*. Though tests have been performed continuously during development, a structured approach provides important insights into how to improve the performance by tuning the parameters and weights. After obtaining desirable settings, the performance of the MA is tested on a variety of instances.

As this technical study is comprehensive, only the most interesting findings and results are presented in this chapter. Additional results are presented in Appendix C and D. The test methodology is discussed and presented in Section 9.1. In Section 9.2, different parameter settings are tested. These parameter settings include population initialization, crossover and mutation rates, as well as elitism rate and tournament size. The section concludes the assessment of parameter settings by testing the local search of the MA. Section 9.3 examines how the objectives impact the solutions, and conduct tests to find desirable objective weights. After desirable parameter settings are found, and the objective weights are set, the MAs performance on six different instances is examined in Section 9.4. The technical study is concluded by studying the impact of adding another objective; the number of exterior corners. The MAs ability to minimize exterior corners, as well as the effect adding this corner objective has on the other objectives, are assessed in Section 9.5.

The MA is implemented in JAVA using the IDE IntelliJ IDEA. Python is exploited for data analytics and visualization of the generated test results. The specification of the hardware and software used to implement and perform tests on the MA is presented in Table 9.1.

Table 9.1: Details of the computer hardware and software used for the memetic algorithm

CPU	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
RAM	32 GB
JAVA Version	11.0.4
Python Version	3.7
JAVA IDE	Intellij IDEA Version 2018 3.5
Python IDE	Pycharm Version 2017 2.2

9.1 Methodology

The six RSDs presented in Chapter 8 are used as test instances in this technical study. All tests in Sections 9.2 and 9.3 are performed using RSD $6N$ as input. As Levanger Middle School consists of 4400 square meters of room area spread over two floors, this RSD is considered reasonable sized for a single floor with its 2061 square meters. The values chosen as preliminary settings are based on testing during the development phase. These settings are referred to as the base case, and is displayed by a dashed line in the charts. When there is a trend in the data indicating that changing a parameter setting is beneficial, that parameter setting is adjusted, and the new value is used in subsequent tests. In other words, the base case is updated continuously. The complete list of initial settings can be found in Appendix B.

The first phase of the testing isolates a parameter to assess its impact and find a desirable value. An individual parameter value is tested by running the algorithm 30 times with a population size equal to 30, for 150 generations. The mean fitness of the best individual from each run is chosen as the basis for comparison. As only one solution is propagated from the MA to the mathematical model in stage two, it is reasonable to consider the best individuals. Additionally, violating feasibility can allow significant improvement in the remaining objectives. Thus, to set a reasonable ground for comparison, only the runs producing feasible solutions after the 150 generations are considered in the average. This holds for all tests conducted in this chapter. The resulting charts for the parameter settings tests show the number of generations on the x-axis and the mean fitness score on the y-axis. Note that the y-axis follows a logarithmic scale. For a single parameter, multiple values are tested, and all other settings are kept equal. Subsequently, to gain further insight into the algorithm, combinations of mutation probabilities are adjusted and tested.

After desirable parameter settings are found, the objective function is tested. The same test approach is used as with the parameter settings, by evaluating them each individually, but with a different evaluation method. This evaluation method is explained in the first paragraphs of Section 9.3. The objective variables of the MA are for convenience repeated in Table 9.2, along with the exterior corners objective f_8.

When the parameter settings and the weights of the objective function are fixed, the

MA is tested on the six instances of different complexity. Two terms are frequently used when evaluating the solutions. A feasible solution fulfils overlap (f_1) and connectivity (f_2). An objective is fulfilled if its corresponding value is zero. A satisfactory solution is feasible, and also fulfils objective f_3 - f_5. The algorithms ability to produce feasible and satisfactory solutions, as well as the run time, are considered when evaluating its performance. Additionally, the satisfactory solutions are compared on the two remaining objectives, f_6 and f_7. When assessing the MA's ability to minimize exterior corners by including objective f_8, the same test methodology is chosen along with a visual study of the layouts.

Table 9.2: Objective variables of the MA along with the additional tested objective, f_8 exterior corners

	Objective variables, MA
f_1	Overlap
f_2	Connectivity
f_3	Narrow hallways
f_4	Door-neighbour distance
f_5	Window access
f_6	Hallway area
f_7	Excess neighbourhood area
f_8	Exterior corners

9.2 Parameter settings

The major part of the MA is the genetic algorithm. A challenge with GAs is the number of parameters that need to be tuned. Changing one parameter can have a significant impact on the algorithm, and changing several can have a different impact than the sum of changing them one by one. On the other hand, the number of parameter settings provides flexibility to tweak and tune the algorithm. Due to the number of parameters and their interdependencies, finding an optimal combination of parameter settings is not expected. However, by combining the results of the study with intuition, a good approximation can be found. It is important to note that genetic algorithms are by nature non-deterministic. Therefore, a certain degree of variance in the results is expected.

9.2.1 Crossover

The results in Figure 9.1 indicates that the algorithm produces the best results when the crossover rate p_c is set at $p_c = 0.0$ or $p_c = 0.2$. The crossover rate in the base case is low, at $p_c = 0.4$, since the initial hypothesis was that crossover introduces too much random change for a single generation. While the results strengthen this hypothesis, crossover does help with getting out of local optima. Therefore, it is not completely removed, and the crossover rate is fixed at $p_c = 0.2$.

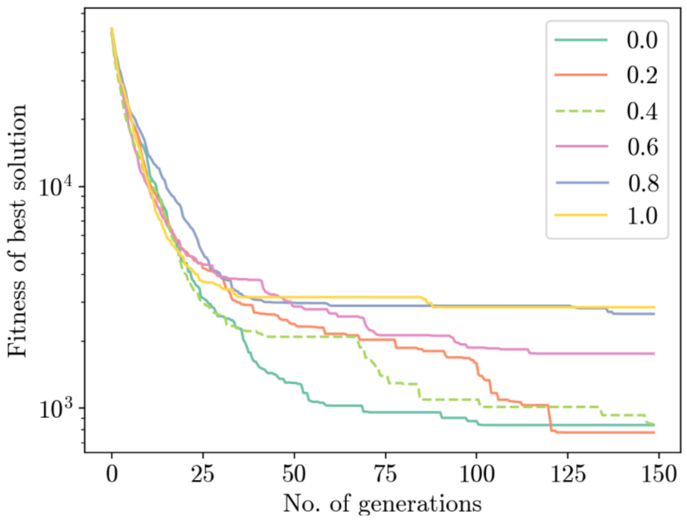


Figure 9.1: Fitness of the population for different crossover rates, p_c

9.2.2 Mutations

The effects of changing the mutation rate p_m is assessed in the following. Figure 9.2 shows how the MA performs with different mutation rates. With a mutation rate of $p_m = 0.0$, the algorithm does not find any feasible solutions. Thus, it is not part of the plot. The poor performance is expected as the mutations are the backbone of the implemented GA. Furthermore, the figure illustrates that a high mutation rate yields desirable results. However, $p_m = 1.0$ performs worse than $p_m = 0.6$ and $p_m = 0.8$. This is likely due to excessive alterations of the individuals, resulting in a worsening of the best solutions. Based on the trend in the graph and experience from prior testing, the mutation rate is fixed at $p_m = 0.6$. This allows substantial jumps in the solution space while avoiding excessive alterations.

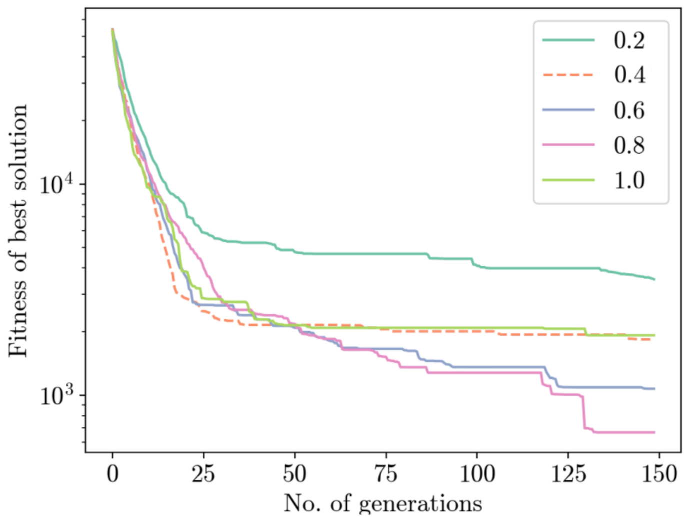


Figure 9.2: Fitness of the population for different mutation rates, p_m

Tests of the mutation operators and their final mutation probabilities are presented in Appendix C. The results lack consistent trends, which strengthens the hypothesis regarding the interdependencies of the parameter settings, as well as the inherent randomness of GA. Considering the lack of consistent trends in the test data, and the vast number of possible mutation probability combinations, intuition is combined with the test results to determine desired mutation probabilities.

9.2.3 Elitism

Figure 9.3 shows test results for different elitism rates, p_e. An elitism rate of zero produces no feasible solutions. This is an expected result as there is no guarantee for keeping the best solutions. The solution improves drastically by turning the elitism rate on, if even just slightly. The results also indicate that too much elitism does not give the algorithm enough freedom to explore, as an elitism rate at $p_e = 0.8$ and $p_e = 1.0$ yields significantly worse results than the base case $p_e = 0.2$. Additionally, a low elitism rate, such as $p_e = 0.05$ and $p_e = 0.1$ performs poorly, as it causes the algorithm to discard potentially good solutions. From the results, it can be concluded that an elitism rate of $p_e = 0.4$ efficiently balances exploration and exploitation, and the rate is therefore fixed at $p_e = 0.4$.

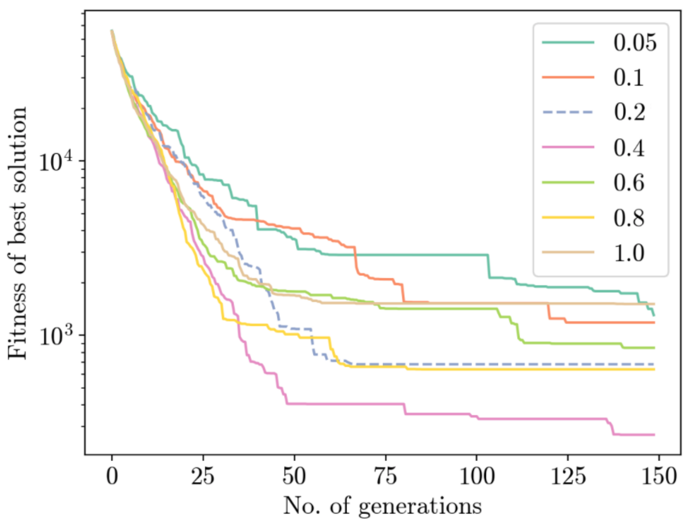


Figure 9.3: Fitness of the population with different elitism rates, p_e

9.2.4 Local search

The algorithms performance for different values of p_l is shown in Figure 9.4, where p_l is the probability of performing local search in a generation. The results clearly show that a higher probability is favourable, both in terms of better solutions and faster convergence. This is also the expected result and confirms that local search is an important contribution to the algorithm. The local search rate in the base case is quite low, as the initial hypothesis was that a low probability p_l would steer the algorithm in the right direction without ending up in a local minimum. This hypothesis is rejected by the results, as a higher rate p_l yields better results. Thus, the local search rate p_l is set to 1.0, meaning that the LS is applied in every generation.

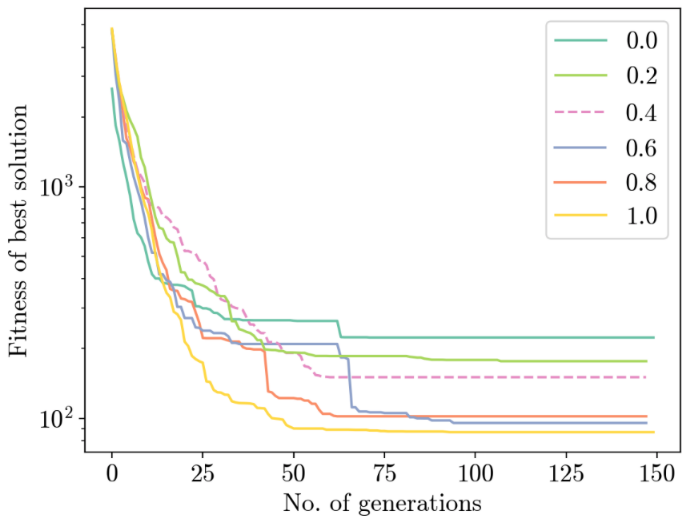


Figure 9.4: Fitness of the population with different local search probabilities, p_l

9.2.5 Population initialization and tournament size

Tests are conducted to determine a beneficial tournament size and to assess the different initialization approaches. The complete results are presented in Appendix C. As discussed in Section 5.4, the MA uses two different approaches for initializing an individual, a heuristic approach and a random approach. A probability parameter p_h determines the probability of the heuristic approach being chosen to initialize an individual. The results are presented in Section C.1 and show a trend where a higher p_h yields better solutions. Based on the results, p_h is set to 0.8. The tests conducted on the tournament size do not provide any insight on whether to increase or decrease the tournament size, it is therefore kept at $p_t = 0.4$. See Section C.2 for complete results. The initial and final parameter settings of the memetic algorithm are stated in Table 9.3.

Table 9.3: Initial and final parameter settings

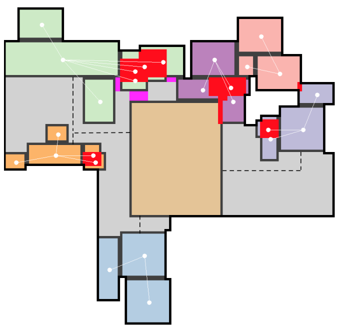
Initial parameter settings		*Final parameter settings*	
GA rates		**GA rates**	
Initialization rate	0.1	Initialization rate	0.8
Crossover rate	0.4	Crossover rate	0.2
Mutation rate	0.4	Mutation rate	0.6
Elitism rate	0.2	Elitism rate	0.4
Tournament size	0.4	Tournament size	0.4
Mutation probabilities		**Mutation probabilities**	
Move room random	0.2	Move room random	0.1
Swap wall-sharing side	0.4	Swap wall-sharing side	0.4
Change room dimension	0.3	Change room dimension	0.1
Move overlapping room	0.3	Move overlapping room	0.4
Move to not-attached door-neighbour	0.6	Move to not-attached door-neighbour	0.6
Move neighbourhood random	0.4	Move neighbourhood random	0.4
Move attached door-neighbours	0.3	Move attached door-neighbours	0.5
Swap rooms	0.4	Swap rooms	0.4
Local search		**Local search**	
Local search probability	0.4	Local search probability	1.0

9.3 Objective function

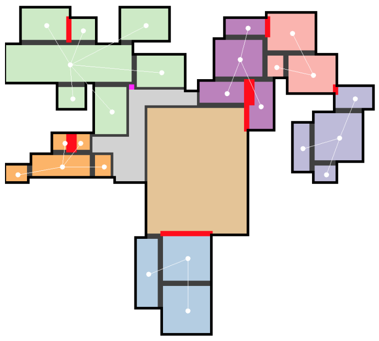
In this section, tests are performed to examine how the objectives impact the solutions, such that their corresponding weights can be accurately determined. Additionally, as optimizing for numerous objectives simultaneously is challenging, removing any excess objectives is favourable. Consequently, tests are conducted to assess whether or not objectives that seemed necessary during development are sufficiently handled by a combination of the other objectives. For instance overlap (f_1) may be considered through the door-neighbour objective (f_4), as two rooms violate f_4 if they overlap. Once again, the algorithm is run with a population size of 30 for 150 generations. When testing an objective, its weight is set to zero to examine how the MA performs without it. The evolution of the solutions are evaluated visually, and the final objective values are considered to observe the impact of the objectives.

The results for the objective function tests are as expected. All objectives are considered necessary, and kept in the MA. To exemplify how the tests are performed the connectivity objective is addressed in the following. The results for the other objectives are presented in Appendix D.

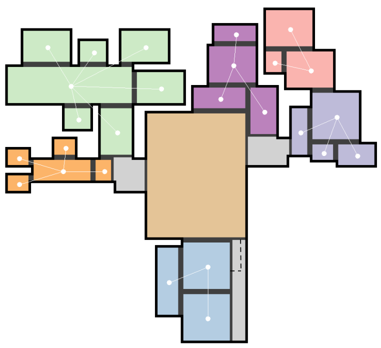
Figure 9.5 displays the evolution of a layout when the connectivity weight is set to zero. The figure shows the layout after 5, 20, 50 and 150 generations. Recall that overlap is illustrated in red, while narrow hallways are coloured in magenta. If there exists a direct path through hallways between the hub and a main room, it is illustrated by a dashed black line. The layouts violate connectivity and have a minimal amount of hallway area. Figure 9.5 (a) shows a layout with a connectivity score of 3, as three neighbourhoods are not connected to the hub, which is a requirement for connectivity. After 150 generations, four out of six neighbourhoods are not connected to the hub. The results indicate that the connectivity weight is necessary.



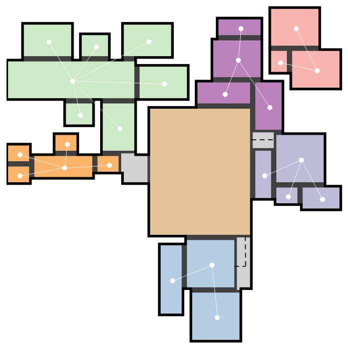
(a) Generation 5. Connectivity score of 3.



(b) Generation 20. Connectivity score of 6.



(c) Generation 50. Connectivity score of 5.



(d) Generation 150. Connectivity score of 4.

Figure 9.5: Development of a layout over 150 generations with connectivity turned off. The dashed black lines show direct pathways through hallways to the main room of a neighbourhood. After 150 generations, only two out of four neighbourhoods are connected to the hub.

To further assess the necessity of the connectivity objective, the MA is run 30 times, including and excluding the connectivity objective in the fitness function. The average hallway area and connectivity scores are presented in Table 9.4. The table shows that the amount of hallway area decreases drastically when turning off the objective, but the algorithm fails to provide solutions that fulfil connectivity. Hence, it is concluded that the connectivity objective is necessary. Furthermore, the results illustrate the conflicting goals of minimizing building area while maintaining connectivity. The connectivity weight is lowered slightly to facilitate obtaining solutions with less hallway area.

Table 9.4: Average hallway area and connectivity score when running the MA including and excluding the connectivity objective (f_2)

	Hallway area(%)	**Connectivity**
Including f_2	21.2	0.2
Excluding f_2	5.1	4.3

9.4 Performance testing

To assess the performance of the MA and gain deeper insight into its capabilities, a number of tests using the final settings are conducted. These settings are found in Appendix B. The six RSDs presented in Chapter 8 are used as test instances.

First, the fitness of the solutions generated for each instance is studied. Figure 9.6 shows the average fitness of the solutions for each instance for generations 0 to 150. The results are as expected as the average fitness is higher, thus worse, for the complex instances. By complex means an instance with either more rooms, neighbourhoods or total area of rooms, or a combination of the three. To exemplify, more rooms and neighbourhoods require more hallway area to fulfil connectivity, which adversely affects the fitness score. The complex instances also converge slower, which is consistent with the expectations. For example, increasing the number of rooms gives more door-neighbour relationships to fulfil, and additional operations are necessary to reach convergence.

An interesting finding is the difference in fitness between instance $3N$ and $7N$. They have an equal number of rooms, but for $3N$, the algorithm obtains a significantly better score than for $7N$. On the other hand, the fitness score of the solutions for instance $6N$ and $7N$ are close to equal, even though $6N$ has both more area and six rooms more than $7N$. The results show that the number of rooms, the total area, and the number of neighbourhoods determine the complexity of the instances, and that comparing instances must be done on a case-by-case basis.

9.4.1 Generation of feasible and satisfactory solutions

Table 9.5 shows the fraction of runs the algorithm produces feasible and satisfactory solutions. The results indicate a pattern of decreasing fraction of both feasible and satisfactory solutions as the instance becomes more complex. This is compatible with the expected behaviour. More rooms lead to more chances of overlap, and more neighbourhoods lead to more rooms requiring connectivity, which complicates finding feasible solutions. Similarly, the requirements a solution must meet to be considered satisfactory are harder to fulfil as the instances become more complicated. The fraction of feasible solutions decreases with a lower rate than the satisfactory ones, which is natural as the objectives to fulfil for a solution to be feasible is a subset of the objectives for a satisfactory solution.

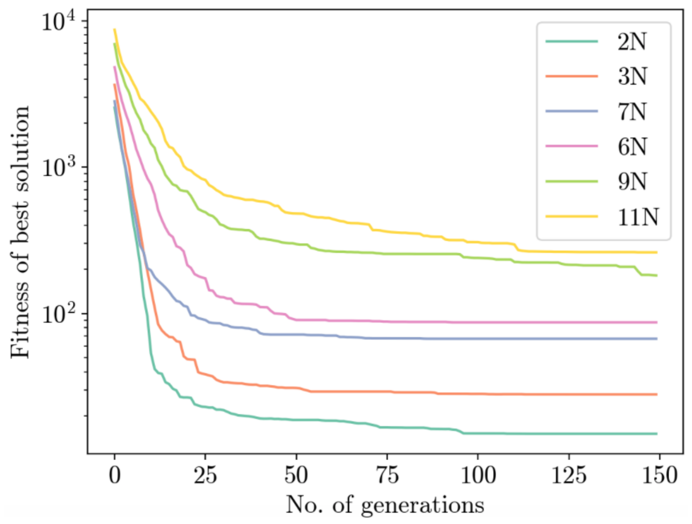


Figure 9.6: The six instances and their averasinglege fitness throughout the generations

Table 9.5: Fraction of feasible and satisfactory solutions generated for the six instances tested

Instance	**Fraction of feasible solutions**	**Fraction of satisfactory solutions**
$2N$	1.0	1.0
$3N$	1.0	1.0
$7N$	1.0	1.0
$6N$	0.87	0.83
$9N$	0.70	0.57
$11N$	0.53	0.43

For the instances with no more than 21 rooms ($2N$, $3N$ and $7N$), all runs provide feasible and satisfactory solutions. When the number of rooms increases, the fraction of feasible and satisfactory solutions decreases accordingly. The objective values of the largest instance are examined to study the bottleneck of reaching feasible and satisfactory solutions. Table 9.6 presents the average objective values for the infeasible runs of instance $11N$. The table shows the average score for each objective, as well as fitness score, for generations 0 to 150. The abbreviations are explained in Table 9.7, and are further used for readability. The table shows an average connectivity score of zero and an overlap score of 8.8 at generation 150. The numbers are highlighted in blue. When the MA fails to find feasible solutions, it is because of overlapping rooms. As a result, the overlap weight, w_1, is increased and the MA is run using $11N$ as input with the updated weight. This is further referred to as instance $11N_o$.

Table 9.6: Average objective values of the infeasible runs of instance $11N$

		Average objectives						
Generation	Fitness	O	HA(%)	D	C	NH	WA	ENA
0	8626.8	566.5	52.4	232.2	0.0	46.4	5.5	1616.1
10	2318.2	197.1	47.9	13.1	0.0	19.1	1.1	1547.1
20	1089.1	94.9	44.7	2.2	0.0	11.1	1.1	1557.1
30	852.8	62.6	43.4	2.2	0.0	5.5	1.1	1656.1
40	654.6	42.4	41.6	6.6	0.0	3.3	0.0	1778.1
50	617.6	35.3	41.3	6.6	0.0	4.4	0.0	1683.1
60	611.6	35.3	39.3	6.6	0.0	4.4	0.0	1678.1
70	598.5	27.2	39.4	8.8	0.0	4.4	0.0	1740.1
80	467.4	12.1	38.4	9.9	0.0	5.5	0.0	1759.1
90	455.4	12.1	37.1	9.9	0.0	4.4	0.0	1808.1
100	443.4	8.8	36.7	9.9	0.0	3.3	0.0	1808.1
120	438.4	8.8	35.5	9.9	0.0	3.3	0.0	1827.1
150	438.4	8.8	35.5	9.9	0.0	3.3	0.0	1827.1

Table 9.7: Objective abbreviations

Objective abbreviations	
O	Overlap
HA	Hallway area
D	Door-neighbour distance
C	Connectivity
NH	Narrow hallways
WA	Window access
ENA	Excess neighbourhood area

Table 9.8 is an updated version of Table 9.5, which includes instance $11N_o$. Increasing the overlap weight increases the fraction of feasible solutions of instance $11N$ drastically from 53% to 100%, producing only feasible solutions. Also, the fraction of satisfactory solutions increases from 43% to 57%. As feasibility is vital for the quality of the solutions, a low fraction of feasible solutions might result in having to re-run the MA to obtain adequate results. Consequently, the fraction of feasible solutions is important to find desirable solutions in an acceptable amount of time. However, increasing the fraction of feasible solutions may come at a cost, as it can adversely impacts the remaining objectives.

Table 9.8: Fraction of feasible and satisfactory solutions generated for the six instances tested, including the additional instance $11N_o$

Instance	**Fraction of feasible solutions**	**Fraction of satisfactory solutions**
$2N$	1.0	1.0
$3N$	1.0	1.0
$7N$	1.0	1.0
$6N$	0.87	0.83
$9N$	0.70	0.57
$11N$	0.53	0.43
$11N_o$	1.0	0.57

Figure 9.7 is an extension of Figure 9.6 where instance $11N_o$ is added. The figure shows that the average fitness for the feasible instances of $11N_o$ is worse than that of instance $11N$. Thus, the increased weighting on overlap negatively affects the overall fitness.

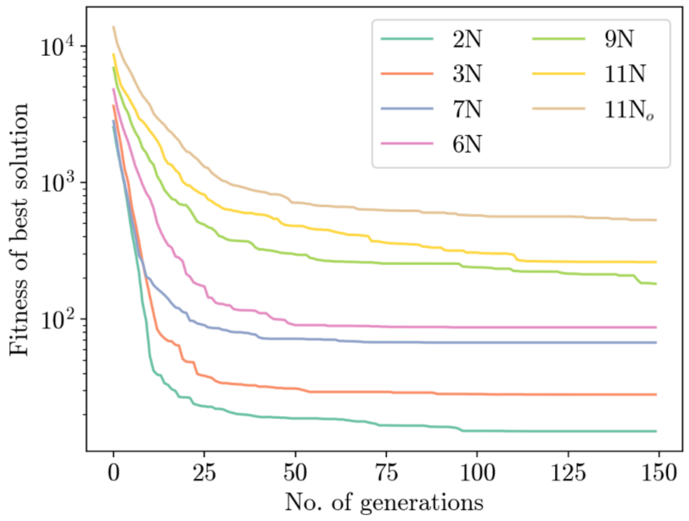
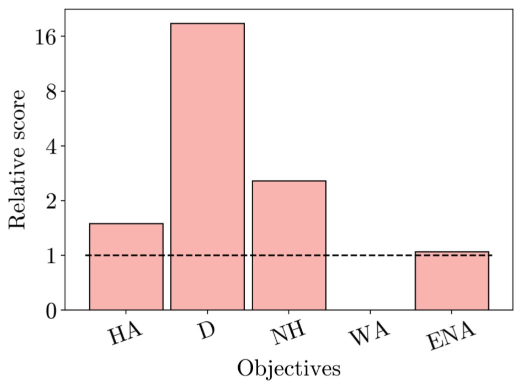
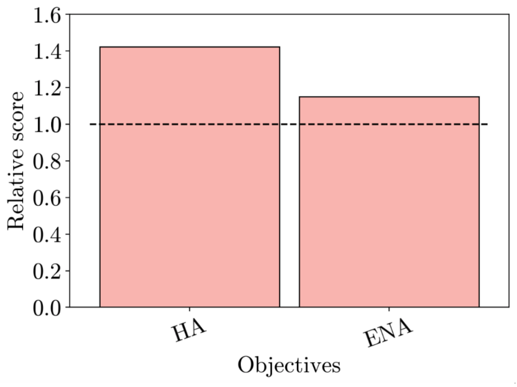


Figure 9.7: The six instances and their average fitness throughout the generations, along with the additional instance $11N_o$

Figure 9.8 compares the feasible and satisfactory solutions generated for the two instances $11N$ and $11N_o$. Each bar shows the objective value for instance $11N_o$ divided by the objective value for $11N$. This number is referred to as the *relative score* of the objective. For instance, if the average hallway area score for $11N_o$ and $11N$ is 1000 and 500 respectively, the relative score of HA would be 2. Thus, the hallway area for instance $11N_o$ would be twice as large as for instance $11N$ on average. If the average objective values were equal, also including both zeros, the relative score would be 1. A relative score of 1 is illustrated by a dashed line in the figures. Figure 9.8 (a) compares the two instances on the average objective values of the feasible solutions generated at generation 150, while (b) considers the satisfactory solutions.



(a) Feasible solutions



(b) Satisfactory solutions

Figure 9.8: Relative scores of the different objectives comparing $11N$ and $11N_o$. (a) compares the feasible solutions while (b) compares the satisfactory solutions.

The results show notably worse scores for the feasible solutions for $11N_o$ compared to $11N$. Even though all runs of instance $11N_o$ turn out feasible, the solutions fail to fulfil door-neighbour relationships as illustrated by the bar D. The algorithm also obtains a larger amount of narrow hallways and hallway area for instance $11N_o$. The only objective where the MA performs better on instance $11N_o$ than $11N$ is the window access objective. Looking at Figure 9.8 (a), the relative score of WA is zero. That is because the average WA of $11N_o$ is 0 and 1.0 for $11N$. Thus, the algorithm only fulfils this objective slightly more for $11N_o$ than $11N$. For the satisfactory runs, the hallway area is on average 40% greater with the increased overlap weight, as shown in Figure 9.8 (b). The results indicate that a higher priority of the overlap objective indirectly give less priority to the remaining objectives, resulting in worse fitness scores.

For the larger instances, the results provide two possibilities; increasing the overlap weight, resulting in more feasible and satisfactory solutions on average, or keeping the overlap weight unadjusted, resulting in fewer, but far better, feasible and satisfactory solutions. Besides minimizing cost, it is desirable to generate a wide range of different layouts which can be used as inspiration for architects. Thus, the choice depends on the run time. If the run time is high, increasing the overlap weight may be the best option to generate a sufficient amount of feasible solutions. Still, the results show that feasibility comes at a high cost with regards to the remaining objectives. If the run time allows for running the MA several times, the same amount of enhanced feasible and satisfactory solutions can be generated in an acceptable amount of time by keeping the overlap weight unadjusted. Enhanced satisfactory solutions imply less building costs as the objective value HA decreases, which is desirable. The overlap weight is therefore kept unadjusted until the run time of the algorithm is assessed.

9.4.2 Run time

The run times of the algorithm on the six instances are presented in Table 9.9. All times are given in minutes. The two columns show the average run time until a feasible and satisfactory solution is found. The table shows that the time to find a feasible or satisfactory solution increases with the complexity of the instance. This is consistent with the expectations as more operations are needed to fulfil i.e. connectivity, window access and door-neighbour relationships when the number of rooms and neighbourhoods increase.

Table 9.9: Run time to feasible and satisfactory solutions for the six test instances. Time is given in minutes.

Instance	**Run time, feasible solutions**	**Run time, satisfactory solutions**
$2N$	1.2	1.3
$3N$	3.0	3.7
$6N$	6.0	6.7
$7N$	4.4	4.9
$9N$	12.6	19.3
$11N$	22.4	39.3

The run time to feasible and satisfactory solutions are plotted for each instance in Figure 9.9, making it possible to study the relationship between the run time and the number of rooms of an instance. The instances are first sorted after the number of rooms, and then the total area of rooms. The figure shows that the run time is not linear in the number of rooms, but rather exponential. In general, increasing the complexity of the instances increases the time used to find a feasible and satisfactory solutions. Even though these results are consistent with the expectations, many aspects affect the results as elements vary among the instances. The combination of site size, the number and total area of rooms, and the number of neighbourhoods is unique for all instances and cannot be compared by one single factor. Even two instances with the same number of rooms and neighbourhoods, but different ones, can vary in complexity as there are differences in the number of possibilities to for instance fulfil door-neighbour relationships. Thus, the results discussed for Figure 9.9 may not be generalized.

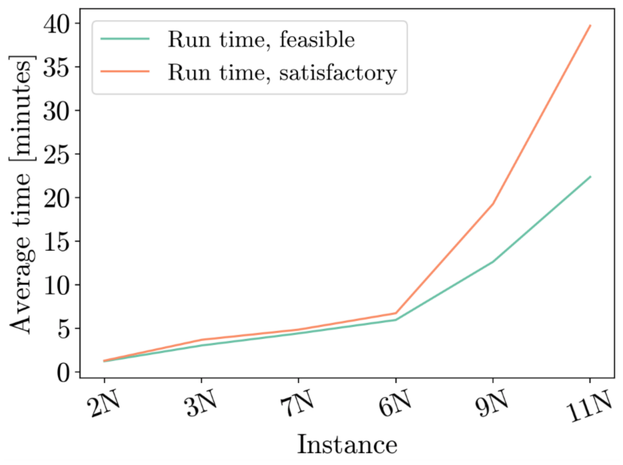


Figure 9.9: The six instances and their run time to feasible and satisfactory solutions.

Even though the run time increases with the complexity of an instance, the findings show that time is not a bottleneck. The algorithm spends 39 minutes on average on finding a satisfactory solution for the largest instance ($11N$), while about 7 minutes for instance $6N$. Based on these results, the run time allows for running the MA several times in an acceptable amount of time, as the time only increases linearly in the number of extra runs. Thus, to complete the discussion in Section 9.4.1, the overlap weight for instance $11N$ is kept unadjusted. As a result, a wide variety of enhanced layouts can still be obtained for large instances within a few hours.

9.5 Exterior corners objective

The building costs consist of two main components; the total building area and the number of exterior corners. Until now, the MA only considers the building area through the hallway area objective. Chapter 4 discussed the capabilities of GA and MAs, presenting the hypothesis that the MA is inadequate as a stand-alone solution method to minimize building costs in terms of area and exterior corners. To study the impact of also considering the other component, exterior corners, an additional objective of minimizing the number of exterior corners, f_8, is added for comparison. Review Chapter 7 for elaboration on the objective and how it is calculated.

The algorithm is run for all six instances with the additional objective and is compared to the runs where the objective is excluded. The settings found in Section 9.2 are used. Table 9.10 shows the fraction of feasible and satisfactory solutions for the test instances with and without objective f_8. As seen in Table 9.10, the fraction decreases when adding the objective for all instances except $2N$ and $3N$. As these are quite small instances, the MA manages to handle the extra objective and still find feasible and satisfactory solutions. Observe that the fraction of both feasible and satisfactory solutions remains unchanged for $3N$, while it decreases by 30% for instance $7N$. Adding a new objective to the algorithm implicitly decreases the focus on other objectives. A probable explanation for the decrease in the fraction of feasible and satisfactory solutions for $7N$, is the higher difficulty of ing connectivity for a seven neighbourhood instance than for three.

For the largest instances, $9N$ and $11N$, the algorithm does not manage to find any satisfactory solutions. This points to the conclusion that the MA is unsuitable for handling exterior corners. Still, instance $6N$ with its 27 rooms is considered reasonable sized for a single floor. Satisfactory solutions are found for instance $6N$ when adding the exterior corners objective. Thus, if the MA were to solve this instance to satisfaction while minimizing building area and exterior corners, it could be a sufficient stand-alone method for solving smaller floors. Still, the fraction of satisfactory solutions is reduced by 50%.

Table 9.10: Fraction of feasible and satisfactory solutions for the test instances, including and excluding objective f_8

	Excluding f_8		*Including f_8*	
Instance	Fraction of feasible solutions	Fraction of satisfactory solutions	Fraction of feasible solutions	Fraction of satisfactory solutions
$2N$	1.0	1.0	1.0	1.0
$3N$	1.0	1.0	1.0	1.0
$7N$	1.0	1.0	0.70	0.70
$6N$	0.87	0.83	0.67	0.40
$9N$	0.70	0.57	0.67	0.0
$11N$	0.53	0.43	0.63	0.0

Figure 9.10 compares the satisfactory solutions of instance $2N$, $7N$ and $6N$ with and without the exterior corners objective. Exterior corners is abbreviated EC. Once again, the bars show the relative score of each objective, where a number greater than one imply that the algorithm not considering objective f_8 performs better. Not surprisingly, the number of exterior corners decreases drastically by adding the objective. For instance $2N$ and $7N$ the number of exterior corners decreases by 50%, while it decreases by 30% for instance $6N$. However, this decrease comes at a cost. The hallway area is more than doubled for $2N$ and $7N$. When minimizing exterior corners comes at the cost of increased hallway area, the memetic algorithm fails to reach its goal of generating layouts with a small amount of building area.

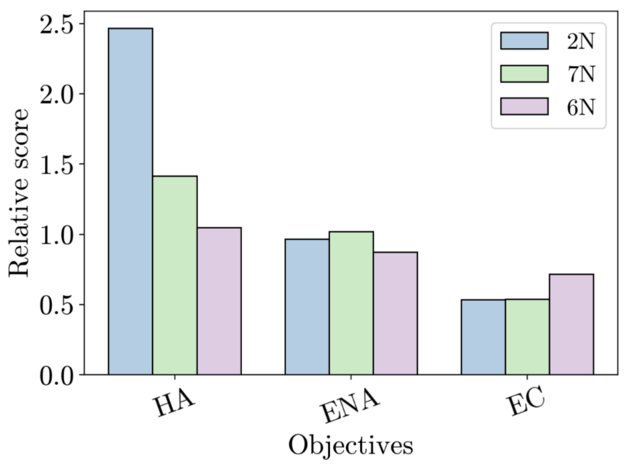
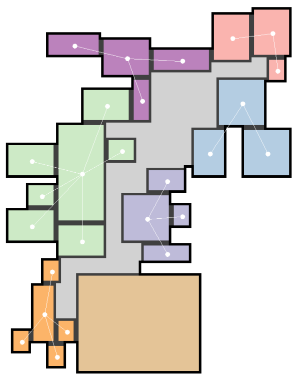


Figure 9.10: Instances tested with and without the exterior corners objective

Figure 9.10 reveals a pattern of decreasing negative effects on building area as the instance becomes larger or more complex. For instance $6N$, the hallway area is only slightly larger on average when including the exterior corners objective. This may be interpreted as if larger schools require more hallway area to ensure flow and connectivity and thus exterior corners may only increase this area to a limited extent. For smaller instances, less hallway area is required. Thus, minimizing exterior corners may have a greater negative impact on the building area.

As the performance of the algorithm for instance $6N$ shows positive trends regarding minimizing building area and exterior corners at once, the solutions are examined visually. Figure 9.11 shows two satisfactory solutions of $6N$ where the exterior corners objective is excluded in (a) and included in (b). The fitness and hallway area are close to equal for the two solutions. By comparing the two layouts, layout (b) contains notably fewer exterior corners than (a) and is more naturally shaped. However, although 36 exterior corners is a great improvement from 70, it is not sufficient. It is easy to find quick fixes of layout (b) which decreases the number of exterior corners without increasing other objectives. Hence, the MA is considered inadequate to handle exterior corners as it is not able to find these fixes.


(a) MA run with instance $6N$, excluding the exterior corners objective. 70 exterior corners.

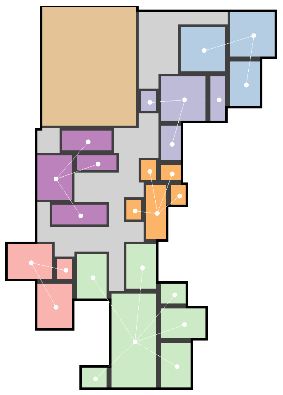

(b) MA run with instance $6N$, including the exterior corners objective. 36 exterior corners.

Figure 9.11: Layouts of instance $6N$ when the MA excludes the exterior corners objective f_8 from the fitness function in (a), while including the objective in (b)

The study of adding the exterior corners objective f_8 confirms that the MA with its settings is inadequate as a stand-alone solution method for minimizing building costs. The algorithm cannot find satisfactory solutions for larger instances, and fewer exterior corners comes at the cost of greater hallway area for smaller instances. Possibly, the MA could have performed better by conducting additional tests to find the "optimal" settings for solving the MA including the exterior corners objective. Still, by the arguments in Section 4, an MA is likely inadequate regardless of the settings as multiple consecutive moves are required to minimize exterior corners.

Chapter 10

Technical study, mathematical model

This chapter studies the mathematical model presented in Chapter 6, which constitutes stage two of the solution method. The objective of this study is to assess various modelling alternatives to optimize the performance of the algorithm as a whole.

The mathematical model is solved using the commercial optimization solver Gurobi, and the model is implemented in Java using the Gurobi Java interface. Python is exploited for data analytics and visualization of the generated test results. The tests are conducted on the computing cluster *Solstorm*, provided by the Department of Industrial Economics and Technology Management at NTNU. The specifications of the hardware and software used are presented in Table 10.1.

Table 10.1: Details of the computer hardware and software used for the mathematical model

CPU	2.4GHz Intel Xeon Gold 5115 CPU – 10 core
RAM	96 Gb
JAVA Version	11.0.4
Python Version	3.7
Gurobi Version	9.0.2
JAVA IDE	Intellij IDEA Version 2018 3.5
Python IDE	Pycharm Version 2017 2.2

Section 10.1 presents findings during preliminary testing. The test instances used are presented in Section 10.2. In Section 10.3 the test methodology for the technical study is discussed. Lastly, Sections 10.4 - 10.8 examine test results and assess the performance and capabilities of the various modelling heuristics. The sections present the most interesting results, while additional results are presented in Appendix E.

10.1 Preliminary testing

Initially, the mathematical model is implemented as described in Sections 6.2 - 6.4. Preliminary tests show that the computational complexity of the problem instances are too high for the model to find solutions within an acceptable amount of time. Therefore, the valid inequalities and symmetry breaking constraints, presented in Sections 6.5 and 6.6 respectively, are added to the model. These are not tested in the technical study as preliminary tests have proven them to have a positive, but small, impact on the run time of the model. The modelling heuristics presented in Section 6.7 have a much larger impact on the run time. Therefore, the technical study focuses on the effectiveness of these heuristics. The valid inequalities and symmetry breaking constraints are included in the model in all subsequent tests.

Additionally, the preliminary tests conclude that consecutively applying the model to the neighbourhoods is too time demanding, and the deficits of optimizing the neighbourhoods in parallel, discussed in Section 6.7.4, are negligible. Hence, in the tests conducted in this chapter, the model optimizes the neighbourhoods concurrently.

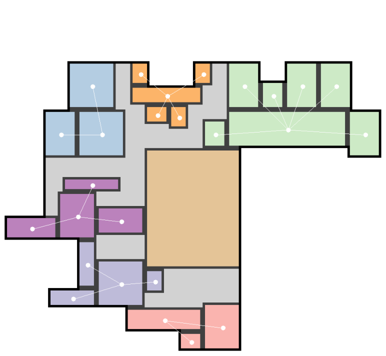
The memetic algorithm in stage one seeks to minimize the total building area, which in practice means to minimize hallway area. Preliminary testing shows that this cramps the neighbourhoods together, leading to overlapping extended envelopes. In turn, this gives the mathematical model insufficient leeway to place the rooms. In most cases, this yields a high number of corners. To address this issue, it is necessary to update the memetic algorithm such that the output of stage one, in general, contains more hallway area. Thus, the hallway area objective presented in Section 5.2 is updated such that if the hallway area is less than a determined percentage of the total building area, the objective value is set to zero. The threshold is set to 30%. Consequently, in subsequent tests, if the hallway area objective is zero it does not mean that the solution does not contain hallways, it means that the total hallway area is less than 30% of the total building area. Note that the hallway area objective in stage three remains unchanged, and is still subject to minimization.

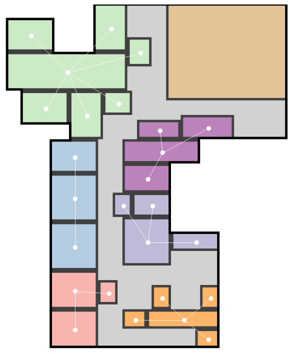
10.2 Test instances

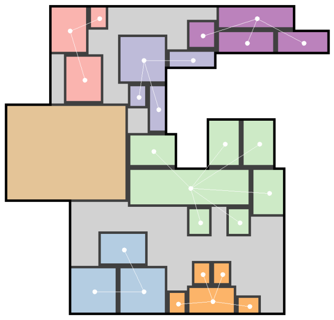
To accurately determine the effectiveness of the mathematical model, tests are performed on a variety of layouts generated from RSD $6N$ and $9N$, presented in Chapter 8. To account for the non-deterministic nature of the memetic algorithm, each model heuristic is tested on six different test instances. The test instances are generated by the MA with the final settings presented in Chapter 9. Three of these are from $6N$, and the remaining three are from $9N$. Together $6N$ and $9N$ consist of 12 unique neighbourhoods. Table 10.2 presents the neighbourhoods with key characteristics, and Figure 10.1 provides a visualization of the six test instances used as input in the technical study of the mathematical model.

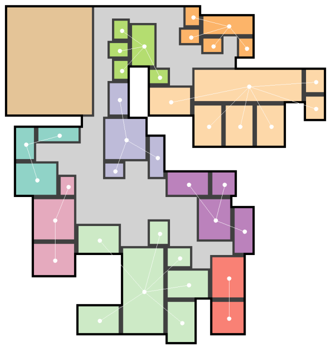
Table 10.2: The number of rooms, total area and assigned colour of each neighbourhood

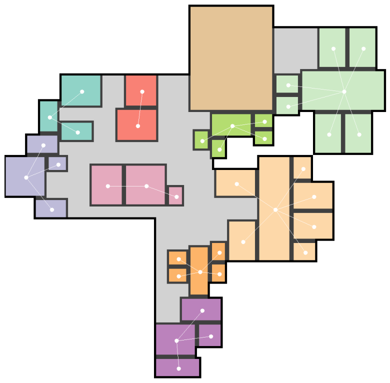
Neighbourhoods			
Neighbourhood	# of rooms	Total area (m^2)	Colour
Music area	4	231	🟪
9th grade offices	5	140	🟧
Gym	4	200	🟪
Arts & Crafts	3	270	🟦
9th grade	7	540	🟩
Cooking	3	180	🟥
10th grade offices	5	140	🟩
10th grade	7	560	🟨
Science	3	160	🟩
Library	2	144	🟥
Administration	3	200	🟪
Hub	1	500	🟫


(a) $6N_1$


(b) $6N_2$


(c) $6N_3$


(d) $9N_1$


(e) $9N_2$

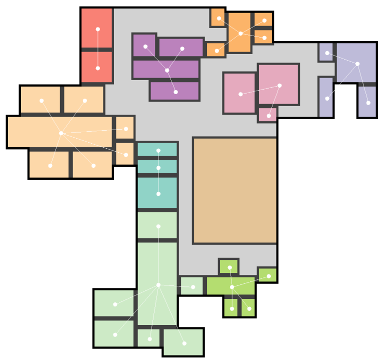

(f) $9N_3$

Figure 10.1: Initial layouts of the test instances used in the technical study of the mathematical model

10.3 Methodology

Initially, tests are performed on the model described in Sections 6.2 - 6.6. This implementation is referred to as the *basic model*. The performance of the model is judged on both quantitative and qualitative criteria. Key performance indicators are how well the model meets its explicit objective of minimizing corners in each neighbourhood, and the run time. Additionally, the model is assessed on its impact on the complete layout, and its compatibility with the subsequent stage of the solution method. This is done by evaluating the resulting solution on the objectives of the memetic algorithm and the number of exterior corners, and by qualitatively evaluating visual representations of the corresponding layouts.

Each test run consists of applying the model to each neighbourhood in a test instance individually. For a single run, a time limit of eight hours is decided to be reasonable. As the neighbourhoods are optimized concurrently, the model has eight hours to find the best possible solution for each neighbourhood. If the model is unable to find a solution that is guaranteed to be optimal within the given time, it returns the current best solution. If the model is unable to generate a feasible solution at all, the neighbourhood remains unchanged, and the model returns the neighbourhood generated from stage one. When the model terminates for all neighbourhoods, the neighbourhoods are put back in the layout and hallways are regenerated to form a complete layout.

After the performance of the basic model has been assessed, the effect of adding the modelling heuristics presented in Section 6.7 are studied. This is done by adding one of these heuristics to the model, re-running the tests and examining the change in the results. If the results indicate a positive effect of adding the heuristic, it is included in the model and used in subsequent tests. The order in which the modelling heuristics are tested is based on results from preliminary testing and interdependencies of the heuristics.

10.4 Basic model

The basic model is implemented as specified in Sections 6.2 - 6.6. Table 10.4 and 10.5 illustrate the performance of the model on the $6N$ and $9N$ instances, respectively. The table abbreviations are explained in Table 10.3. The tables show the number of corners in each neighbourhood before and after running the model. The initial number of corners, C_I, correspond to the corners of the neighbourhoods in the layout generated by stage one. The number of corners after applying the basic model to the neighbourhoods is denoted C_F^B. Additionally, the tables show the time until the model finds the best solution obtained, T_B, and the total run time, T. The time is given in minutes. $T = 480$ indicates that the model ran for the total time at disposal, and terminated without being able to guarantee that the problem is solved to optimality. If $T < 480$, the model can guarantee optimality of the solution found at time T_B. Lastly, when the model fails to find any feasible solutions, it is illustrated as $T_B =$"-" in the tables. When no feasible solution is found, the model returns the neighbourhood given by the memetic algorithm. Hence, $C_I = C_F$.

Table 10.3: Abbrevations for Table 10.4 and 10.5

C_I	Initial corners
C_F^B	Final corners
T_B	Time until best solution is found
T	Total run time

Table 10.4: Initial and final number of corners and run time for the $6N$ instances run with the basic model. Time is given in minutes.

	$6N_1$				$6N_2$				$6N_3$			
	C_I	C_F^B	T_B	T	C_I	C_F^B	T_B	T	C_I	C_F^B	T_B	T
Music area	14	6	445	480	12	6	480	480	10	6	160	480
9th grade offices	16	10	478	480	12	8	142	480	12	6	46	480
Gym	12	6	260	480	10	6	480	480	12	8	336	480
Arts & Crafts	8	6	161	480	4	4	108	108	8	4	176	176
9th grade	14	14	-	480	20	20	-	480	18	18	-	480
Cooking	8	6	44	480	8	6	24	480	10	6	37	480

Table 10.4 shows that the model finds feasible solutions for all neighbourhoods except *9th grade* in all $6N$ instances. As *9th grade* contains the most rooms, it is expected to be the most challenging neighbourhood. *9th grade* is also the largest neighbourhood in terms of area which yields a large extended envelope. Collectively, this results in a search space the model is unable to handle. Furthermore, the model is only able to find and guarantee an optimal solution for *Arts & Crafts* in the instances $6N_2$ and $6N_3$.

The results are similar for the $9N$ instances, presented in Table 10.5. The model fails to find solutions for the most challenging neighbourhoods, *9th grade* and *10th grade*. A surprising result is that no feasible solution is found for the fairly simple neighbourhood *Music area* in instance $9N_1$. As illustrated in Figure 10.1 (d), the *Music area* 🟪 has a highly unnatural shape, and three other neighbourhoods lie close to it. A probable explanation for why no solution is found is that the shape of the *Music area* yields a large extended envelope, which increases the search space, and the surrounding neighbourhoods restrict the possibilities for placing the rooms. As for the $6N$ instances, the model fails to solve the neighbourhoods to optimality for the $9N$ instances. The exceptions are the *Science* and *Library* neighbourhoods for all test instances and the *Gym* for instance $9N_3$. The basic model generally fails to solve the neighbourhoods to optimality, and for some neighbourhoods fails to find any feasible solutions. Consequently, it can be concluded that it is insufficient to handle the desired complexity within the given time.

Table 10.5: Initial and final number of corners and run time for the $9N$ instances run with the basic model. Time is given in minutes.

	$9N_1$				$9N_2$				$9N_3$			
	C_I	C_F^B	T_B	T	C_I	C_F^B	T_B	T	C_I	C_F^B	T_B	T
10th grade offices	16	8	135	480	12	6	254	480	12	6	323	480
Music area	12	12	-	480	10	8	187	480	10	8	205	480
9th grade offices	14	8	109	480	12	8	68	480	10	6	187	480
Gym	14	8	158	480	14	6	154	480	10	4	425	425
Library	4	4	1	1	6	4	1	1	6	4	1	1
9th grade	20	20	-	480	16	16	-	480	12	12	-	480
Science	8	4	22	22	12	4	15	15	6	4	28	28
Administration	6	6	2	480	6	6	43	480	8	6	93	480
10th grade	12	12	-	480	14	14	-	480	16	16	-	480

Despite the complexity and run time issues, Table 10.4 and 10.5 clearly show that the behaviour of the model is as intended. For all neighbourhoods where feasible solutions are found, the resulting number of corners is less than the initial number. Figure 10.2 shows the sum of the initial, TC_I, and final number of corners, TC_F^B, in all neighbourhoods, for all instances. The number of corners drastically decreases for all instances, ranging from a 23% decrease for $9N_1$ to 33% for $6N_1$.

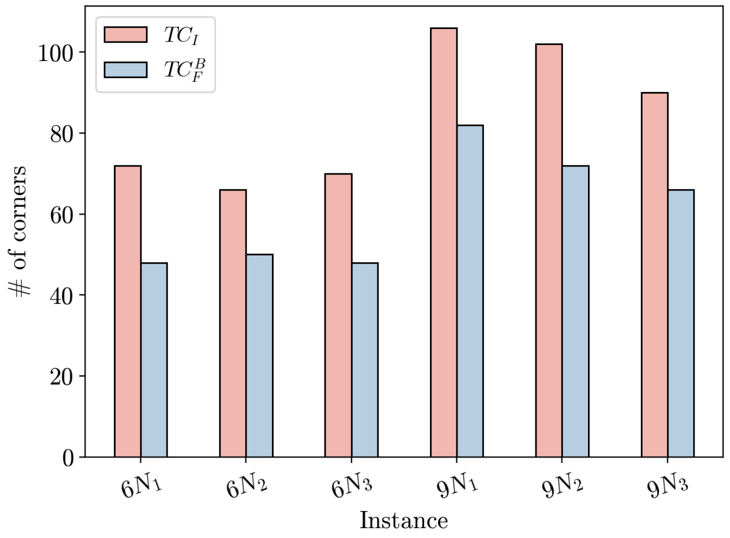


Figure 10.2: The total number of corners before, TC_I, and after, TC_F^B, running the basic model for all the six test instances

Table 10.6 evaluates the model performance by the objectives of the memetic algorithm, and the exterior corners objective (f_8) used in stage three. The objective abbreviations are repeated in Table 10.7. Table 10.6 (a) and (b) show the objective values of the instances before and after the model is run on all neighbourhoods. The columns "Pre" and "Post" denote the objective values before and after running the model. Clearly, the excess neighbourhood area decreases for all instances. This is an expected result as minimizing corners leads to more compact neighbourhoods. Furthermore, Table 10.6 (*a*) and (*b*) show that the mathematical model has an adverse effect on window access. Initially, all instances have a window access score of zero, and after running the model, they all have positive scores. A negative impact on window access is expected as the model does not take this objective into account.

Table 10.6: The objective values of the resulting solutions using the objectives in stage one and three

	$6N_1$		$6N_2$		$6N_3$	
	Pre	Post	Pre	Post	Pre	Post
O	0	0	0	0	0	1
HA(%)	0	31.0	0	41.3	0	35.2
D	0	0	0	0	0	0
C	0	1	0	0	0	1
NH	0	0	0	0	0	0
WA	0	2	0	3	0	4
ENA	710	423	731	512	686	362
EC	30	32	20	14	22	24

(a) $6N$ instances

	$9N_1$		$9N_2$		$9N_3$	
	Pre	Post	Pre	Post	Pre	Post
O	0	2	0	2	0	0
HA(%)	0	0	0	0	0	37.4
D	0	0	0	0	0	0
C	0	6	0	4	0	2
NH	0	0	0	8	0	21
WA	0	1	0	1	0	4
ENA	981	770	800	658	1115	623
EC	48	72	50	52	34	44

(b) $9N$ instances

Table 10.7: Objective abbreviations

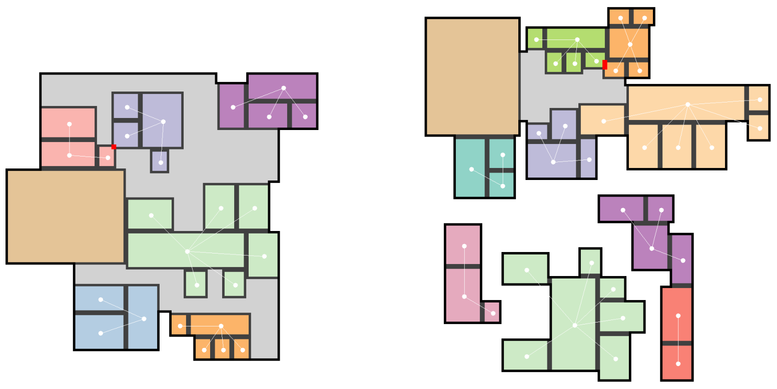
Objective abbreviations	
O	Overlap
HA	Hallway area
D	Door-neighbour distance
C	Connectivity
NH	Narrow hallways
WA	Window access
ENA	Excess neighbourhood area
EC	Exterior corners

The model yields positive overlap or connectivity scores for all instances except $6N_2$. Consequently, the corresponding solutions are considered infeasible. Furthermore, Table 10.6 shows that the door-neighbour distance is zero both before and after running the model across all instances. The reason is that all the test instances considered are initially satisfactory, and the mathematical model requires fulfilling the adjacency constraints. Consequently, the value of this objective remains zero.

In general, Table 10.6 clearly illustrates that the mathematical model adversely affects the fitness of a layout. This is expected as the model is not developed to optimize for the objectives used in stage one and three. The idea behind the solution method is that the local search in stage three corrects the negative impact of the mathematical model in regards to these objectives. Thus, stage three ensures feasible and satisfactory solutions. However, the resulting layouts after stage two must facilitate for the local search to be able to create desirable solutions.

Figure 10.3 (*a*) and (*b*) show the resulting layouts after running the basic model on $6N_3$ and $9N_1$, respectively. Figure 10.3 (*a*) shows that most neighbourhoods have taken a geometric simplistic and more natural shape. The exceptions are the *9th grade* 🟩 for both instances and the *Music area* for instance $9N_1$, for which the model fails to find a feasible solution. Furthermore, for instance $6N_3$ there is a slight overlap between the neighbourhoods *Cooking* 🟥 and *Gym* 🟪, and connectivity is not fulfilled for the *Music area* 🟪, as its main room is not adjacent to a hallway or the hub. Consequently, the solution is infeasible. However, it is likely that this will be fixed by the local search, as minor changes would yield a feasible solution.

Figure 10.3 (*b*) shows that connectivity is not fulfilled for as much as six of the nine neighbourhoods for instance $9N_1$. The *Gym* 🟪 neighbourhood is located such that it prevents a hallway from reaching the unconnected neighbourhoods. The local search does not have much leeway to relocate the *Gym* as the surrounding neighbourhoods lie close. In this case, it is rather unlikely that the local search is able to find a number of consecutive moves to make the solution feasible. The figure illustrates the challenges of large instances. Increasing the number of neighbourhoods complicates the task of finding non-conflicting locations for the rooms and neighbourhoods.



(a) $6N_3$ (b) $9N_1$

Figure 10.3: The resulting layouts after running the basic model on instance $6N_3$ and $9N_1$

The test results of the basic mathematical model show that the model is able to decrease the number of corners in the neighbourhoods. However, it is not able to handle the complexity of the test instances, as no feasible solutions are found for the largest neighbourhoods. Assessing the objectives used in stage one and three, the results also indicate that the model adversely affects the complete layouts. Most notably, essential objectives, such as connectivity and window access, are insufficiently considered. The deficits of the resulting layouts for the $9N$ instances are likely to be unmanageable for the local search in stage three.

10.5 Lock main room

To improve the performance of the model, the heuristic explained in Section 6.7.1 is implemented. In short, the heuristic locks the position of the main room before applying the model to a neighbourhood.

Table 10.8 and 10.9 present the results for the $6N$ and $9N$ instances. T_B and T correspond to running the model with the lock main room heuristic. For comparison, the final number of corners with and without the heuristic are displayed in the tables, denoted C_F^L and C_F, respectively. The extended model finds solutions for the *9th grade* neighbourhood in four out of the six instances. As the basic model was not able to find any feasible solutions for *9th grade*, the results indicate that locking the main room makes the model more suited to solve the larger neighbourhoods. On the other hand, it is not able to find a feasible solution for *10th grade* in any of the $9N$ instances. In addition to be more prone to find feasible solutions, the model also guarantees optimality for most neighbourhoods. The exceptions are *9th grade* and *10th grade*, for which the model is unable to prove optimality in any of the instances. The tables clearly show that the heuristic improves the performance. It is, however, evident that the model still struggles with the largest neighbourhoods.

Table 10.8: Initial and final number of corners and run time for the $6N$ instances with the lock main room heuristics

	$6N_1$					$6N_2$					$6N_3$				
	C_I	C_F	C_F^L	T_B	T	C_I	C_F	C_F^L	T_B	T	C_I	C_F	C_F^L	T_B	T
Music area	14	6	4	38	38	12	6	4	213	213	10	6	4	1	1
9th grade offices	16	10	6	32	480	12	8	6	67	480	12	6	6	24	260
Gym	12	6	4	18	18	10	6	4	35	35	12	8	8	8	19
Arts & Crafts	8	6	4	5	5	4	4	4	0	0	8	4	4	5	5
9th grade	14	14	14	-	480	20	20	8	348	480	18	18	14	297	480
Cooking	8	6	6	1	4	8	6	6	1	1	10	6	6	1	7

Table 10.9: Initial and final number of corners and run time for the $9N$ instances with the lock main room heuristics

	$9N_1$					$9N_2$					$9N_3$				
	C_I	C_F	C_F^L	T_B	T	C_I	C_F	C_F^L	T_B	T	C_I	C_F	C_F^L	T_B	T
10th grade offices	16	8	4	43	43	12	6	6	9	18	12	6	6	36	480
Music area	12	12	4	168	168	10	8	4	54	54	10	8	6	24	204
9th grade offices	14	8	6	58	480	12	8	6	5	26	10	6	4	18	18
Gym	14	8	6	7	131	14	6	4	24	24	10	4	4	26	26
Library	4	4	4	1	1	6	4	4	1	1	6	4	4	1	1
9th grade	20	20	16	381	480	16	16	6	309	480	12	12	12	-	480
Science	8	4	4	4	4	12	4	4	1	1	6	4	6	1	1
Administration	6	6	6	1	1	6	6	6	1	3	8	6	6	3	3
10th grade	12	12	12	-	480	14	14	14	-	478	16	16	16	-	480

Locking the main room restricts the possibilities for placing the other rooms, as they need to be attached to it. Hence, this heuristic likely excludes a significant number of feasible solutions, which can adversely affect the results. Figure 10.4 shows the sum of the corners in the neighbourhoods for all instances. The figure compares the final number of corners generated by the basic model, TC_F, and the extension locking the main room, TC_F^L. The extended model significantly outperforms the basic model on all instances. Thus, the benefits of narrowing down the solution space outweighs the negative impact of excluding solutions. There are usually multiple desirable, feasible solutions, and even though the heuristic excludes some desirable solutions, the remaining ones are easier to obtain.

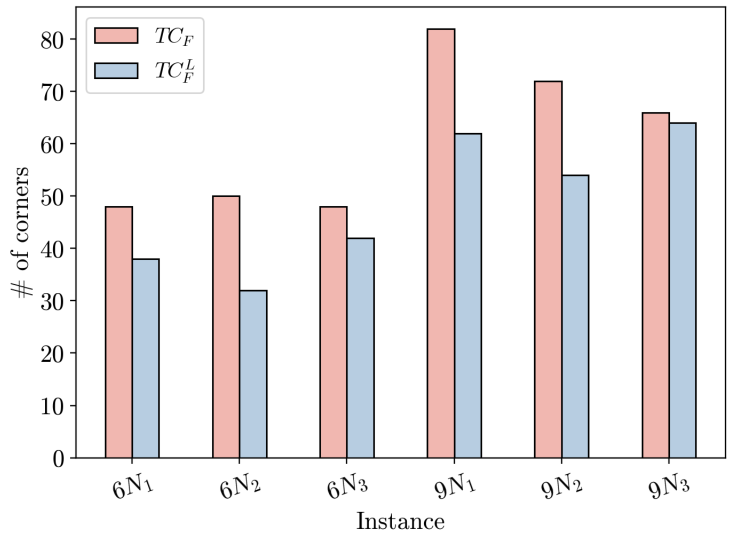


Figure 10.4: The total number of final corners with, TC_F^L, and without, TC_F, the lock main room heuristic

The improved efficiency of the model is further illustrated by Figure 10.5. The figure shows the average time until the best feasible solution found is obtained for each neighbourhood in the $9N$ instances. The figure compares the time T_B when running the basic model to the time T_B^L where the main room is locked. If the model fails to find a solution, the time is set to the maximum time of 480 minutes, as the model, in this case, returns the solution given from stage one. The figure shows that $T_B^L < T_B$ for all neighbourhoods except *10th grade*. It is evident that decreasing the solution space by locking the main room facilitates obtaining solutions in a more efficient manner.

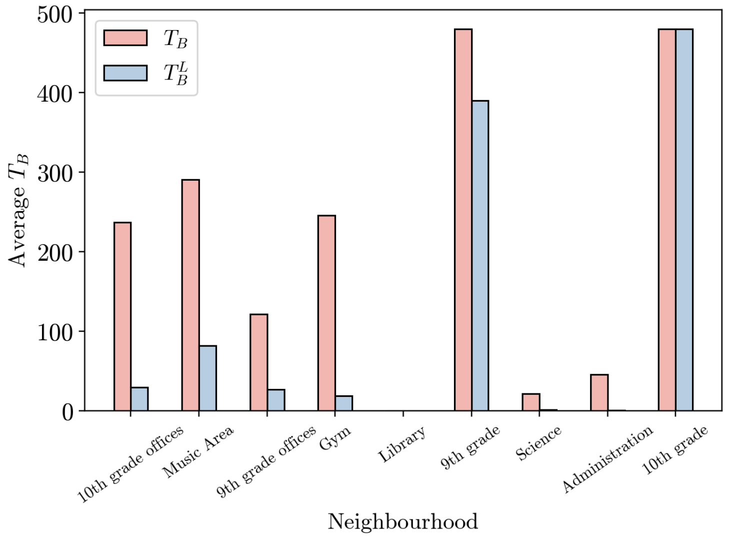
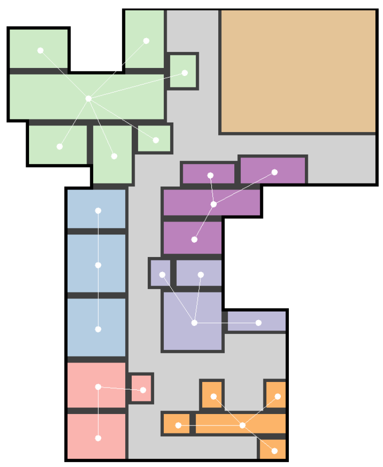


Figure 10.5: The basic model and the lock main room heuristic is compared on the average time until the best feasible solution is found for all neighbourhoods in $9N$, denoted T_B and T_B^L respectively

Figure 10.6 (a) and (b) visualize instance $6N_2$ before and after the model is run with the lock main room heuristics. Observe that the position, shape and size of the main rooms are equal in both (a) and (b). The figure illustrates that desirable neighbourhoods can be obtained when the main room is locked. Specifically, observe the major improvement to the *9th grade* neighbourhood 🟩 now that the model is able to find a feasible solution. The main issue with the resulting layout in (b) is the violation of connectivity. The neighbourhoods *Cooking* 🟥, *Gym* 🟪 and *9th office* 🟧 are not connected to the rest of the school. Still, the local search is likely able to fix this by moving the *Gym* 🟪 to the right, thus making space for hallways.


(a) Initial layout

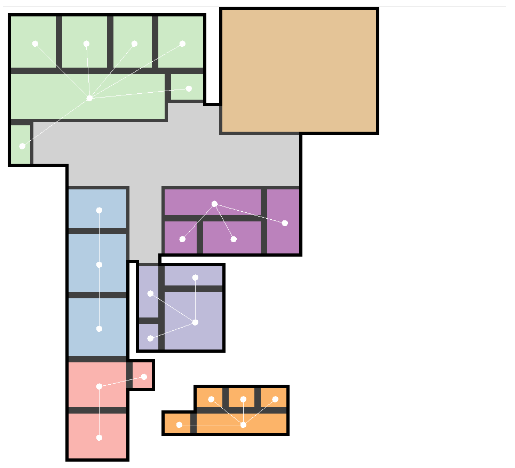

(b) With the lock main room heuristics

Figure 10.6: Instance $6N_2$ before and after the model is run with the lock main room heuristic

Including the lock main room heuristic shows positive results, as the model yields fewer corners, decreased run time and more often guarantees optimality. The heuristic is therefore included in subsequent tests. Regarding the objectives used in stage one and three, locking the main room yields similar results as the basic model, and the results are therefore not discussed. See Section E.1 for complete results.

10.6 Window objective

To address the issue of lacking window access in the layouts generated in stage two, the heuristic presented in Section 6.7.3 is implemented. Figure 10.7 presents the total number of corners for each test instance, with, TC_F^W, and without, TC_F, the window heuristic. The figure shows that the heuristic exclusively results in worse or unchanged solutions in terms of corners. The heuristic is not intended nor expected to decrease the number of corners. Nevertheless, the significant increase is an unexpected result.

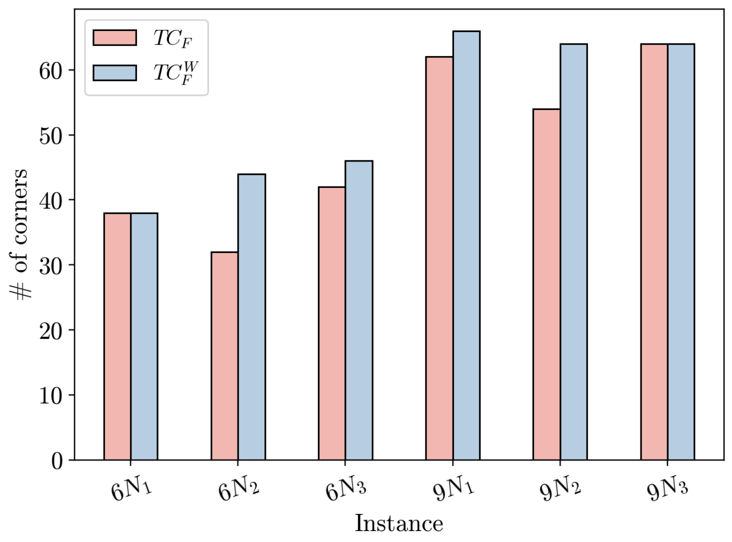


Figure 10.7: Total number of corners in each test instance with and without the window heuristic, denoted by TC_F and TC_F^W respectively

Observe from Table 10.10 and 10.11 that the model does not find any feasible solutions for neither *9th grade* nor *10th grade* before the time limit is reached. Previous to implementing the window heuristic, the model found feasible solutions for the *9th grade* neighbourhood in four out of six instances. Thus, the increase in corners is explained by the absence of feasible solutions for *9th grade*. Only considering the neighbourhoods where a feasible solution is found for both versions of the model, the difference in the total number of corners is insignificant. Hence, the heuristic does not adversely impact the resulting neighbourhoods in terms of number of corners when solutions are found.

Table 10.10: Initial and final number of corners and run time for the $6N$ instances with the window heuristic

	$6N_1$					$6N_2$					$6N_3$				
	C_I	C_F	C_F^W	T_B	T	C_I	C_F	C_F^W	T_B	T	C_I	C_F	C_F^W	T_B	T
Music area	14	4	4	16	16	12	4	4	433	433	10	4	4	7	7
9th grade offices	16	6	6	43	480	12	6	6	55	480	12	6	6	140	480
Gym	12	4	4	13	13	10	4	4	31	31	12	8	8	17	17
Arts & Crafts	8	4	4	6	6	4	4	4	1	1	8	4	4	9	9
9th grade	14	14	14	-	480	20	8	20	-	480	18	14	18	-	480
Cooking	8	6	6	1	6	8	6	6	1	1	10	6	6	1	6

Table 10.11: Initial and final number of corners and run time for the $9N$ instances with the window heuristic

	$9N_1$					$9N_2$					$9N_3$				
	C_I	C_F	C_F^W	T_B	T	C_I	C_F	C_F^W	T_B	T	C_I	C_F	C_F^W	T_B	T
10th grade offices	16	4	4	292	292	12	6	6	2	30	12	6	6	151	480
Music area	12	4	4	305	305	10	4	4	43	43	10	6	6	6	241
9th grade offices	14	6	6	55	480	12	6	6	3	29	10	4	4	4	4
Gym	14	6	6	30	253	14	4	4	30	30	10	4	4	15	15
Library	4	4	4	1	1	6	4	4	1	1	6	4	4	1	1
9th grade	20	16	20	-	480	16	6	16	-	480	12	12	12	-	480
Science	8	4	4	1	1	12	4	4	1	1	6	6	6	1	1
Administration	6	6	6	1	1	6	6	6	1	3	8	6	6	1	4
10th grade	12	12	12	-	480	14	14	14	-	480	16	16	16	-	480

Figure 10.8 shows the average run time for each neighbourhood in the $9N$ instances without the window heuristic, T_B, and with the window heuristics, T_B^W. The test results show that this heuristic increases the run time of the model quite significantly. A possible explanation is that the increased complexity of the objective function has a negative impact on the efficiency of the commercial solvers heuristics. The increased complexity prevents the model from obtaining solutions to the *9th grade* neighbourhood, which explains the increase in the number of corners.

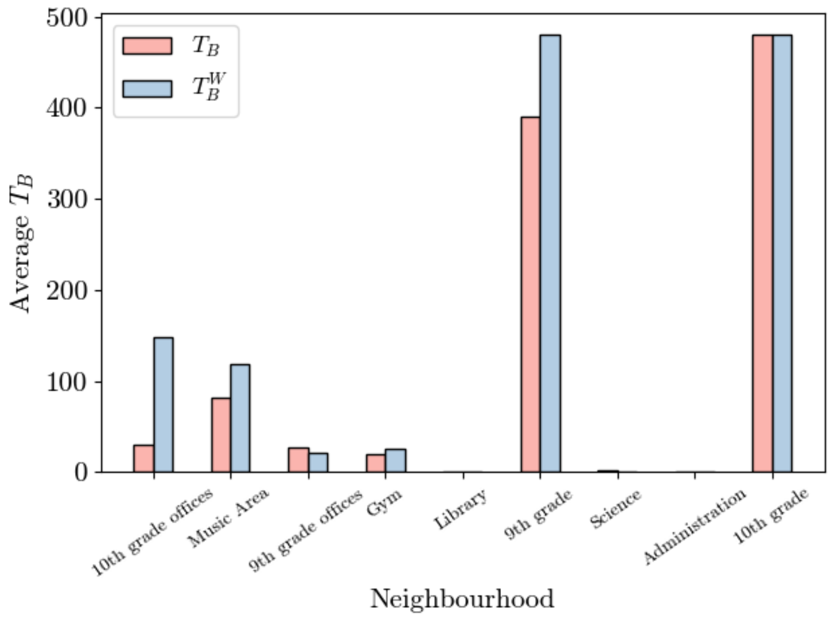


Figure 10.8: Average run time for each neighbourhood of the $9N$ instances, with, T_B^W, and without, T_B, the window heuristic

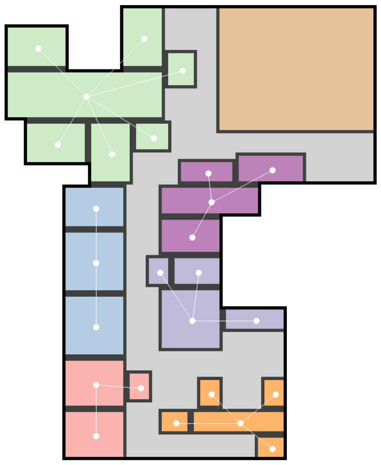
The majority of the rooms with window requirements are classrooms located in the

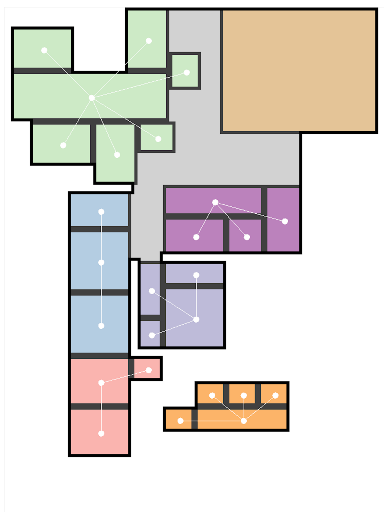
9th grade and *10th grade* neighbourhood. Since the model fails to find feasible solutions for these neighbourhoods, the resulting solutions cannot be used to evaluate the heuristic. Hence, it cannot be concluded from these tests whether or not the heuristic works as intended. Preliminary tests show that the two heuristics that are left to be tested speed up the model significantly. It is likely that including these heuristics will allow obtaining feasible solutions for the *9th grade* and *10th grade* neighbourhoods within the time limit, while keeping the window heuristic. The window heuristic is therefore kept for subsequent tests, and is reevaluated if a solution to these neighbourhoods are found. If the model fails to find feasible solutions within the time limit, this heuristic can be removed to speed up the model.

10.7 Lock hallways

Previous tests reveal that the model struggles to maintain connectivity. Section 6.7.2 addresses this challenge, and proposes a heuristic. With the lock hallways heuristic implemented, the model restricts rooms from completely covering an area that has been deemed desirable for connectivity in the layout from stage one.

Figure 10.9 visually illustrates the impact of locking hallways. 10.9 (a) shows the initial layout of instance $6N_2$, while (b) and (c) show the resulting layout of instance $6N_2$ before and after adding the lock hallways heuristic, respectively. Note that the layout in (b) is the output of the model when applying the window access heuristics, which did not find a feasible solution for *9th grade* 🟩. Recall that the lock hallways heuristic finds the longest stretch of the main room in the input layout which is adjacent to a hallway and prevents the model from placing the remaining rooms such that they completely cover this stretch. Consider the neighbourhood *Gym* 🟪. In the layout in Figure 10.9 (a), the longest stretch adjacent to a hallway from its main room is on the west side. Thus, including the lock hallways heuristic restricts the model from placing the rooms such that they completely cover the west wall of the main room. As Figure 10.9 (c) shows, the rooms of the neighbourhood are then placed on the east side. Consequently, this change allows for a hallway on the west side, which connects the neighbourhoods *Gym* 🟪, *9th office* 🟧 and *Cooking* 🟥, resulting in fulfilling the connectivity requirement. Additionally, the neighbourhoods in (c) contain the same number of corners as in (b). Hence, the heuristic does not adversely affect the main objective of the model in this particular case.


(a) Initial layout


(b) Without the lock hallways heuristic

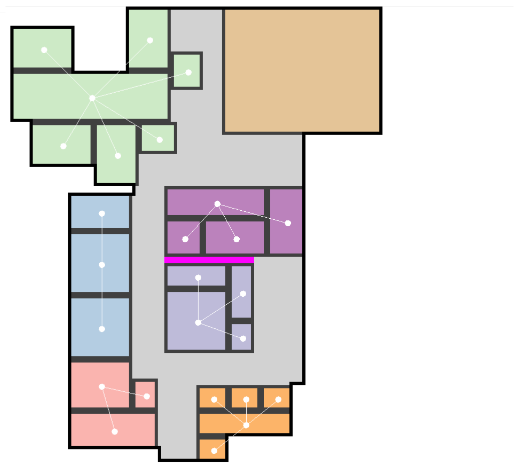

(c) With the lock hallways heuristic

Figure 10.9: Layouts illustrating the effect of the lock hallway heuristic on test instance $6N_2$

More generally, the heuristic tends to have a negative impact on the number of corners in the neighbourhoods. Figure 10.10 shows the total number of corners in each instance with and without the heuristic, denoted TC_F^H and TC_F respectively. The figure shows that the number of corners is unchanged for instance $6N_1$ and $6N_2$, and slightly worsened for the remaining instances. As locking the hallways restricts the feasible positions of the rooms, it is expected that desirable solutions in terms of corners are excluded from the solution space.

To assess whether or not the lock hallways heuristic performs as intended, the connectivity scores for the different model versions are compared. Table 10.12 shows the connectivity score of all model versions for all the six test instances. The basic model

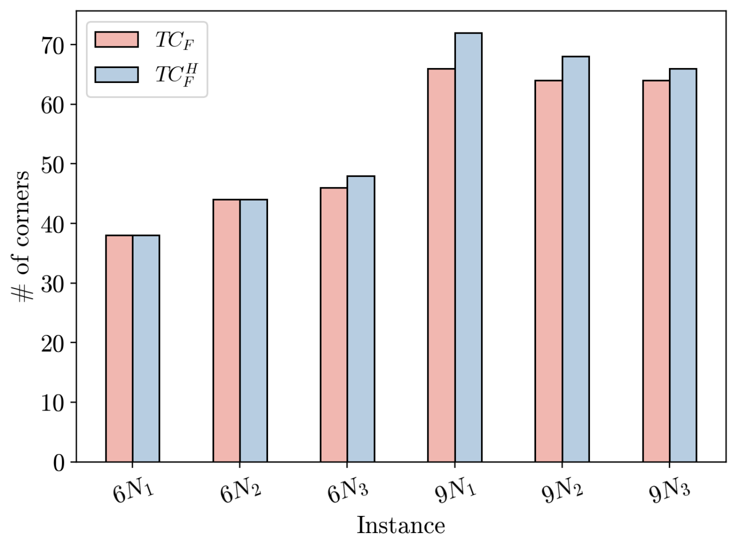


Figure 10.10: The total number of corners in each test instance with, TC_F^H, and without, TC_F, the lock hallways heuristic

is denoted basic, and as the heuristics are implemented, the model is denoted by the last added heuristic. Table 10.12 shows that including the lock hallways heuristic outperforms the previous versions. From these results, it can be concluded that the heuristic performs as intended. As mentioned, connectivity is often achieved by stage three of the algorithm, even when several neighbourhoods are not connected after stage two. On the other hand, when every neighbourhood is connected after stage two, it is more likely that the local search can locate efficient hallways in the final layout and minimize the number of exterior corners. It is, therefore, desirable that neighbourhoods are connected after stage two.

Table 10.12: Connectivity score for the different model versions across all test instances

	Connectivity					
	$6N_1$	$6N_2$	$6N_3$	$9N_1$	$9N_2$	$9N_3$
Basic	1	0	1	6	4	2
Lock main room	1	3	1	3	2	4
Window	1	3	0	2	0	2
Lock hallways	1	0	1	0	0	1

The lock hallways heuristic successfully improves the connectivity score directly, and yields solutions for which the local search in stage three is more likely to generate desirable layouts. Hence, the benefit of improved connectivity management outweighs the slight increase in the number of corners. Additionally, the heuristics impact on the run time is insignificant. See Section E.2 for time complexity results. Consequently, the heuristic is included in subsequent tests.

10.8 Split neighbourhood

To enhance the performance of the model to a level where it is able to solve the largest neighbourhoods, the split neighbourhood heuristic described in Section 6.7.5 is implemented. Results from preliminary testing suggest that it is beneficial to split neighbourhoods consisting of five or more rooms. Neighbourhoods with less than five rooms are handled as before. Consequently, for the $6N$ and $9N$ instances, only the neighbourhoods *9th grade*, *9th grade offices*, *10th grade* and *10th grade offices* are split. Table 10.13 and 10.14 show that adding the heuristic makes the model capable of finding feasible solutions for all neighbourhoods across all test instances. Additionally, the model terminates before the maximum given time for all neighbourhoods, meaning that optimality is guaranteed. However, the heuristic first optimizes for a subset of the rooms in a neighbourhood, fixes these in place and then optimizes for the remaining rooms. Consequently, solved to optimality now means that the placement of the initial subset of rooms is optimal, and the placement of the rooms in the final subset given the placement of the first rooms is optimal. Hence, the resulting solution might not be the optimal solution if the model considers all rooms in the neighbourhood at once. The times presented in Table 10.13 and 10.14 are the total time spent placing all rooms in each neighbourhood.

Table 10.13: Initial and final number of corners and run time for the $6N$ instances adding the split neighbourhood heuristic

	$6N_1$					$6N_2$					$6N_3$				
	C_I	C_F	C_F^S	T_B	T	C_I	C_F	C_FS	T_B	T	C_I	C_F	C_F^S	T_B	T
Music area	14	4	4	19	19	12	4	4	114	114	10	4	6	15	98
9th grade offices	16	6	6	20	20	12	6	6	31	32	12	8	6	39	39
Gym	12	4	4	16	16	10	4	4	75	75	12	8	6	1	10
Arts & Crafts	8	4	4	14	15	4	4	4	1	1	8	4	4	3	3
9th grade	14	14	14	77	120	20	20	6	124	300	18	18	6	85	229
Cooking	8	6	6	4	4	8	6	6	1	1	10	6	6	1	4

Table 10.14: Initial and final number of corners and run time for the $9N$ instances adding the split neighbourhood heuristic

	$9N_1$					$9N_2$					$9N_3$				
	C_I	C_F	C_F^S	T_B	T	C_I	C_F	C_F^S	T_B	T	C_I	C_F	C_F^S	T_B	T
10th grade offices	16	4	6	6	6	12	6	8	41	41	12	6	6	208	208
Music area	12	4	4	36	36	10	4	4	250	250	10	6	6	6	316
9th grade offices	14	6	6	2	2	12	6	6	18	235	10	4	4	14	14
Gym	14	6	6	20	24	14	4	6	34	97	10	4	6	83	84
Library	4	4	4	1	1	6	4	4	1	1	6	4	4	1	1
9th grade	20	20	8	431	431	16	16	8	466	466	12	12	6	328	450
Science	8	4	6	1	2	12	4	8	1	2	6	6	6	1	1
Administration	6	6	6	1	3	6	6	6	1	1	8	6	6	1	5
10th grade	12	12	8	267	465	14	14	8	280	459	16	16	8	422	422

To evaluate whether or not the limit of five or more rooms is appropriate, the impact on the neighbourhoods that are split is assessed. Figure 10.11 shows the total number of corners for the neighbourhoods with five and seven rooms for all test instances with, TC_F^S, and without, TC_F, the heuristic. The neighbourhoods containing five rooms are *9th grade offices* and *10th grade offices* and the neighbourhoods containing seven rooms are the *9th grade* and *10th grade*. The figure shows that the number of corners for the neighbourhoods with five rooms are unaffected by the heuristic, while for seven rooms it is significantly improved. The improvement for the neighbourhoods containing seven rooms is expected as the model previously struggled to find solutions for these neighbourhoods. For the neighbourhoods with five rooms, the initial hypothesis was that placing rooms in two steps adversely affects the solutions, as the previous model finds solutions for these neighbourhoods. The results, however, indicate that splitting the neighbourhoods does not affect the solutions in terms of corners at all. In addition, as the run time for the neighbourhoods with five rooms decreases drastically with the split heuristic, the limit is kept at five rooms.

Assessing the impact of the heuristic on the objectives used in stage one and three, the connectivity and window access scores are most notable. See Appendix E for results on the additional objectives. Table 10.15 compares the connectivity score obtained by the current model, denoted split, and the previous model version denoted lock hallways.

Table 10.15: Connectivity score for the different model versions across all test instances

	Connectivity					
	$6N_1$	$6N_2$	$6N_3$	$9N_1$	$9N_2$	$9N_3$
Lock hallways	1	0	1	0	0	1
Split	1	4	0	0	0	3

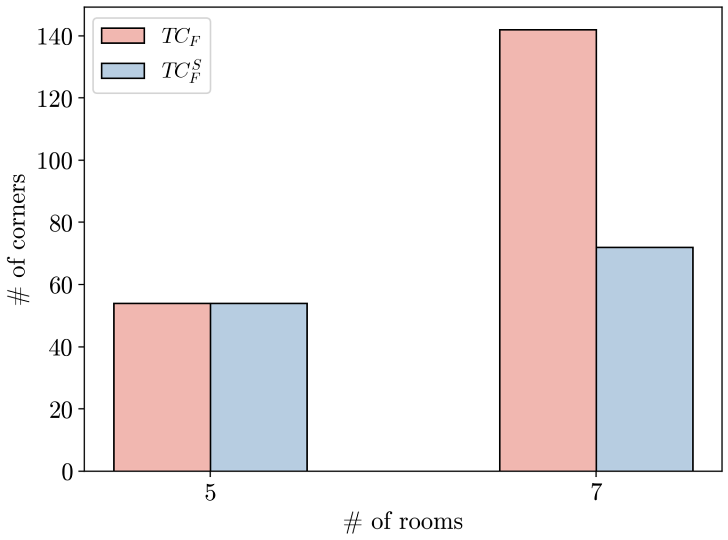
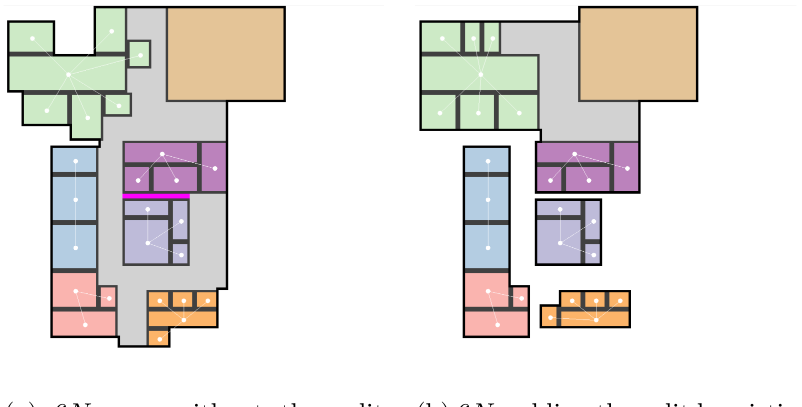


Figure 10.11: Total number of corners for the neighbourhoods containing five and seven rooms over all test instances with, TC_F^S, and without, TC_F, the heuristic.

The table indicates that the split heuristic has a negative impact on the connectivity score. However, the adverse impact on connectivity is instance specific. To exemplify, consider Figure 10.12 which shows the resulting layout of $6N_2$ when excluding the split heuristic in (a), and adding the heuristic in (b). Figure 10.12 (b) illustrates that as the *9th grade* 🟩 neighbourhood is solved, it is placed such that the hallways connecting four neighbourhoods is blocked. Consequently, the connectivity score increases from zero to four by implementing the heuristic. Still, connectivity can be restored by moving *9th grade* slightly to the left or the *Music area* 🟪 slightly down or to the right, which is a manageable task for the local search in stage three.



(a) $6N_2$ run without the split heuristic

(b) $6N_2$ adding the split heuristic

Figure 10.12: The resulting layouts after running the model with and without the split neighbourhood heuristic for instance $6N_2$. When adding the heuristics, connectivity is violated for several neighbourhoods.

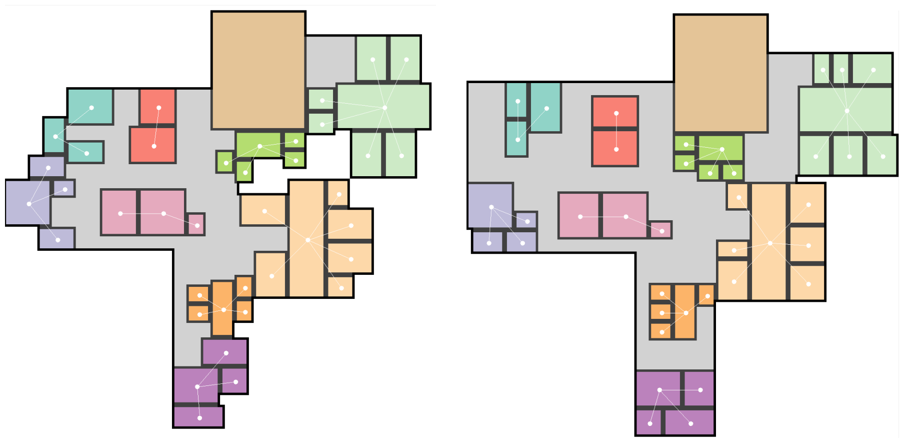
As mentioned, it is necessary to reevaluate the window objective heuristic when the model finds feasible solutions for the larger neighbourhoods. Table 10.16 shows the window access scores obtained by different model versions for all instances. As the previous test of the window heuristic is inconclusive, an additional model version is included, which contains all heuristics except the window heuristic, denoted ”Split w/o window”. The window heuristic can now be evaluated by comparing ”Split” and ”Split w/o windows”. First, observe that Split yields better window access scores than Lock hallways. Hence, including the split heuristic positively affects the window access for rooms. Nevertheless, the improvement on window access is not directly ensured by the split heuristic. It is rather a result of finding feasible solutions for the large neighbourhoods *9th grade* and *10th grade*, which in turn puts the window heuristic into effect. The table also shows that removing the window heuristic again yields worse window access scores, confirming the proficiency of the window heuristic. Additionally, as the model now comfortably finds feasible solutions for all neighbourhoods in all test instances, the increased run time as a result of implementing the window heuristic is considered insignificant. Consequently, the window heuristic is kept.

Table 10.16: Window access score for the different model versions across all test instances

	Window Access					
	$6N_1$	$6N_2$	$6N_3$	$9N_1$	$9N_2$	$9N_3$
Lock hallways	1	2	4	6	1	0
Split w/o window	1	1	4	6	1	0
Split	0	0	1	4	1	0

The split neighbourhood heuristic speeds up the model to the point where it can find feasible solutions and guarantee optimality for the largest neighbourhoods. Naturally, this causes a significant improvement on the main objective of the mathematical model, the minimization of corners. Hence, it is concluded that the split neighbourhood heuristic should be implemented in the model.

Figure 10.13 visually illustrates the desirable effects of the mathematical model. The figure shows the layout of instance $9N_2$ before and after running the final model with the desired heuristics. As the model finds feasible solutions for all neighbourhoods, they all take a geometric simplistic and more natural shape in the output layout. Additionally, the solution fulfils both connectivity and overlap, which makes it feasible. Furthermore, only one room which requires window access does not have it. The room without window access is the southwest room in the *9th grade* 🟩 neighbourhood. This room can gain window access by simply moving its neighbourhood northwards or eastwards.



(a) Initial layout. Input to the mathematical model

(b) Final layout. Output of the mathematical model run with all heuristics

Figure 10.13: Layouts before and after the final mathematical model is run for instance $9N_2$

To summarize, the model creates neighbourhoods with significantly fewer corners and more natural shapes by relocating and resizing rooms. However, the basic model has apparent deficiencies. First, the model is unable to handle the complexity of the problem instances. Secondly, it does not sufficiently take the objectives of stage one and three into account, which adversely impacts the complete layouts. Including the heuristics lock main room and split neighbourhood significantly increases the performance of the model, allowing to find solutions and prove optimality for the most complex neighbourhoods. Additionally, the window and lock hallways heuristics mitigate the models adverse impact on window access and connectivity, respectively. The final model containing the four heuristics above displays the desired behaviour, as it is able to solve all the test instances and sufficiently maintain the objectives of stage one and three. Consequently, the mathematical model in the final three-stage algorithm contains all heuristics considered in this chapter.

Chapter 11

Technical study, local search

This chapter tests the third and final stage of the solution method. The study tests and compares different values for the search area length of the local search, as well as different approaches to select the order of the neighbourhoods. The objectives of the local search were introduced in Chapter 7, and are for convenience repeated in Table 11.1. The fitness of the solutions generated in this stage is a weighted sum of these objectives.

Table 11.1: Objective variables of the local search

Objective variables, LS	
f_1	Overlap
f_2	Connectivity
f_3	Narrow hallways
f_5	Window access
f_6	Hallway area
f_8	Exterior corners

The hardware and software used to implement and test the local search are the same as for the memetic algorithm and specified in Table 11.2.

Table 11.2: Details of the computer hardware and software used for the local search

CPU	Intel Core i5 CPU @ 2.70GHz
RAM	8 GB
JAVA Version	11.0.4
Python Version	3.7
JAVA IDE	Intellij IDEA Version 2018 3.5
Python IDE	Pycharm Version 2017 2.2

11.1 Methodology

The input to the local search is a layout generated in stage two. All tests are performed on four test instances, two of which are generated from $6N$, and two from $9N$. Six search area lengths and three selection approaches are tested. The lengths are tested first. The tests are performed for each combination of instance and length using the random order selection of neighbourhoods. When subsequently testing the selection approaches the most desirable search area length, based on the test results. After examining the results, the length obtaining the most promising results is applied when testing the selection approaches. The same test approach holds for the selection approaches, where tests are conducted for each combination of instance and selection approach. Each combination is tested by running the algorithm ten times. Both the mean objective values of the runs and the best run are chosen as the basis for comparison. Preliminary testing reveals a low time complexity of the local search. This allows the three-stage algorithm to run the local search multiple times, picking the best solution as the final layout. Thus, the best run, which is the run obtaining the lowest fitness score, is interesting to study. However, the objective values of a single run might not give an accurate indication of how well a combination generally performs compared to another. Comparing the mean objective values of different combinations can help determine which solution approach to apply. Thus, the mean is also studied. The final solution generated in the last stage should be feasible. Therefore, only the runs producing feasible solutions are considered in the mean and best objectives.

11.2 Test instances

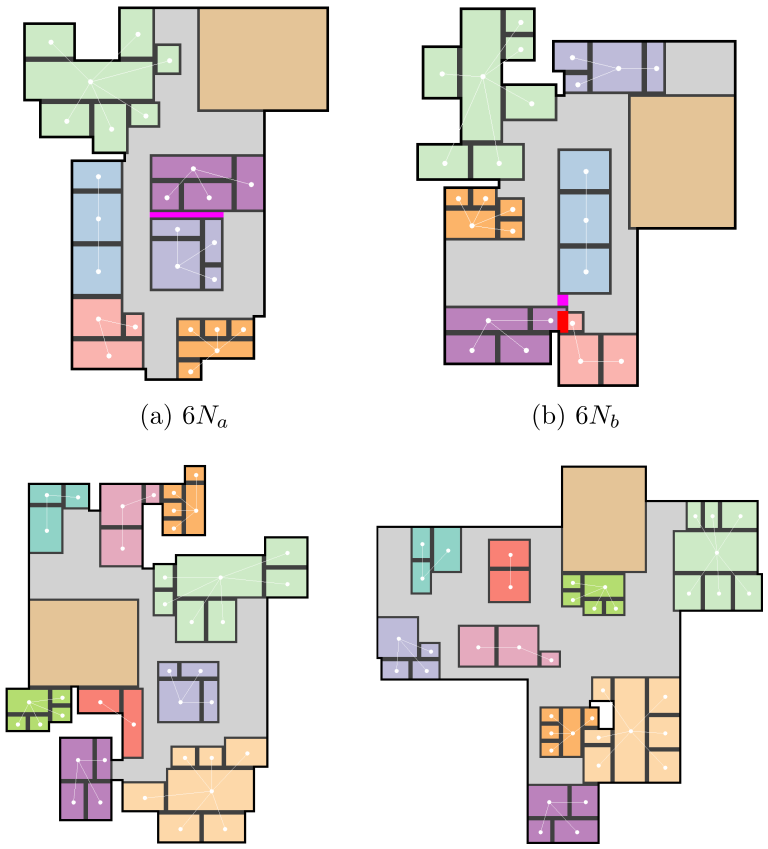
To accurately determine the performance of the local search, tests are performed on a variety of layouts generated from instance $6N$ and $9N$. These instances are outputs from stage two and are carefully picked to form a representative collection, where the instances contain different properties and qualities. Table 11.3 presents the test instances with their corresponding initial fitness and objective values. As the table shows, the instances vary from being feasible, infeasible, and satisfactory. Recall that a feasible solution fulfils objective f_1 and f_2, while a satisfactory solution also fulfils objective f_3 and f_5. For the rest of this chapter, the objectives will be referred to by the abbreviations in Table 11.4 due to readability. Figure 11.2 provides a visualization of the four test instances. To ensure diversity of the instances, some test instances contain neighbourhoods which were insufficiently solved by the mathematical model. This holds for *9th grade* 🟩 in (a) and (b) and *10th grade* 🟧 in (c).

Table 11.3: Initial fitness and objective values of the test instances

		Objective variables					
Instance	Fitness	f_1	f_2	f_3	f_5	f_6	f_8
$6N_a$	672.2	0.0	0.0	14.0	2.0	29.2	24.0
$6N_b$	968.6	8.0	0.0	4.0	2.0	30.6	30.0
$9N_a$	2183.2	0.0	3.0	0.0	4.0	27.6	48.0
$9N_b$	411.6	0.0	0.0	0.0	0.0	38.6	24.0

Table 11.4: Objective abbreviations of the local search objectives

Objective abbreviations	
O	Overlap
C	Connectivity
NH	Narrow hallways
WA	Window access
HA	Hallway area
EC	Exterior corners



(a) $6N_a$ (b) $6N_b$

(c) $9N_a$ (d) $9N_b$

Figure 11.1: Initial layouts of the test instances used as input in the technical study of the local search

11.3 Search area length

In the following, the length of the search area is tested. The lengths evaluated are $\{2, 5, 10, 12, 15, 18\}$, which are chosen based on preliminary testing. The test results for the average and best run for each combination of instance and length are shown in Table 11.5 and 11.6, respectively. Table 11.5 shows the mean objective values of each combination, while Table 11.6 shows the objective values of the best run of an instance for each length. Both tables display the objective values of the instances before conducting the local search. A feasible run implicates a connectivity and overlap objective score of zero. As only the feasible runs are examined, these rows are excluded from the tables. For both tables, the lowest fitness score obtained for each instance is coloured in blue.

Table 11.5: The mean objective values for different search area lengths. The best average fitness for each instance is marked in blue.

			Search area length					
Instance	**Objective**	**Pre LS**	**2**	**5**	**10**	**12**	**15**	**18**
$6N_a$	Fitness	672.2	376.5	345.8	348.9	344.6	205.5	284.7
	NH	14.0	0.0	0.0	0.0	0.0	0.3	0.0
	WA	2.0	1.1	1.0	1.0	1.0	0.1	0.6
	HA(%)	29.2	14.7	13.9	16.2	15.8	21.7	19.8
	EC	24.0	30.9	31.7	27.7	26.9	24.9	27.4
$6N_b$	Fitness	968.6	504.7	320.0	212.4	273.2	172.5	243.7
	NH	4.0	0.3	0.0	0.0	0.0	0.0	0.0
	WA	2.0	1.6	0.7	0.3	0.6	0.0	0.4
	HA(%)	30.6	23.3	20.8	16.7	17.9	20.0	17.7
	EC	24.0	25.7	27.4	29.1	27.7	27.7	27.7
$9N_a$	Fitness	2183.2	457.7	289.7	263.2	294.0	298.7	373.1
	NH	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	WA	0.0	1.0	0.1	0.1	0.1	0.3	0.7
	HA(%)	38.6	24.1	24.0	21.7	25.5	22.5	22.4
	EC	48.0	32.0	34.0	34.3	29.7	31.7	31.3
$9N_b$	Fitness	441.6	249.1	221.0	229.5	209.8	224.6	214.9
	NH	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	WA	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	HA(%)	38.6	24.5	21.6	22.8	20.5	22.3	20.7
	EC	24.0	27.1	26.9	25.7	26.3	25.7	27.7

As seen in Table 11.5, a length of 15 yields the best results on average for two out of four instances. Compared to the other lengths, the LS generates considerably better solutions for the $6N$ instances with a length of 15, mainly due to its ability to fulfil the window access objective. Table 11.6 shows a similar pattern. Again, using a length of 15 results in the lowest fitness for half of the instances. Additionally, the best fitness of instance $6N_a$ with length 15, is almost equal to the best fitness obtained with a length of 18. Overall, the tables reveal that increasing the length yields better scores. Comparing the tables show that with a greater length, the LS performs better on the best run compared to the shorter lengths, while on average, slightly shorter lengths yields better fitness. For instance does the LS with length 18 perform better in comparison to the other lengths for the best run compared to the average run. This may be explained by the greediness of the LS with a large search area length. A greater length provides more possibilities when moving a neighbourhood, but the LS may make large relocations of a single neighbourhood that are not desirable for the other neighbourhoods. The solutions for the $9N$ instances show smaller differences between the smaller and greater lengths, both for the best and average runs. As $9N$ contains more neighbourhoods, increasing the length might not have the same effect as for smaller instances, because more of the search area is likely to be covered by other neighbourhoods.

Table 11.6: The objective values of the best run for each combination of instance and length. The best fitness obtained for each instance is marked in blue.

			Search area length					
Instance	**Objective**	**Pre LS**	**2**	**5**	**10**	**12**	**15**	**18**
$6N_a$	Fitness	672.2	330.8	334.2	336.4	334.6	160.6	160.2
	NH	14.0	0.0	0.0	0.0	0.0	0.0	0.0
	WA	2.0	1.0	1.0	1.0	1.0	0.0	0.0
	HA(%)	29.2	12.9	11.6	14.8	14.4	19.6	18.8
	EC	24.0	28.0	32.0	26.0	26.0	24.0	26.0
$6N_b$	Fitness	968.6	380.8	138.2	140.2	138.8	124.6	144.4
	NH	4.0	0.0	0.0	0.0	0.0	0.0	0.0
	WA	2.0	1.0	0.0	0.0	0.0	0.0	0.0
	HA(%)	30.6	21.9	13.3	16.3	12.5	10.7	16.2
	EC	30.0	26.0	30.0	24.0	24.0	30.0	26.0
$9N_a$	Fitness	2183.2	429.2	236.0	220.0	235.0	212.4	219.2
	NH	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	WA	4.0	1.0	0.0	0.0	0.0	0.0	0.0
	HA(%)	27.6	21.1	22.4	20.0	21.7	19.7	20.5
	EC	48.0	32.0	30.0	32.0	32.0	30.0	30.0
$9N_b$	Fitness	441.6	232.6	188.4	188.4	152.6	161.4	190.2
	NH	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	WA	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	HA(%)	38.6	23.1	19.8	19.8	13.5	17.2	18.3
	EC	24.0	26.0	20.0	20.0	26.0	24.0	26.0

As the mathematical model in stage two may produce infeasible solutions, the local search must be able to turn an infeasible solutions into a feasible one. Thus, the fraction of runs in which the local search generates feasible solutions is key. Table 11.7 shows the fraction of feasible solutions for each combination of length l and test instance. The results show that the LS generates solely feasible solutions for almost all combinations. The only exception is instance $9N_a$, where its initial violation of connectivity likely is the cause. Lengths of 2 and 18 yield feasible solutions for the $9N_a$ instance 57% and 43% of the runs, respectively. A length of 2 possibly creates a too small search area for the LS to obtain feasibility, while a local search with length 18 may take too greedy choices aiming to fulfil connectivity.

A time study is conducted and shows that the longest average run lasts for about nine minutes. The results are given in Table F.1 in Appendix F. Due to the low time complexity, time does not affect the choice of length. Based on the objective value results and the algorithms ability to produce feasible solutions, a search area length of 15 is chosen for the local search.

Table 11.7: The fraction of feasible solutions generated for each combination of instance and length

	Search area length l					
Instance	**2**	**5**	**10**	**12**	**15**	**18**
$6N_a$	1.0	1.0	1.0	1.0	1.0	1.0
$6N_b$	1.0	1.0	1.0	1.0	1.0	1.0
$9N_a$	0.57	1.0	1.0	1.0	1.0	0.43
$9N_b$	1.0	1.0	1.0	1.0	1.0	1.0

11.4 Selection approach of neighbourhoods

In the following, different selection approaches of neighbourhoods are tested and evaluated. Two heuristics are implemented and compared to the random order selection. The heuristics were introduced in Chapter 5.7 and is for convenience repeated in Table 11.8 along with the random order selection.

Table 11.8: Three alternative selection approaches

Number	Selection approaches
1	Order by largest neighbourhood
2	Order by neighbourhoods violating objectives f_1, f_2 and f_5
3	Random order

A study is conducted on both the fraction of feasible solutions and the time complexity of the different approaches. The results are presented in Appendix F. Each combination of selection approach and test instance yields solely feasible solutions. The results also indicate no practical difference in the time complexity of the different approaches.

The objective values for the average and best run for each combination of instance and selection approach are shown in Table 11.9 and 11.10, respectively. The numbers for the selection approach correspond with those in Table 11.8. As the runs considered are all feasible, the overlap and connectivity objectives are excluded from the tables. The largest neighbourhood heuristic is deterministic, and is only run once. Thus, the objective values are similar in the two tables, as the mean and best objective values do not differ.

The tables show that the random order selection has the best performance. As instance $9N_b$ is initially feasible and fulfils the window access objective, heuristic 2 turns into a random order approach for this instance, as described by Algorithm 4 in Section 7.2. Thus, the random order selection performs best on three out of four instances, for both the mean and best runs. It is also observed from Table 11.10

that random order performs equally good or strictly better on all objectives for three instances. An unexpected result is the performance of heuristic 2. This heuristic achieves the most satisfying results for instance $6N_a$ both on average and for the best run. As this initial instance fulfils both the connectivity and overlap objective, as opposed to instance $6N_b$ and $9N_a$, the heuristic does not perform best where expected. Heuristic 1 obtains the worst results, possibly due to its determinism, yielding the same order in each iteration. Based on the results, random order is chosen as the selection approach.

Table 11.9: The mean objective value for different selection approaches. The best average fitness obtained for each instance is marked in blue.

			Selection approach		
Instance	**Objective**	**Pre LS**	**1**	**2**	**3**
$6N_a$	Fitness	672.2	339.0	185.0	205.5
	NH	14.0	0.0	0.0	0.3
	WA	2.0	1.0	0.0	0.1
	HA(%)	29.2	13.4	23.3	21.7
	EC	24.0	30.0	24.0	24.9
$6N_b$	Fitness	968.6	331.2	276.0	172.5
	NH	4.0	0.0	0.0	0.0
	WA	2.0	1.0	0.6	0.0
	HA(%)	30.6	12.9	17.6	20.0
	EC	30.0	28.0	29.4	27.7
$9N_a$	Fitness	2183.2	309.4	327.5	298.7
	NH	0.0	0.0	0.1	0.0
	WA	4.0	0.0	0.4	0.3
	HA(%)	27.6	29.5	22.3	22.5
	EC	48.0	30.0	32.3	31.7
$9N_b$	Fitness	441.6	236.6	215.1	224.6
	NH	0.0	0.0	0.0	0.0
	WA	0.0	0.0	0.0	0.0
	HA(%)	38.6	22.4	21.2	22.3
	EC	24.0	30.0	26.0	25.7

Table 11.10: The objective values of the best run for each combination of instance and selection approach. The best fitness obtained for each instance is marked in blue.

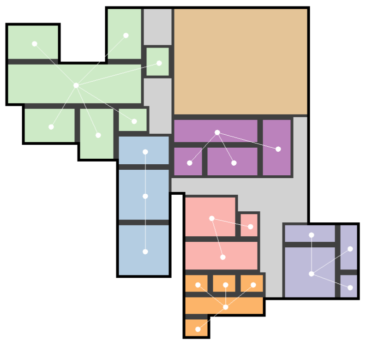
			Selection approach		
Instance	**Objective**	**Pre LS**	**1**	**2**	**3**
$6N_a$	Fitness	672.2	339.0	147.6	160.6
	NH	14.0	0.0	0.0	0.0
	WA	2.0	1.0	0.0	0.0
	HA(%)	29.2	13.4	17.5	19.6
	EC	24.0	30.0	24.0	24.0
$6N_b$	Fitness	968.6	331.2	144.6	124.6
	NH	4.0	0.0	0.0	0.0
	WA	2.0	1.0	0.0	0.0
	HA(%)	30.6	12.9	13.6	10.7
	EC	30.0	28.0	32.0	30.0
$9N_a$	Fitness	2183.2	309.4	239.0	212.4
	NH	0.0	0.0	0.0	0.0
	WA	4.0	0.0	0.0	0.0
	HA(%)	27.6	29.5	21.6	19.7
	EC	48.0	30.0	34.0	30.0
$9N_b$	Fitness	441.6	236.6	196.4	161.4
	NH	0.0	0.0	0.0	0.0
	WA	0.0	0.0	0.0	0.0
	HA(%)	38.6	22.4	18.4	17.2
	EC	24.0	30.0	28.0	24.0

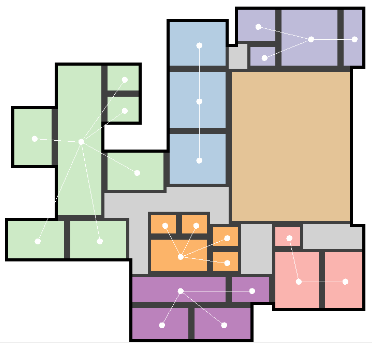
The technical study certifies the desired behaviour of the local search. The LS manages to turn infeasible solutions feasible, turn feasible solutions satisfactory, and improve satisfactory solutions in terms of building costs. Figure 11.2 visualizes the final layouts of the test instances after employing a local search with a search area length of 15 and random selection order of neighbourhoods. Table 11.11 compares the fitness and objective values of the instances before and after the local search, denoted "Pre" and "Post". By running the local search, all instances become satisfactory, while decreasing the hallway area substantially. For instance $9N_b$, which is initially satisfactory, the hallway area decreases by 67%.

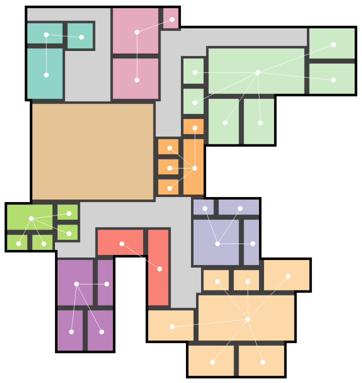
Additionally, the number of exterior corners of instance $9N_a$, which initially contains notably more exterior corners than the other instances, decreases drastically. For the other instances, the number remains unchanged. This can be explained by their low initial number of exterior corners, and obtaining even fewer corners may not be possible without affecting the other objectives negatively.

Table 11.11: Fitness and objective values of the test instances before and after the local search

Instance		**Fitness**	**O**	**C**	**NH**	**WA**	**HA(%)**	**EC**
			Objective variables					
$6N_a$	*Pre*	672.2	0.0	0.0	14.0	2.0	28.2	24.0
	Post	160.6	0.0	0.0	0.0	0.0	19.6	24.0
$6N_b$	*Pre*	968.6	8.0	0.0	4.0	2.0	30.6	30.0
	Post	124.6	0.0	0.0	0.0	0.0	10.7	30.0
$9N_a$	*Pre*	2183.2	0.0	3.0	0.0	4.0	27.6	48.0
	Post	212.4	0.0	0.0	0.0	0.0	19.7	30.0
$9N_b$	*Pre*	411.6	0.0	0.0	0.0	0.0	38.6	24.0
	Post	181.4	0.0	0.0	0.0	0.0	17.2	24.0


(a) $6N_a$


(b) $6N_b$


(c) $9N_a$

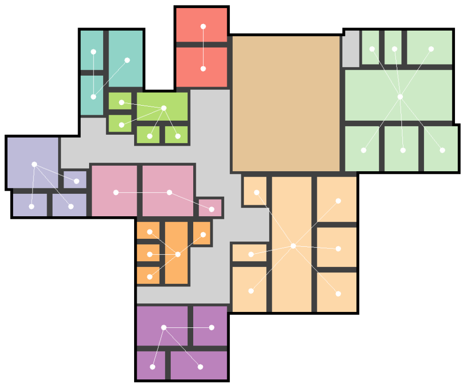

(d) $9N_b$

Figure 11.2: The final layouts of the test instances after the local search is conducted

Chapter 12

Performance test

This chapter evaluates the performance of the three-stage algorithm as a solution method for the SLP. The algorithm is run five times for each of the six RSDs presented in Chapter 8, giving a total of 30 solutions to the SLP. For all performance tests, the three stages of the algorithm are implemented with the settings that were concluded to be the most desirable in each of the technical study chapters, along with a stopping condition. A population size of 50 is chosen for the MA, and the stopping condition is set at 300 generations. Stage two, the mathematical model, is given a time limit of eight hours. For each test, stage three is run 15 separate times, each generating a unique solution. The best solution, in terms of weighted fitness score, is the final result of the test. With these stopping conditions, stage one and stage three take less than an hour each for the most extensive test instances. Stage two generally uses close to the full eight hours for all instances. The tests are conducted on the computing cluster *Solstorm*, provided by the Department of Industrial Economics and Technology Management at NTNU. The hardware used for the performance tests are specified in Table 12.1.

Table 12.1: Details of the computer hardware used for the performance tests

CPU	2.4GHz Intel Xeon Gold 5115 CPU – 10 core
RAM	96 Gb

Since the purpose of this thesis is to generate layouts meant to be iterated on by architects, it is desirable but not an absolute requirement that they fulfil the specifications in the RSD. The results show that all 30 solutions sufficiently meet these requirements and that they are all feasible. Thus, the solutions are further assessed on how well they minimize building costs and their geometric simplicity.

12.1 Objectives

The algorithm seeks to minimize building area and the number of exterior corners, as they are the two main cost drivers in a layout design from an architectural point of view. Recall that since the RSD lists the rooms with a suggested size, it is the hallway area that is subject to minimization by the three-stage algorithm.

The best, average and worst hallway area score from the five tests performed on the different RSD is presented in column HA in Table 12.2. Hallway area is almost non-existent for solutions generated from the $2N$ and $3N$ RSDs. This is a reasonable result as these solutions place the neighbourhoods adjacent to the hub, thus reducing the need for hallways. For the other RSDs, the average percentage of hallway area is between 14.2% and 24.7%, increasing with the number of neighbourhoods. The difference between the best and worst solutions in terms of hallway area also increases with the number of neighbourhoods. The hallway area percentage in Levanger Middle School is 39%. Comparing this to the test results for the $9N$ instance, which is more complex in terms of total room size and number of neighbourhoods, the algorithm obtains a significantly lower hallway area percentage in its solutions. The average hallway area score in solutions for the $9N$ instance is 20.9 percentage points than Levanger Middle School. An explanation for some of this difference is that the algorithm does not consider the fact that some hallways will be subject to heavy traffic and therefore should be wider than the minimum limit set in the algorithm. Also, the algorithm uses the main room in each neighbourhood as an access point to a greater extent than the real-world Levanger Middle School. Still, after taking this into consideration, it can be concluded that the algorithm performs very well in terms of minimizing building area.

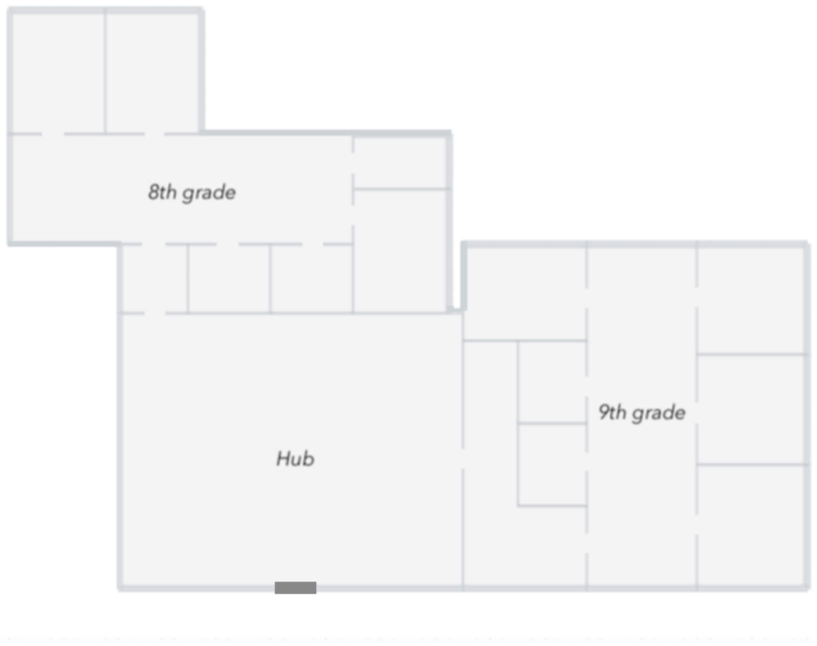
Table 12.2: Hallway area and exterior corner results from performance tests

		Objectives	
RSD		**HA(%)**	**EC**
$2N$	*Best*	1.3	12.0
	Average	2.8	15.6
	Worst	4.4	18.0
$3N$	*Best*	1.9	12.0
	Average	3.1	14.0
	Worst	4.1	16.0
$6N$	*Best*	13.2	18.0
	Average	14.2	18.0
	Worst	16.5	22.0
$7N$	*Best*	15.0	16.0
	Average	18.2	17.6
	Worst	20.2	20.0
$9N$	*Best*	13.8	26.0
	Average	20.9	33.2
	Worst	25.7	46.0
$11N$	*Best*	19.2	24.0
	Average	24.7	35.8
	Worst	30.2	42.0

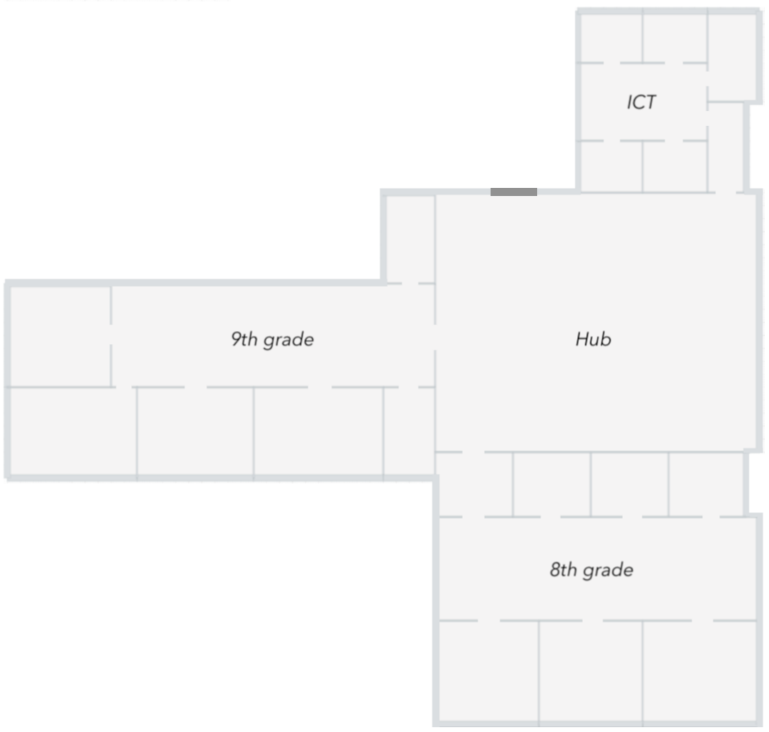
Column EC in Table 12.2 shows the number of exterior corners in the final solutions. As with the hallway area results, the best, average and worst value obtained in the performance tests are presented. Examining the average value for the various RSDs, the number of exterior corners for each of the two largest instances are almost double that of the four smaller instances. The results indicate that this objective becomes considerably more difficult to satisfy as the number of neighbourhoods increase over a certain threshold. The difference between the best and worst solutions for both $9N$ and $11N$ is large, with a gap of 20 and 18 exterior corners, respectively. The footprint of Levanger Middle School is a rectangle, thus it has only four exterior corners. Considering the test results in Table 12.2, it is clear that the current implementation of the three-stage algorithm is not suited to generate buildings with such a small number of exterior corners. A major reason for this is the strict implementation of door-neighbour requirements. Architects, in general, use these specifications in the RSD more as proximity guidelines than requirements. Allowing rectilinear rooms, as opposed to solely rectangular, will likely allow the algorithm to generate more efficient solutions in terms of exterior corners. At the same time, it will also increase the run time drastically. Still, to generate buildings with as few as four exterior corners for the largest RSDs, it is likely that fundamental changes to the three-stage algorithm is needed. For example, to initially determine the footprint of the building.

Figure 12.1 shows solutions for each of the six RSDs, generated by the three-stage algorithm. These are not necessarily the best in terms of hallway area or the number of exterior corners. When selecting these solutions, a subjective evaluation of the practicality and aesthetics in the resulting layout is included. Also, the algorithm does not differentiate between different exterior corners. By assessing the layouts manually, it becomes clear that small alterations can remove many of the exterior corners. An example is the small gap between the *8th grade* and *9th grade* neighbourhoods in Figure 12.1 (*a*). Since the solutions are meant for inspiration and a starting point the architect can iterate on, a manual assessment of the layout can take this into consideration.

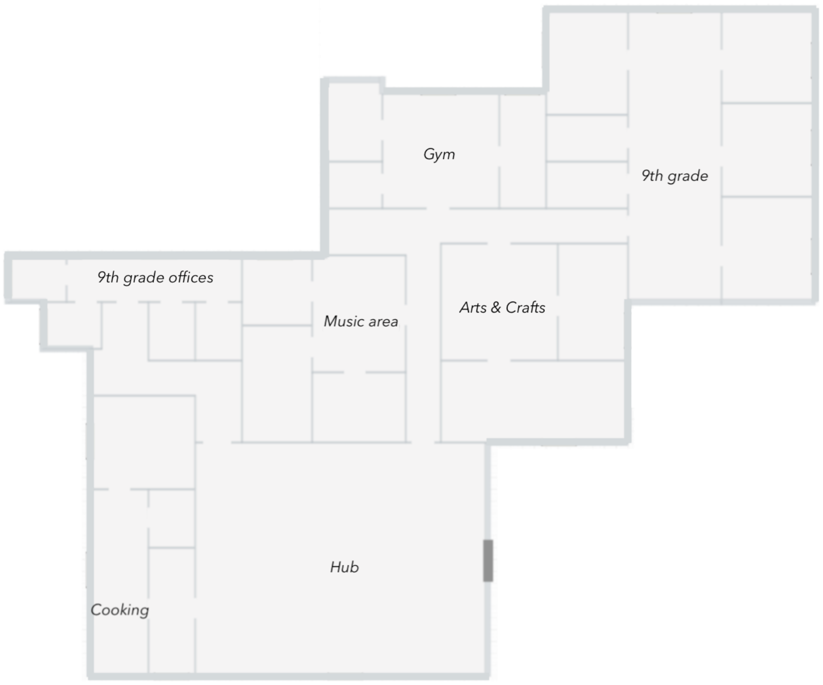
Lastly, the algorithm is assessed by studying the geometric simplicity of the layouts in Figure 12.1. A qualitative evaluation of their geometric simplicity is performed. Part of the evaluation is done by assessing the number of corners in the solutions, both interior and exterior. For all layouts, all individual neighbourhoods are geometrically simple with natural shapes and few corners. This result is expected since this is the main objective of the mathematical model, and the technical study revealed that the model finds optimal solutions for all neighbourhoods within the eight-hour time limit. The geometric complexity of the layouts grow at an increasing pace as the number of neighbourhoods increase. Comparing the $3N$ and $6N$ layouts in Figure 12.1 (*b*) and (*c*) shows there is quite a small difference in terms geometric complexity between these two layouts. This is a notable result as going from $3N$ to $6N$ constitutes a 100% increase in terms of neighbourhoods. The difference in complexity between the $9N$ layout and the $6N$ layout is greater, even though this only constitutes a 50% increase in the number of neighbourhoods. Loosening the restrictions on room sizes and door-neighbour requirements would likely let the three-stage algorithm produce layouts that is geometrically more simplistic.



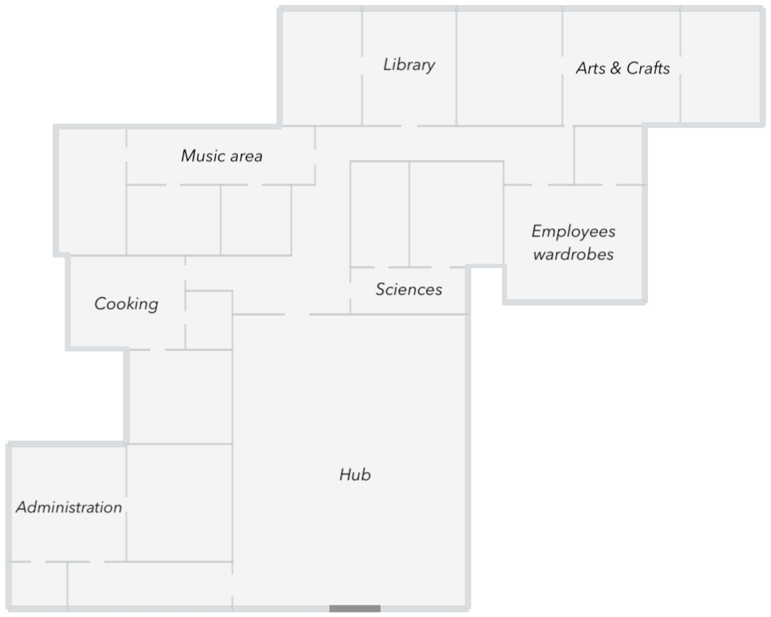
(a) $2N$


ICT
9th grade
Hub
8th grade

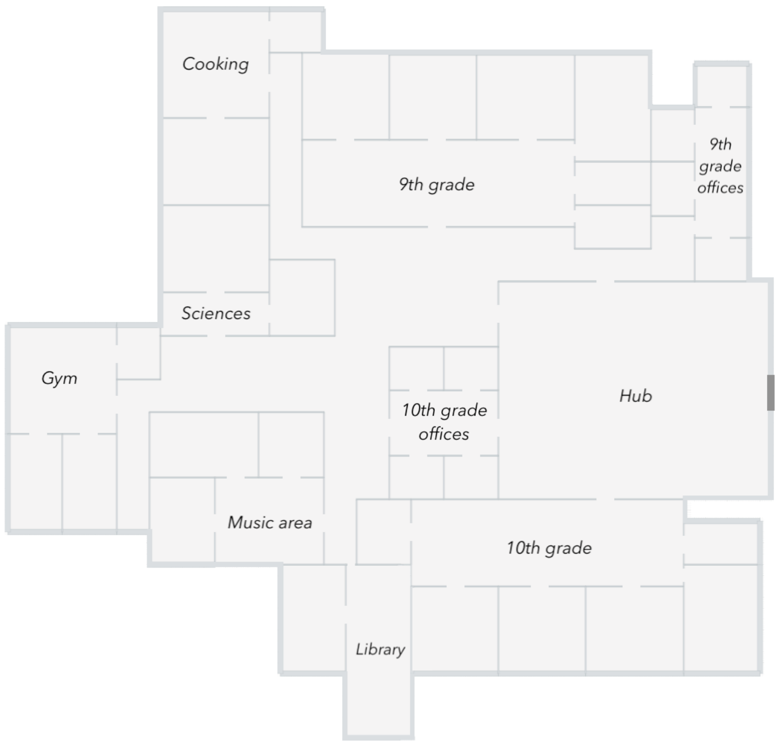
(b) $3N$


Gym
9th grade
9th grade offices
Music area
Arts & Crafts
Hub
Cooking

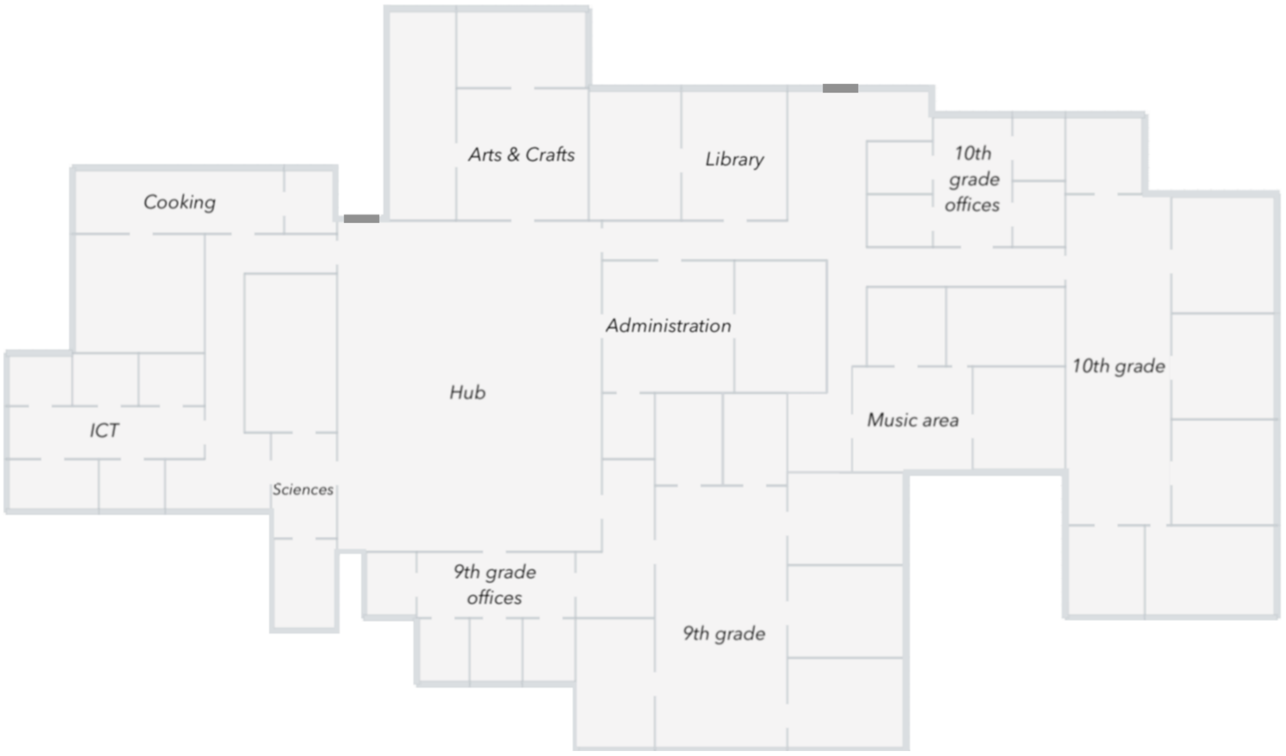
(c) $6N$


Library
Arts & Crafts
Music area
Employees wardrobes
Sciences
Cooking
Hub
Administration

(d) $7N$


Cooking
9th grade
9th grade offices
Sciences
Gym
10th grade offices
Hub
Music area
10th grade
Library

(e) $9N$



(f) $11N$

Figure 12.1: Blueprint of the layouts generated by the three-stage algorithm for each of the six RSDs

Chapter 13

Concluding remarks

This thesis has discussed the problem of developing a school layout, and proposed a multi-stage algorithm for generating layout designs. The process of designing a school layout is a multi-objective task that requires balancing conflicting goals and complying with complex regulations. Modelling the problem as an optimization problem makes it possible to generate multiple solutions that are guaranteed to meet quantitative regulations, while optimizing for desirable objectives. The algorithm proposed in this thesis can serve as a valuable decision support tool for architects. The solutions generated show a broad spectre of layouts, which architects can use as starting points in the layout developing process. The layouts comply with the requirements in the RSD and are optimized for building costs in terms of area and exterior corners.

The generation of school layouts has to our knowledge never been studied in the field of operations research. This thesis defines the School Layout Problem (SLP) as the problem of placing rooms and hallways to form a single floor building in two dimensions with no fixed footprint, and sheds light on the specific challenges the problem poses. Furthermore, the thesis concludes that little research is directly comparable or resembles the complexity of the SLP. As a school may contain more than 50 rooms on a single floor, the SLP requires a solution method that scales well. Thus, a robust three-stage algorithm is developed to generate layouts. The stages are carefully assembled, and allows for exploiting the strengths of the multiple solution approaches. The solution method consists of a memetic algorithm (MA), a mathematical model and a local search (LS). These compose stage one, two and three, respectively.

Based on the findings in the technical studies, it is apparent that concurrently considering all elements of the SLP is overly complex. Thus, the multi-stage approach is developed to successively consider the aspects, resulting in a more manageable complexity in each stage. In turn, this allows the stages of the algorithm to sufficiently handle their respective considerations, such that they collectively solve the SLP as a whole. Nevertheless, tests show that if a stage completely disregards the considerations taken in the other stages, it tends to impact the complete solution adversely.

Thus, adjustments to each stage are made to make them more compatible. For example, for the mathematical model, heuristics are implemented to maintain objectives which are fulfilled in stage one. These heuristics improve the algorithms ability to generate feasible solutions.

The implemented algorithm solves instances containing up to 50 rooms, which is plentiful for the single-floor SLP studied in this thesis. The resulting layouts meet the requirements in the RSDs while containing a small amount of hallway area and few exterior corners. In Levanger Middle School, the hallways make up 40% of the total building area. The layouts that are generated when applying the algorithm to instances of comparable sizes contain merely 20% hallways. Additionally, the number of exterior corners are minimized to the extent where the school buildings take a natural shape. However, there is still a need for improvement. The algorithm is unable to obtain fewer exterior corners as a result of rigidly adhering to the requirements in the RSD. Specifically, keeping the size of the rooms fixed in two of the stages and a strict implementation of proximity requirements.

The results show that the algorithm implemented generates a great variety of layout suggestions from the same RSD. Hence, the algorithm manages to map out parts of the solution space and turn a list of requirements into a suggested layout. To make the solutions more applicable to a real-world layout, the SLP should be extended, as aspects such as noise pollution and emergency exit access is not considered in this thesis. Still, the layouts adhere to the requirements in the RSD and can provide inspiration and a starting point for an architect. However, as designing a school layout is a highly practical problem with many implicit criteria, architects take artistic freedom in the layout designs. For instance, they may not completely adhere to adjacency requirements, but rather consider the proximity of rooms. As municipalities do not create RSDs to serve as input to optimization methods, considering the requirements in the RSD as absolute may not be best practice. Doing so is likely to exclude desirable solutions. Assessing which deviations are reasonable is not an appropriate task for an optimization method.

To make the layouts generated by the algorithm more effective as a decision support tool for architects, removing some of the simplifications made to the real-world SLP should be considered. Creating layouts for multiple floors and allowing rooms to take on more than just rectangular shapes are natural problem extensions. Changes to the algorithm could also provide even more helpful solutions. A natural extension of the current algorithm is to handle natural lighting and flow capacity requirements in a more sophisticated manner. Another improvement of the algorithm would be to revise the implementation of proximity requirements given in the RSD. The current enforcement is stricter than what is demanded in the RSD. A better way to manage these requirements would let the algorithm create layouts that satisfy the objectives to a greater extent.

Based on the results, the use of optimization techniques to generate school layouts appear to be promising. The implemented algorithm provides value to the architect in terms of decision support, as the layouts generated perform well on several quantitative and qualitative objectives. Hence, the results of this thesis advocate further research on the area.

Bibliography

Abotaleb, I., K. Nassar, and O. Hosny (2016). “Layout optimization of construction site facilities with dynamic freeform geometric representations”. In: *Automation in Construction* 66, pp. 15–28.

Bazaati, S. (2017). *Construction site layout planning considering traveling distance between facilities: Application of particle swarm optimization.*

Besbes, M., M. Zolghadri, R. Costa Affonso, F. Masmoudi, and M. Haddar (2020). “A methodology for solving facility layout problem considering barriers: genetic algorithm coupled with A* search”. In: *Journal of Intelligent Manufacturing* 31.3, pp. 615–640.

Brenner, U. (2018). “Soft packings of rectangles”. In: *Computational Geometry* 70-71, pp. 49–64.

Chatzikonstantinou, I. and E. Bengisu (2016). “Interior spatial layout with soft objectives using evolutionary computation”. In: *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2306–2312.

Chu, C. and E. Young (2004). “Nonrectangular Shaping and Sizing of Soft Modules for Floorplan-Design Improvement”. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 23, pp. 71–79.

Chwif, L., M. R. P. Barretto, and L. A. Moscato (1998). “A solution to the facility layout problem using simulated annealing”. In: *Computers in Industry* 36.1, pp. 125–132.

Di Pieri, A. (2013). *Algorithms for two-dimensional guillotine packing problems.*

Dino, I. G. (2016). “An evolutionary approach for 3D architectural space layout design exploration”. In: *Automation in Construction* 69, pp. 131–150.

Drira, A., H. Pierreval, and S. Hajri-Gabouj (2007). “Facility layout problems: A survey”. In: *Annual Reviews in Control* 31.2, pp. 255–267.

Dutta, K. and S. Sarthak (2011). “Architectural space planning using evolutionary computing approaches: a review”. In: *Artificial Intelligence Review* 36.4.

Dyckhoff, H. (1990). “A typology of cutting and packing problems”. In: *European Journal of Operational Research* 44.2, pp. 145–159.

Erozan, İ. and E. Çalışkan (2020). “A multi-objective genetic algorithm for a special type of 2D orthogonal packing problems”. In: *Applied Mathematical Modelling* 77, pp. 66–81.

Feng, J. and A. Che (2018). “Novel integer linear programming models for the facility layout problem with fixed-size rectangular departments”. In: *Computers & Operations Research* 95, pp. 163–171.

Flack, R. W. J. and B. J. Ross (2011). “Evolution of Architectural Floor Plans”. In: *Applications of Evolutionary Computation*. Ed. by C. Di Chio, A. Brabazon, G. A. Di Caro, R. Drechsler, M. Farooq, J. Grahl, G. Greenfield, C. Prins, J. Romero, G. Squillero, E. Tarantino, A. G. B. Tettamanzi, N. Urquhart, and A. Ş. Uyar, pp. 313–322.

Fügenschuh, A., K. Junosza-Szaniawski, and Z. Lonc (2014). “Exact and approximation algorithms for a soft rectangle packing problem”. In: *Optimization* 63.11, pp. 1637–1663.

Gürsel, D. and Ü. Göktürk (2017). “Multiobjective Design Optimization of Building Space Layout, Energy, and Daylighting Performance”. In: *Journal of Computing in Civil Engineering* 31.5, p. 04017025.

Hammad, A. W., D. Rey, and A. Akbarnezhad (2017). “A Cutting Plane Algorithm for the Site Layout Planning Problem with Travel Barriers”. In: *Computers & Operations Research* 82.

Haupt, R. L. and S. E. Haupt (1998). *Practical Genetic Algorithms*.

He, K., P. Ji, and C.-M. Li (2015). “Dynamic reduction heuristics for the rectangle packing area minimization problem”. In: *European Journal of Operational Research* 241, pp. 674–685.

Hermanrud, I. E., C. F. Lystad, and P. J. Narvhus (2019). *Floor Plan Optimization in Schools using Genetic Algorithms*.

Hosseini nasab, H., S. Fereidouni, S. Ghomi, and M. Fakhrzad (2018). “Classification of facility layout problems: a review study”. In: *The International Journal of Advanced Manufacturing Technology* 94.

Hu, N.-Z., H.-L. Li, and J.-F. Tsai (2012). “Solving Packing Problems by a Distributed Global Optimization Algorithm”. In: *Mathematical Problems in Engineering* 2012.

Ibaraki, T. and K. Nakamura (2006). “Packing Problems with Soft Rectangles”. In: *Hybrid Metaheuristics*. Ed. by F. Almeida, M. J. Blesa Aguilera, C. Blum, J. M. Moreno Vega, M. Pérez Pérez, A. Roli, and M. Sampels, pp. 13–27.

Jain, S. and H. C. Gea (1998). “Two-dimensional packing problems using genetic algorithms”. In: *Engineering with Computers* 14.3, pp. 206–213.

Ji, P., K. He, Y. Jin, H. Lan, and C.-M. Li (2017). “An iterative merging algorithm for soft rectangle packing and its extension for application of fixed-outline floorplanning of soft modules”. In: *Computers & Operations Research* 86.

Kim, J.-G. and Y.-D. Kim (1999). “A branch and bound algorithm for locating input and output points of departments on the block layout”. In: *Journal of the Operational Research Society* 50.5, pp. 517–525.

Komarudin and K. Y. Wong (2010). “Applying Ant System for solving Unequal Area Facility Layout Problems”. In: *European Journal of Operational Research* 202.3, pp. 730–746.

Lee, Y. H. and M. H. Lee (2002). “A shape-based block layout approach to facility layout problems using hybrid genetic algorithm”. In: *Computers & Industrial Engineering* 42.2, pp. 237–248.

Maag, V., M. Berger, A. Winterfeld, and K.-H. Küfer (2010). “A novel non-linear approach to minimal area rectangular packing”. In: *Annals of Operations Research* 179.1, pp. 243–260.

McKendall, A. R., J. Shang, and S. Kuppusamy (2006). “Simulated annealing heuristics for the dynamic facility layout problem”. In: *Computers & Operations Research* 33.8, pp. 2431–2444.

Michalek, J., R. Choudhary, and P. Papalambros (2002). “Architectural layout design optimization”. In: *Engineering Optimization* 34.5, pp. 461–484.

Murata, H. and E. S. Kuh (1998). “Sequence-pair based placement method for hard/soft/pre-placed modules”. In: *Proceedings of the 1998 international symposium on Physical design*, pp. 167–172.

Scalia, G., R. Micale, A. Giallanza, and G. Marannano (2019). “Firefly algorithm based upon slicing structure encoding for unequal facility layout problem”. In: *International Journal of Industrial Engineering Computations* 10.3, pp. 349–360.

Schanke, T. (2008). *Kunnskapsstatus om skolebygg.*

Shekhawat, K. and J. P. Duarte (2017). “Rectilinear Floor Plans”. In: *Computer-Aided Architectural Design. Future Trajectories.* Ed. by G. Çağdaş, M. Özkar, L. F. Gül, and E. Gürer, pp. 395–411.

Tari, F. G. and H. Neghabi (2018). “Constructing an optimal facility layout to maximize adjacency as a function of common boundary length”. In: *Engineering Optimization* 50.3, pp. 499–515.

Udir (2019). *Statistikk om grunnskolen 2018/19.*

Verma, M. and M. K. Thakur (2010). “Architectural space planning using Genetic Algorithms”. In: *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE).* Vol. 2, pp. 268–275.

Wang, X.-Y., Y. Yang, and K. Zhang (2018). “Customization and generation of floor plans based on graph transformations”. In: *Automation in Construction* 94, pp. 405–416.

Wäscher, G., H. Haußner, and H. Schumann (2007). “An improved typology of cutting and packing problems”. In: *European Journal of Operational Research* 183.3, pp. 1109–1130.

Wong, S. S. Y. and K. C. C. Chan (2009). “EvoArch: An evolutionary algorithm for architectural layout design”. In: *Computer-Aided Design* 41.9, pp. 649–667.

Wu, W., X.-M. Fu, R. Tang, Y. Wang, Y.-H. Qi, and L. Liu (2019). “Data-driven interior plan generation for residential buildings”. In: *ACM Transactions on Graphics* 38.6, 234:1–234:12.

Xie, W. and N. V. Sahinidis (2008). “A branch-and-bound algorithm for the continuous facility layout problem”. In: *Computers & Chemical Engineering* 32.4, pp. 1016–1028.

Young, F., C. Chu, W.-S. Luk, and Y. Wong (2001). “Handling Soft Modules in General Non-slicing Floorplan using Lagrangian Relaxation”. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 20, pp. 687–692.

Zawidzki, M. (2016). “Architectural Functional Layout Optimization in a Coarse Grid”. In: *Discrete Optimization in Architecture: Architectural & Urban Layout.* Ed. by M. Zawidzki, pp. 3–34.

Zhao, Y. and N. Sannomiya (2001). “An Improvement of Genetic Algorithms by Search Space Reductions in Solving Large-scale Flowshop Problems”. In: *IEEJ Transactions on Electronics, Information and Systems* 121.6, pp. 1010–1015.

Appendices

Appendix A

Room specification documents

Table A.1: Overview of neighbourhoods contained within each test RSD

RSD	**Neighbourhoods**
$2N$	8th grade, 9th grade, Hub
$3N$	8th grade, 9th grade, ICT, Hub
$7N$	Arts & Crafts, Cooking, Music area, Sciences, Library, Employees wardrobes, Administration, Hub
$6N$	9th grade, Arts & Crafts, Cooking, Music area, Gym, Employees 9th grade, Hub
$9N$	9th grade, 10th grade, Music area, Sciences, Gym Library, Employees 9th grade, Employees 10th grade, Administration, Hub
$11N$	9th grade, 10th grade, ICT, Arts & Crafts, Cooking, Music area, Sciences, Library, Employees 9th grade, Employees 10th grade, Administration, Hub

Table A.2: The neighbourhoods considered from the RSD of Levanger Middle School

Neighbourhood	**Room type**	**Suggested size (m^2)**	**Aspect ratio bound**
9th grade	Classroom	70	1.5
	Classroom	70	1.5
	Classroom	70	1.5
	Classroom	70	1.5
	Study room	30	2.0
	Meeting room	30	2.0
	Common room (Main room)	200	4.0
8th grade	Classroom	70	1.5
	Classroom	70	1.5
	Classroom	70	1.5
	Classroom	70	1.5
	Study room	30	2.0
	Study room	30	2.0
	Meeting room	30	2.0
	Common room (Main room)	200	4.0
10th grade	Classroom	70	1.5
	Classroom	70	1.5
	Classroom	70	1.5
	Classroom	70	1.5
	Study room	30	2.0
	Meeting room	30	2.0
	Common room (Main room)	200	4.0
ICT	Offices	20	2.0
	ICT guidance	20	2.0
	Computer room	25	3.0
	Resting room	20	2.0
	Meeting room	20	2.0
	Common room (Main room)	60	4.0
Arts & Crafts	Preparation room (Main room)	100	3.0
	Woodwork	100	3.0
	Painting	70	3.0
Cooking	Kitchen	80	3.0
	Storage and cold storage room	20	2.0
	Eating area and entrance area (Main room)	80	3.0
Music area	Dancing/Movement room	60	2.0
	Practice area band	55	2.0
	Storage for music equipment	36	3.0
	Main room	80	3.0
Sciences	Chemistry	80	2.0
	Physics	40	2.0
	Preparation room (Main room)	40	2.0
Library	Library (Main room)	80	3.0
	Computer area	64	3.0
Employees 9th grade	Office 1	20	1.5
	Office 2	20	1.5
	Office 3	20	1.5
	Meeting room	20	1.5
	Common room (Main room)	60	3.0
Employees 10th grade	Office 1	20	1.5
	Office 2	20	1.5
	Office 3	20	1.5
	Meeting room	20	1.5
	Common room (Main room)	60	3.0
Administration	Teachers room	80	2.0
	Warderobe teachers	20	2.0
	Common room (Main room)	100	3.0
Employees wardrobes	Wardrobes (Main room)	120	3.0
	Toilets	30	3.0
Gym	Warderobe, girls	40	3.0
	Warderobe, boys	40	3.0
	Warderobe teacherss	20	1.5
	Gym area (Main room)	100	3.0
Hub	Consists of a number of rooms; canteen, aula, auditorium etc.	500	2.0

Appendix B

Parameter settings, memetic algorithm

Table B.1: Initial and final parameter settings, MA

Initial parameter settings		*Final parameter settings*	
GA rates		**GA rates**	
Initialization rate	0.1	Initialization rate	0.8
Crossover rate	0.4	Crossover rate	0.2
Mutation rate	0.4	Mutation rate	0.6
Elitism rate	0.2	Elitism rate	0.4
Tournament size	0.4	Tournament size	0.4
Mutation probabilities		**Mutation probabilities**	
Move room random	0.2	Move room random	0.1
Swap wall-sharing side	0.4	Swap wall-sharing side	0.4
Change room dimension	0.3	Change room dimension	0.1
Move overlapping room	0.3	Move overlapping room	0.4
Move to not-attached door-neighbour	0.6	Move to not-attached door-neighbour	0.6
Move neighbourhood random	0.4	Move neighbourhood random	0.4
Move attached door-neighbours	0.3	Move attached door-neighbours	0.5
Swap rooms	0.4	Swap rooms	0.4
Local search		**Local search**	
Local search probability	0.4	Local search probability	1.0

Appendix C

Parameter settings tests, memetic algorithm

C.1 Initialization

As discussed in Section 5.4, the model uses two different approaches for initializing an individual, heuristic and random. A probability parameter p_h determines the probability of the heuristic approach being chosen to initialize an individual. Consequently, the probability of the random approach being chosen is $(1 - p_h)$.

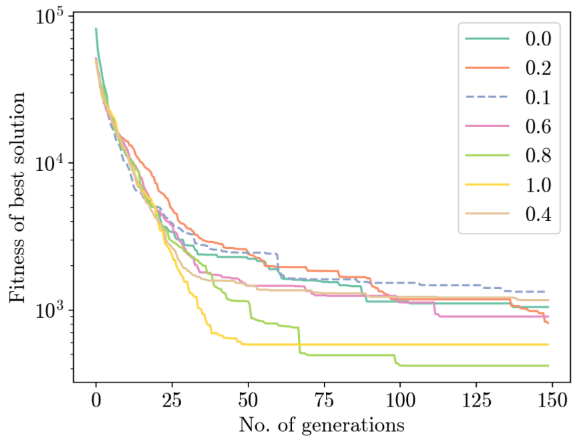


Figure C.1: Fitness of the population for different values of p_h

Figure C.1 shows a trend where a higher p_h yields better solutions. Also, the runs with high p_h converge faster. In the case with only random initialization, the initial objective value is significantly worse than higher values for p_h, and the model needs more iterations to converge. The findings argue that a high probability of the heuristic approach is good, and thus $p_h = 0.8$ for the subsequent tests.

C.2 Tournament size

The results for the tournament size do not reveal any trends on the effect of changing p_t in either direction. A possible explanation is that the tournament size is not very significant. The results are displayed in Figure C.2. As this test does not provide any insight on whether to increase or decrease the tournament size, it is kept fixed at $p_t = 0.4$. Intuitively, this tournament size balances exploration and exploitation.

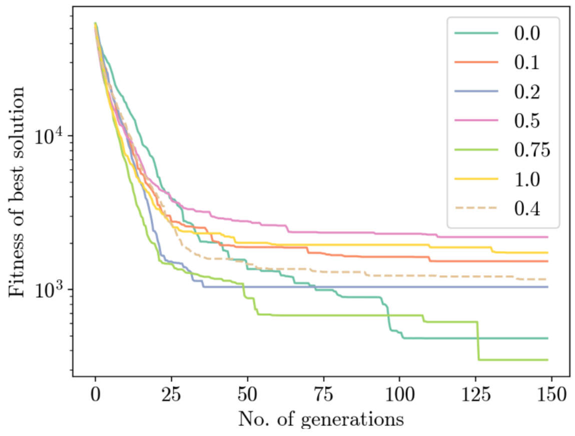


Figure C.2: Fitness of the population for different tournament sizes

C.3 Mutations

In Figure C.3, each of the eight different mutations is turned completely off, by setting their probability q_m to zero, one by one. Table C.1 shows the ID of each mutation. The IDs are used in the legend of Figure C.3, showing which mutation was turned off during that run. E.g., the orange line marked 1 in Figure C.3 means that the move room random-operator was turned off during that run.

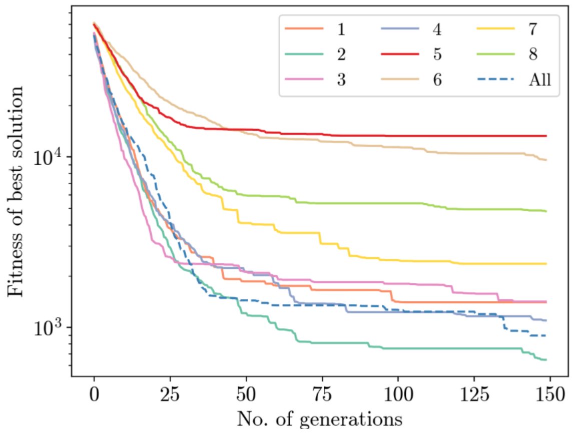


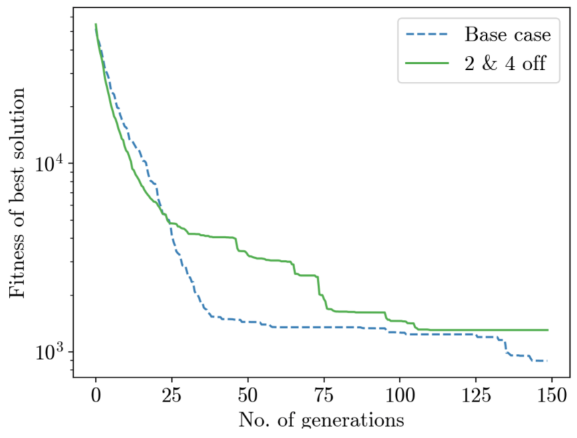
Figure C.3: Fitness of the population with different mutation operators turned off. The number in the legend corresponds to the mutation ID in Table C.1

Table C.1: Mutation IDs

Mutation IDs	
Move room random	1
Swap wall-sharing side	2
Change room dimension	3
Move overlapping room	4
Move to unattached door-neighbour	5
Move neighbourhood random	6
Move attached door-neighbours	7
Swap rooms	8
All mutations included	All

For the rest of this section, each of the mutation-operators is referred to by its ID. Figure C.3 indicates that the algorithm performs slightly better with mutation 2 and 4 turned off. Figure C.4 shows the models performance by turning them both off. The model performs slightly worse than the base case but uses many more generations to obtain the same fitness score. The base case is the run with all mutation operators in function.

Figure C.4: Swap wall-sharing side (2) and move overlapping room (4) operators turned off compared to base case



To test if decreasing the usage of mutation 2 and 4 yields better results, their probability of occurring is adjusted slightly instead of turning the mutations completely off. q_2 is adjusted from 0.4 to 0.3, while q_4 is lowered from 0.3 to 0.2. In addition, since the results in Figure C.3 show that the algorithm performs significantly worse when mutation 5 and 6 are turned off, the probability rates q_5 and q_6 are both increased from 0.4 to 0.5. The model is tested with the new parameter settings. The result is displayed in Figure C.5. The figure shows that the model performs significantly worse than the base case.

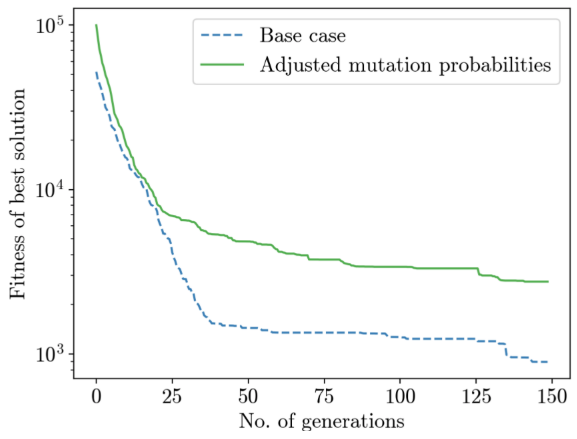
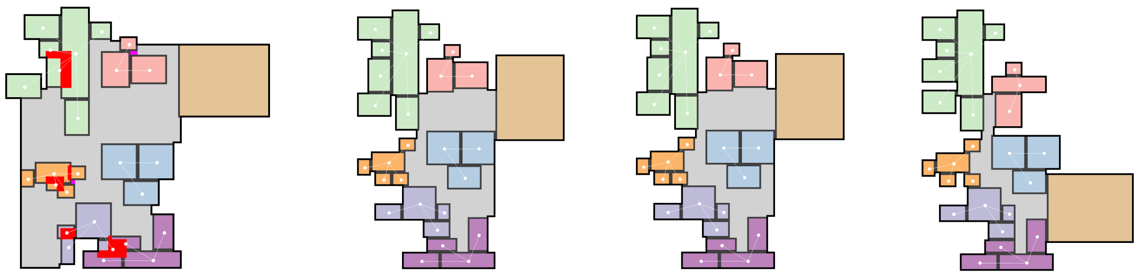


Figure C.5: Fitness of the population after adjusting the mutation probabilities, q_2, q_4, q_5, q_6

Appendix D

Objective function tests, memetic algorithm

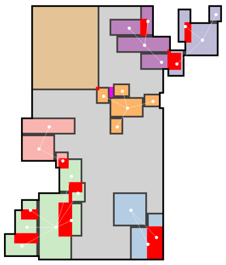
A run including all objectives is illustrated in Figure D.1.

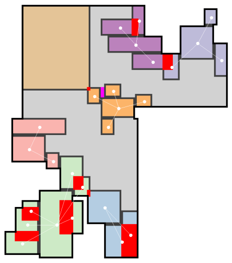


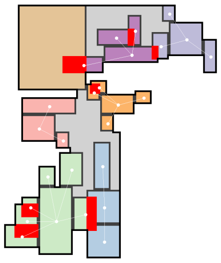
(a) Generation 5 (b) Generation 20 (c) Generation 50 (d) Generation 150

Figure D.1: Development of a run with all objectives turned on

In Figure D.2, the model is tested with an overlap weight of $w_1 = 0$. Keeping in mind that red means overlap, the figure shows a fair amount of overlap after a 150 generations. The effect becomes even clearer when the visual development of the solution is compared to Figure D.1. Since the objective function penalizes overlap in the run shown in Figure D.1, the algorithm gets rid of the overlap as it progresses. Evidently, the overlap objective is necessary and works as expected.


(a) Generation 5


(b) Generation 20


(c) Generation 50

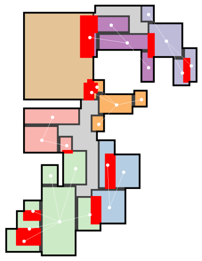
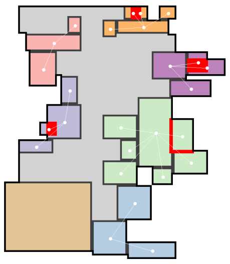
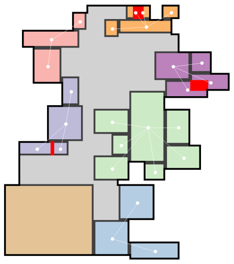
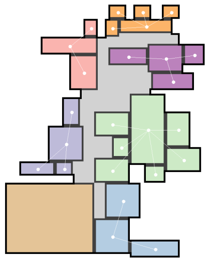

(d) Generation 150

Figure D.2: Development of best solution with the overlap objective, f_1, turned off

Next, the model is tested without the window access objective. The result is illustrated in Figure D.3. One of the classrooms belonging to *9th grade* 🟩 does not have window access. This classroom is circled by a dashed black line. In Figure D.1, window access is satisfied for all classrooms while the other objectives also are attained to the same degree as in Figure D.3. Thus, the window objective is kept in the model.


(a) Generation 5


(b) Generation 20


(c) Generation 50

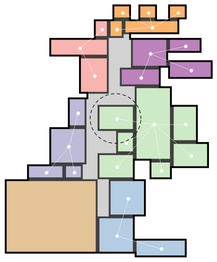
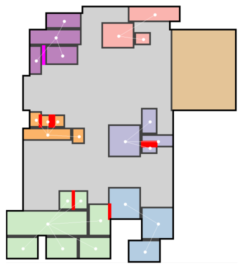
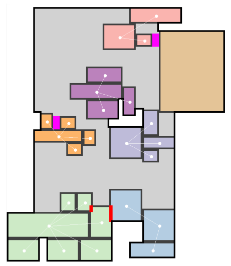
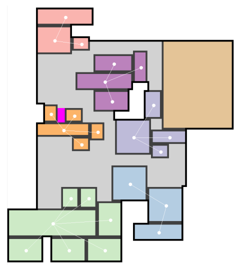

(d) Generation 150

Figure D.3: Development of best solution with the window access objective, f_5, turned off. The circled room requires window access, but does not have it

The results for the remaining objectives are illustrated below in Figure D.5 - D.4.


(a) Generation 5


(b) Generation 20


(c) Generation 50

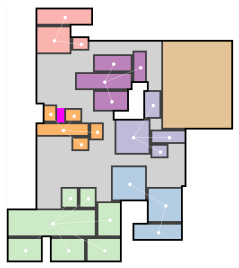
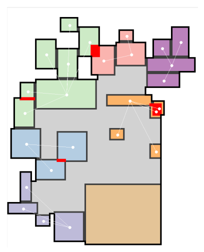
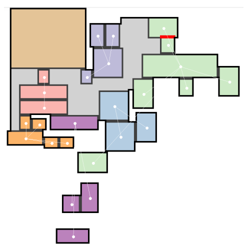
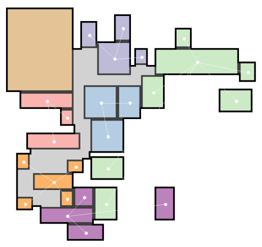

(d) Generation 150

Figure D.4: Development of best solution with the narrow hallway objective, f_3, turned off


(a) Generation 5


(b) Generation 20


(c) Generation 50

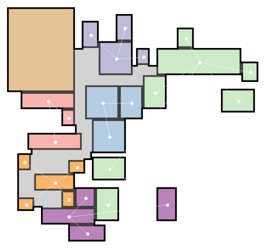
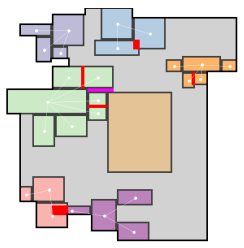
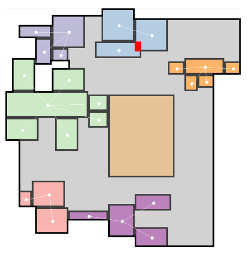
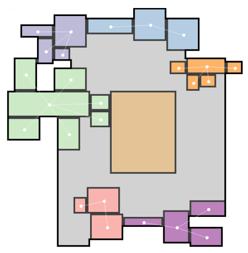

(d) Generation 150

Figure D.5: Development of best solution with the door-neighbour distance objective, f_4, turned off


(a) Generation 5


(b) Generation 20


(c) Generation 50

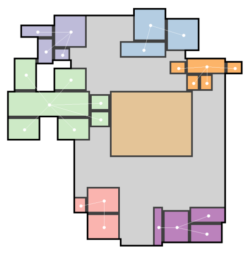
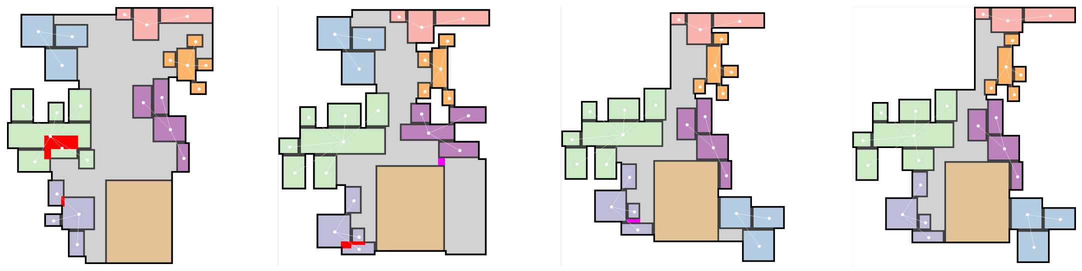

(d) Generation 150

Figure D.6: Development of best solution with the hallway area objective, f_6, turned off



(a) Generation 5 (b) Generation 20 (c) Generation 50 (d) Generation 150

Figure D.7: Development of best solution with the excess neighbourhood area objective, f_7, turned off

Appendix E

Technical study, mathematical model

E.1 Lock main room

Table E.1: The objective values of the resulting solutions using the objectives in stage one and three using the lock main room heuristic

	$6N_1$		$6N_2$		$6N_3$	
	In	Out	In	Out	In	Out
O	0	0	0	0	0	0
HA(%)	0	0	0	0	0	0
C	0	1	0	3	0	1
NH	0	0	0	0	0	0
WA	0	0	0	1	0	0
ENA	710	293	731	279	710	293
EC	30	32	20	30	30	32

(a) $6N$ instances

	$9N_1$		$9N_2$		$9N_3$	
	In	Out	In	Out	In	Out
O	0	0	0	0	0	0
HA(%)	0	561	0	247	0	31
C	0	3	0	2	0	4
NH	0	10	0	0	1	24
WA	0	6	0	3	0	0
ENA	981	606	800	429	1115	603
EC	48	30	50	28	34	42

(b) $9N$ instances

E.2 Lock hallways

Table E.2: Initial and final number of corners and run time for the 6N instances with the lock hallways heuristic

	$6N_1$				$6N_2$				$6N_3$			
Neighbourhood	C_I	C_F^H	T_B	T_F	C_I	C_F^H	T_B	T_F	C_I	C_F^H	T_B	T_F
Music Area	14	4	17	17	12	4	205	205	10	4	1	1
9th grade offices	16	6	46	480	12	6	38	480	12	8	11	333
Gym	12	4	14	14	10	4	75	75	12	8	7	7
Arts & Crafts	8	4	12	12	4	4	1	1	8	4	5	5
9th grade	14	14	-	480	20	20	-	480	18	18	-	480
Cooking	8	6	1	3	8	6	0	1	10	6	0	2

Table E.3: Initial and final number of corners and run time for the 9N instances with the lock hallways heuristic

	$9N_1$				$9N_2$				$9N_3$			
Neighbourhood	C_I	C_F^H	T_B	T_F	C_I	C_F^H	T_B	T_F	C_I	C_F^H	T_B	T_F
10th grade offices	16	4	292	292	12	6	2	30	12	6	151	480
Music Area	12	4	305	305	10	4	43	43	10	6	6	241
9th grade offices	14	6	55	480	12	6	3	29	10	4	4	4
Gym	14	6	30	253	14	4	30	30	10	4	15	15
Library	4	4	0	0	6	4	0	0	6	4	0	0
9th grade	20	20	-	478	16	16	-	480	12	12	-	480
Science	8	4	0	0	12	4	1	1	6	6	0	0
Administration	6	6	0	0	6	6	0	3	8	6	1	4
10th grade	12	12	-	480	14	14	-	480	16	16	-	480

Appendix F

Technical study, local search

The results are as expected where the run time decreases with shorter lengths.

Table F.1: Run time for the local search, for each combination of instance and length. Time is given in seconds

	Search area length l					
	2	**5**	**10**	**12**	**15**	**18**
$6N_a$	2.29	47.86	118.71	118.43	443.29	278.29
$6N_b$	6.57	27.57	126.57	208.0	275.57	307.71
$9N_a$	33.25	70.29	303.14	291.86	570.57	486.0
$9N_b$	13.0	61.71	223.29	252.86	324.71	561.86

The fraction of runs producing feasible solutions for each combination of selection approach and instance is presented in Table F.2. The results show that all selection approaches produce solely feasible solutions for all test layouts.

Table F.2: The fraction of feasible solutions generated for each combination of selection approach and instance

	Selection approach		
	1	**2**	**3**
$6N_a$	1.0	1.0	1.0
$6N_b$	1.0	1.0	1.0
$9N_a$	1.0	1.0	1.0
$9N_b$	1.0	1.0	1.0

Table F.3: Run time for the local search, for each combination of layout and selection approach. Time is given in seconds

	Selection approach		
	1	**2**	**3**
$6N_a$	236.57	256.57	443.29
$6N_b$	578.29	378.86	275.57
$9N_a$	756.43	654.57	570.57
$9N_b$	251.0	358.86	324.71