A K-MEDOIDS-BASED SHAPE CLUSTERING METHOD AND ITS APPLICATIONS IN GENERATIVE DESIGN AND OPTIMIZATION SYSTEMS

A Dissertation

by

SHERMEEN AHMED YOUSIF YOUSIF

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

DOCTOR OF PHILOSOPHY

Chair of Committee,	Wei Yan
Co-Chair of Committee,	Charles Culp
Committee Members,	Stephen Caffey
	Philip Galanter
Head of Department,	Robert Warden

August 2019

Major Subject: Architecture

Copyright 2019 Shermeen Ahmed Yousif Yousif

ABSTRACT

As the number of design candidates in generative and design optimization systems is often excessive and overwhelming, with similar and redundant shapes of design candidates evolving, there is a need for an articulation mechanism that assists designers in the exploration and examination of the design set in a feasible manner. This work is focused on introducing a new Generative Design System (GDS) that facilitates the designers' interaction with such systems and accommodates decision making in the process. The proposed system incorporates an innovative Shape Clustering using K-Medoids (SC-KM) method into other routine processes of parametric modeling and design optimization.

The research methods include an extensive literature study, experimenting, prototyping, and validation procedures. A prototype was carried out, as the apparatus to demonstrate and test the proposed GDS, and the clustering method. In developing, demonstrating, and testing the prototype, three test-cases were pursued. Within the prototype, at first, a process of parametric form generation was carried out to initiate a design model parametrically to allow for a possibly heterogeneous or/and similar set of design options to be produced, in an algorithmic manner. Second, a design optimization process was pursued where the initial parametric model was subjected to building performance evaluation inside a Multi-Objective Evolutionary Algorithm, such as in Test-case 3. In the third process, the new SC-KM method was formulated and applied, using two functionalities: (1) a grid-based shape descriptor was used for a pair-wise shape comparison with the implementation of the Hungarian's algorithm, carried out to find the shape difference score matrix for the analyzed shapes, (2) K-Medoids clustering was employed to group the design shapes into different subsets, each of similar shapes, and identify each group's

Medoid–the representative shape in the group. Applying the algorithmic definition to the samples of three test-cases, the results of the SC-KM showed satisfactory clusterings. Furthermore, verification procedures were conducted for each test-case, and in particular, external validation with the calculation of clustering evaluation metrics was pursued in Test-case 2.

In contrast to the accepted practice of current generative design systems that lack organizational methods, the significance of this work is to expand and advance such systems incorporating cluster analysis as a big-data management strategy and a potential solution. The research provides contributions through the following: (1) introduction and illustration of a fully working prototype of a new generative design system, (2) development, testing, and validation of a new package of algorithms for the developed SC-KM, method. The package of plugins and algorithms will be made available for designers to download as an open-source in a visual programming interface, to be applied to a wide range of related design problems.

DEDICATION

To my role model and the most loving, giving, and caring person who will be proud the most of this doctoral degree, my father,

to my mother in heaven,

to my supportive family whose unconditional love was my strength,

to my loving husband,

and to my precious boys,

I dedicate this dissertation.

ACKNOWLEDGEMENTS

First, and foremost, I would like to thank my committee chair, Dr. Wei Yan, one of the kindest, most knowledgeable, and most supportive and encouraging professors I have ever seen, to whom I owe my success in academia. I also want to thank my co-chair Dr. Charles Culp, who taught me how to succeed in the doctoral study and has been guiding me for years, my committee member, Dr. Stephen Caffey who was always there for me and has been supportive throughout the course of this research, and to Dr. Prof. Galanter, for his guidance and supervision.

I would like to express my gratitude to other faculty members in the department, particularly, Dr. Mark Clayton, for his contributions to my knowledge of computational design research. In addition, I want to thank Dr. Geoffrey Booth and Dr. Valerian Miranda for their encouragement and support to the BIM-SIM computational research group. I also thank Dr. Gabriela Campagnol for her assistance.

My very special thanks to my family members, my father, my beloved husband for his patience and love, and particularly my sisters for their encouragement.

I want also to extend my gratitude to thank my Ph.D. friends Dr. Nessrine Mansour, soon to be Dr. Bara Safarova, Dr. Emad Al-Qattan, soon to be Dr. Nancy AlAssaf, Dr. Jawad Al-Tabtabai, Dr. Chengde Wu, Dr. Hyoungsub Kim, soon to be Dr. Maki Isaka and Amreen Shahjahan, who have been really good friends and supportive throughout this Ph.D. In addition, I would like to thank my other Ph.D. friends, Mehdi Farah Bakhsh, Fatemeh Shahsavari, Zohreh Shaghaghian, Mohammad Alawadhi and all the members of the BIM SIM research group at

Texas A&M University. Also, I want to thank the Aggie Coding Club, particularly Ruben Vazquez-Chapa for his help.

CONTRIBUTORS AND FUNDING SOURCES

This work was supervised by a dissertation committee consisting of Dr. Wei Yan (chair), Dr. Culp (co-chair), and Dr. Stephen Caffey (member) of the Department of architecture and Professor Phillip Galanter (member) of the Department of Visualization.

All work for the dissertation was completed independently by the student. Some of the dissertation content is being published in (Yousif & Yan, 2019).

Graduate study was supported by multiple scholarships from the Department of Architecture, Texas A&M University.

NOMENCLATURE

2D	Two-Dimensional
3D	Three-Dimensional
CAD	Computer-Aided Design
EA	Evolutionary Algorithm
EPSAP	Evolutionary Program for Space Allocation Program
EP	Energy Plus
F1	F-Score
FN	False Negative
FP	False Positive
GB	Grid-Based
GDS	Generative Design System
GH	Grasshopper
HVAC	Heating, Ventilating, and Air Conditioning
IDF	Input Data File
LEED	Leadership in Energy and Environmental Design
MOEA	Multi-Objective Evolutionary Algorithm
ML	Machine Learning
MOO	Multi-Objective Optimization
MPEG	Moving Picture Experts Group
PAM	Partitioning Around Medoids

RI	Rand Index
SC-KM	Shape Clustering using K-Medoids
sDA	Spatial Daylight Anatomy
SPEA-2	Strength Pareto Evolutionary Algorithm
TN	True Negative
TP	True Positive
WWR	Window-Wall-Ratio

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

1. INTRODUCTION

The development and implementation of generative design systems (GDSs) have been extensively studied and researched for architectural design. Such generative models have been applied to whole-building design, structural design, façade, space allocation, optimization of building form and performance, computational reproduction of architectural styles, and urban design (Rodrigues et al., 2017). Despite their capabilities in generating and evaluating a population of design alternatives, GDSs lack organization (articulation) methods of the produced design set, particularly in terms of geometric shapes/forms. This drawback leads to an excessive number of redundant and unnecessary design alternatives that evolve in the process and require extended computation time. In particular, such a large number of evolved designs can be overwhelming for designers and can inhibit effective interaction with the system. There is a significant need to improve current GDSs to facilitate a better examination by designers of the shape qualities of the generated architectural designs and to allow for effective decision making by designers to select and evaluate further their designs of interest.

The basic idea of this research is that methods of clustering, such as those found in statistical analysis that support organizing big data, can be applied to GDSs as articulation methods for the produced forms/shapes, leading to a new generative model. Thus, an organization method, a Shape Clustering using K-Medoids (SC-KM) method, was proposed, developed, and incorporated into GDSs in this work. Departing from a routine parametric generative process to create architectural shapes, and a routine design optimization process, an innovative clustering process was developed and integrated to articulate the set of shapes leading to a clustered and diverse set of design shapes that represent a much larger set of similar and

wanted/unwanted ones. The most important contribution of this study is the method it proposes, rather than the specificities of the particular applied test cases developed to validate that method. The general prototype of the new generative model, functionalities of the new clustering method, application test cases, and validation procedures are explained in this dissertation.

1.1. Background and Terminology

Progress in computational design can be traced from the simple use of computational tools for modeling and representation in the 1980s and 1990s to the more recent complex use for analytic and eventually for automotive, generative and explorative purposes (Celani & Eduardo Verzola Vaz, 2012; Mark et al., 2001). Primarily, the interest in generative methods has significantly increased after the introduction of digital fabrication allowing for free-form exploration as well as mass-customization in a new way (Celani & Eduardo Verzola Vaz, 2012). The rapid prototyping of computational design methods with the integration of programming (scripting) capabilities, interwoven with the recent advancement in computer-based design tools that offer new possibilities of complex geometry modeling, had become essential for architects and designers in today's design process (Shea et al., 2005). Generally, a shift has occurred from using the computer to assist in design to employ the computer as a generative collaborator in the design process. GDSs, as described by Mitchell (1975), are devices capable of generating potential solution candidates for a given design problem. Often in those systems, computation-based parametric variation and transformation rules are used as the main approaches (Celani & Eduardo Verzola Vaz, 2012).

Those generative systems involve two main modules: parametric modeling and programming (Shea et al., 2005). In parametric design, designers can generate a large set of design alternatives based on explicit rules and parameters, allowing for evaluation of those

design alternatives in terms of design quality (Aish & Woodbury, 2005). Representation methods of computational tools include graph-based and text-based (visual programming and scripting). Generative systems are now commonly implemented in the architectural design process through the use of computational methods such as parametric modeling (Woodbury, 2010), and numerical simulations and performance-driven frameworks (Malkawi, 2005) for enhancing the design process (Wortmann & Nannicini, 2017).

Form finding is associated with GDSs; it refers to the search process that often start with parametrization of the initial design, and progress to finding the successful or best candidates from the generated set of designs, often after a process of evaluation of form quality or/and performance to find the best designs (Barnes, 1999). In regards to building performance evaluation, the process often involves the search for fitter solutions in terms of certain performance criteria. In such a performance-driven framework, often building simulation of particular performance criteria is coupled up with optimization to find optimum solutions that satisfy those objectives. Optimization refers to the procedure of making something, such as a design candidate, as effective as possible (Nguyen et al., 2014). Simulation-based optimization is commonly used within generative design systems.

Although they are promising models, a significant drawback in generative protocols is that they can produce an excessive number of design alternatives for humans to perceive and evaluate (Rodrigues et al., 2017), and, thus, can be overwhelming for designers in terms of decision-making and interaction with the systems. Also, in the produced set of designs, multiple shapes can be redundant and very similar in various measures, leading to extended computation time, with no need to be taken into account in the explorative process. This redundancy distracts designers from in-depth examining or from focusing on the diverse design candidates. For

managing this problem, the solution set should be organized and condensed to the highly diverse set, and similar designs need to be eliminated (Brown & Mueller, 2019; Yousif et al., 2018).

Ideally, it would be useful to articulate the design solutions according to their shape/form (similarity/dissimilarity) and present the subsets of similar characteristics and highlight only those diverse solutions or designs of interest to the designers for evaluation. This allows for comparing the organized group types of designs and examining particular candidates (Rodrigues et al., 2017). Clustering, one of the methods for managing a collection of data to lead to meaningful clusters of similar characteristics, can offer a solution. Meaningful in this context refers to an organized set that can be feasibly analyzed by designers. Generally, clustering is associated with another data organization approach, classification. Classification deals with predefined and well-known labels and features, while clustering is unsupervised and handles data with no need for preexisting labels. Clustering is a numerical method for partitioning the dataset (Everitt, 2011). As a promising approach to achieving this articulation goal, the statistical method "cluster analysis" or clustering can be used in generative systems. Cluster analysis is a collective term that includes a range of methods and techniques that delineate groups in a collection of data (Anderberg, 1973). It refers to partitioning data points or objects into multiple clusters so that those objects inside a cluster share similar characteristics but are very different to the objects in other clusters (Han et al., 2011). Clustering's main purpose is exploratory in data mining and has been used in Machine Learning, pattern recognition, image analysis, information retrieval, and other areas (Han et al., 2011).

Importantly, there is no established method to integrate articulation techniques for architectural forms/shapes in generative design (Brown & Mueller, 2019). Despite its potentials, the incorporation of clustering methods into architectural GDSs remains limited. One related

study of using clustering techniques for architectural floor plans is the work of Rodrigues et al. (2017). The study of Brown & Mueller (2019) offers an insight to the issues of overwhelming number and redundancy of produced solutions in generative protocols, and reviews recent attempts of diversity techniques and metrics to solve the problem. Both studies assert the importance of further investigations in finding and implementing articulation methods in generative systems.

Despite that, there are few studies that attempt to address and solve the excessiveness of unwanted design solutions in architectural design, more tools, and plugins for Machine Learning and big data management are being produced. Developed recently, the three tools (Ivy, Owl, and Ant) represent experimental plugins for combining Machine Learning techniques into the visual programming platform (Grasshopper) and include K-Means clustering methods. However, in those tools, the use of clustering was designed for mesh segmentation and not for architectural shapes. Away from clustering, other commercial tools are available for shape comparison to find discrepancies between two CAD models such as the commercial tool “Kubotek3D Compare”, which has been tested in the preparatory phase of this study. While having various functions, the tool lacks the capability to give a shape difference score, which is important in the shape clustering method formulated in those work. Given this initial background, there is still a gap in the area of incorporating shape comparison and clustering into generative systems, that this research attempts to fill in.

1.2. Research Problems

Computational tools are useful for rapidly producing design solutions with varied parameters and powerful in evaluating the performance of those designs (Cvetkovic & Parmee, 2002); however, the creative process is done by the designers, and the generative exploratory

scheme should be effective and support successful interaction between designers and the system. Reviewing existing literature and methods, current GDSs still lack established articulation mechanisms implemented for successful form finding (Brown & Mueller, 2019; Rodrigues et al., 2017; Yousif & Yan, 2019). Collectively, for designers, "*it is just not feasible to rate solutions according to a performance criterion and then select the top-ranked ones, especially for unclear and subjective problems*" (Rodrigues et al., 2017, p. 1). Importantly, in the search for expressive designs in generative schemes, the range of the design solutions must be diverse in terms of forms/shapes, since the selection done by designers from the optimal solution set is often based on aesthetic preferences (Brown & Mueller, 2019), in addition to functional performances. As such, it is required to organize the design solutions resulting from generative systems to be presented to the designers so they can examine the generated solutions in an organized and feasible way. For architectural form/shape examination, it would be important to compare the geometric features and rate the design set based on shape similarity and difference.

1.3. Research Overview

Given the general background of the research and the identified problems, the following subsections are dedicated to addressing the research objectives and methods. As a primary objective, this research is directed to introduce a new generative design system that can benefit designers and architects in the explorative search process in enhancing their selection and examination activity of the emerging design solutions in such systems in terms of geometric characteristics. The methods used in this work have been determined to achieve the purposes and functionalities of the developed clustering method and the system framework.

1.3.1. Objectives and Propositions

Investigating and experimenting with current generative design methods and studies, as well as identifying their potentials and limitations have motivated this work. In response to the identified problems, the research was targeted to improve GDSs systems in terms of achieving the following: (1) find a shape similitude/dissimilitude measure, (2) develop and implement a clustering method, and (3) incorporate the clustering method into a generative framework with testing and validation. Thus, the two main objectives pursued were:

1. To develop a new shape clustering method consisting of a series of algorithmics that is capable of clustering a dataset of shapes into groups of similar geometric characteristics and finding the representative shape for each group. The clustering results would be an articulated and condensed set of design shapes.
2. To create a system framework of a generative design model that incorporates the formulated clustering method into the whole form-finding process. The clustering method was developed and implemented in two test-cases (Test-case 1 and 2), and the fully working system prototype was illustrated and tested in Test-case 3. In each test-case, a validation study was conducted.

1.3.2. Research Strategy (Methods)

The methods used in this study include literature study, experimenting, prototyping, testing, and validation of findings. The suggested generative system was developed through five tasks following the methods described by Kunz (1989) as required tasks to create a computational model, and informed by the required characteristics of design prototypes offered by (Gero, 1990). Those tasks can be briefly explained as:

1. Identifying the purposes of the system: for this task, literature review and experimenting with existing models and methods were needed to identify the research problems and determine how the propositions of the new proposed system should be achieved. The primary problems found were the lack of a fully developed clustering method and the need for a design system that illustrates the incorporation of the proposed SC-KM into the generative design process.
2. Describing the representation model of the domain: in this task, a prototype would be developed to represent the suggested model and test its functionalities. To achieve the system's purposes, the sub-tasks/processes were determined to be the following:
   (a) Parametric form generation: a parametric modeling functionality for allowing parametric modifications leading to the generation of multiple design shapes.
   (b) Design optimization (performance evaluation and optimization): the process includes the use of a generative tool with simulation and optimization capabilities.
   (c) Developing the SC-KM method for design presentation: a clustering process in which the generated designs are clustered according to their geometric differences. The complete process of the suggested SC-KM consists of sub-tasks to (1) implementing a shape description technique, (2) measuring shape difference according to geometric analysis and comparison, and (3) performing the SC-KM. To find geometric differences, a cross-reference shape analysis amongst the generated solutions was conducted. This method consists of a set of algorithms, incorporating the Hungarian algorithm in shape comparison. The K-Medoids algorithm, a distance-based cluster analysis, was used to cluster the shapes into groups of similarities and find the representative of each group.

3. Specifying the reasoning to analyze the model: throughout its development, each functionality had to be tested and evaluated (and improved) to achieve the system's purposes.
4. Creating an interface: to communicate the system, an interface is needed for demonstrating the prototype and its components (connected functionalities and algorithms).
5. Testing the model validity: importantly, to achieve successful clustering of architectural shapes, the modules and functionalities of the SC-KM method were tested and validated.

1.4. Research Contributions

The developed clustering method has applications in GDSs to organize the resulting data in a conceivable manner to support designers' exploration and evaluation. Using the clustering process as the final phase in a generative design process, the architects and designers can be presented with the clustered design shapes and their representatives, and can then choose the ones that they prefer, thus changing the existing workflow of the generative system. Alternatively, the clustering process can be done at any time in the generative run, to filter the shapes with high dissimulate in the generative process. For instance, in early iterations, this organization of shapes will help eliminate the shapes with similar geometric characteristics and select the representative shapes of these many similar ones, leading to a diverse but smaller population for designers to examine, and reduction in computation time for following processes of performance simulation or optimization. Another use of the method is to help designers focus on their shapes of interest, the specific shape they want to further evolve.

1.5. Research Significance

This research provides a validated clustering method and a fully working generative system as a prototype that enables designers to evaluate and choose from a clustered set of designs more effectively in the process. The research introduces a system that impacts the architectural design process by:

- prototyping a new GDS in which the SC-KM is incorporated to demonstrate the system workflow and functionalities to facilitate applications by designers to design problems.
- providing a scientific contribution by developing a new SC-KM through formulating an algorithmic set for achieving successful clustering of design shapes.
- improving generative systems to allow for better human-computer interaction.

The significance of this work lies in its applicability in general design frameworks, and in particular in the architectural design process for both education and practice endeavors. In terms of practice, the main value is that the SC-KM integrated generative model can be applied directly in conceptual and schematic design phases of generative protocols. Architects and design practitioners can benefit from the method proposed here to control the design search space, working in a hybrid mode of moving between the evaluation of building performance, and evaluation of shape characteristics in an organized process. For design education, the applications of this work are situated in the parametric and generative exploration design process where architecture students can use the SC-KM method in both speculative projects and real building design.

The value of this work is to help overcome the challenge of considering the building design process as simply a building performance optimization problem. Such an assumption leads to an ill-defined problem based on mere performance and thus results in designs that can be misleading in the sense that they are optimal without addressing other important design criteria. The argument here is that complexity of architectural design process is inevitable and thus has to be considered, and one important aspect of architecture is the design shape and its characteristics, and shape studies can be facilitated as explained in this dissertation.

1.6. Outline of the Sections

This dissertation includes 5 sections, described as follows:

- Section 1 - Introduction: This section gives an introductory overview to the dissertation, offers a background on the research, describes the research problems in GDSs, proposes the research objectives to solve such problems, explains the needed clustering method and its incorporation into the proposed system, and finally gives a brief introduction to the expected contributions and research significance.
- Section 2 - Background: This section provides a review of literature on the methods related to three processes of form-finding in the proposed generative model: (1) parametric modeling and GDSs, (2) design optimization including evaluation and optimization, and (3) SC-KM-related methods and studies. The gap in the body of knowledge is identified, and an introduction to the methods and algorithms for the suggested clustering method is made.
- Section 3 - Research Methods: This section gives a detailed description of the methods used to conduct this study. It also elaborates on the development of the

system prototype, its components, functionalities, applications, and validation procedures.

- Section 4 - Test-Cases Development and Validation: This section provides the details on the development process of the three experimental test-cases carried out. For each test-case, an introduction is given, its development phases are explained, its outcomes are presented and discussed, and its validation studies are elaborated.
- Section 5 - Conclusion and Future Work: This section includes the contributions of the research, the limitations as well as future work.

2. BACKGROUND

In this section, first, background is offered, dedicated for a description of related work done by other scholars who studied similar methods to the one described here or brought new insights for further development. In addition, a review of the tools and algorithms that were most significant to the development of this project is carried out. Importantly, some earlier applications and testing of some of the methods, applied to architectural design problems, were conducted and also briefly described in the following subsections.

Research on articulating design solutions produced in computational generative schemes in architecture has been only recently addressed (Brown & Mueller, 2019; Rodrigues et al., 2017). In organizing big data, particularly in Machine Learning, two methods are typically pursued: classification and clustering (Rodrigues et al., 2017). Clustering is an important research area; it refers to the process of creating groups of objects that are similar in some way (Velmurugan & Santhanam, 2010). Clustering is labeled as an unsupervised learning method, as it deals with finding a structure or organizing a collection of unlabeled data (Jain et al., 1999; Velmurugan & Santhanam, 2010). Unlike classification, which deals with predefined classes, clustering does not tackle classified data which makes it advantageous in finding interesting hidden patterns with no predefined knowledge (Velmurugan & Santhanam, 2010).

There is limited use of clustering methods in research on generative design systems. One of the few related studies is the work of Rodrigues et al. that compares multiple descriptors of 2D shapes and utilizes the Ward linkage clustering method for architectural floor plans (Rodrigues et al., 2017; Ward Jr, 1963). The Ward linkage method, that is explained in Subsection 2.3.3, was used to cluster a synthetic dataset of 72 floor plans, applying and comparing four shape

representations: (1) point distance, (2) turning function, (3) grid-based model, and (4) tangent distance (Rodrigues et al., 2017). The study authors point out the need to further investigate clustering algorithms for architectural layouts.

Relevant to this work is the study of Brown & Mueller (2019) on exploring and reviewing different diversity metrics used in generative design protocols. For (Brown & Mueller, 2019), seeking design diversity amongst the possible design alternatives is important to avoid obtaining, or simulating, repeated and similar candidates and to enhance generative mechanisms ensuring that "the results they produce are diverse enough to be interesting to designers" (p. 2). Importantly, their work reviews available approaches to diversity and asserts that there is no established method yet to achieve diversity in exploratory systems. In a prior work of this research, a diversity measure was developed to condense the design set into a highly diverse one (Yousif et al., 2017). In continuing experimentation with design diversity, it became obvious that clustering methods have the capability to lead not only to a highly diverse set of designs but also retain and organize all designs and thus clustering techniques were further investigated.

In mechanical engineering and product design, the work of Jayanti et al. (2009) represents one of the early attempts to address the need for clustering in managing CAD repositories to sort and retrieve the 3D models according to shape similarity. The aim is to provide engineers with an organized design repository. The study compares five shape representation techniques that are often applied in engineering, targeted for clustering evaluation. It considers the 2D drawings of the 3D CAD models and applies a K-Means clustering method (Jayanti et al., 2009). The work focuses on evaluation mechanisms of the clustering results, and encourages further investigation of shape representations, clustering methods, and clustering

effectiveness measures; it asserts the need to research how different clustering algorithms are suitable for different shape representations (Jayanti et al., 2009).

Shape representation and shape comparison are significant for performing the shape clustering method investigated here. Shape representation is associated with finding effective and descriptive shape features (Zhang & Lu, 2004). An area related to shape comparison is pattern recognition; the work of (Cha & Gero, 1998) has laid foundations for a shape pattern recognition system based on structural shape representation. In another work, de las Heras, et al. (de las Heras et al., 2014) have used an approach to retrieve designs with similar properties from a dataset, while Dutta et al. applied a graph-based method to recognize symbols in floor plans such as furniture and fenestration (Dutta et al., 2014).

An attempt to introduce clustering, particularly K-Means clustering, to generative design in visual programming platforms has been done by the authors of the Ivy tool for Grasshopper (Nejur & Steinfeld, 2016). However, the K-Means clustering in the tool was applied to mesh segmentation and analysis, and not to the whole building shape or form. Other tools that use clustering include the Machine Learning-based tools: “Owl” (Zwierzycki et al., 2018), and “Ant” (Abdelrahman & Toutou, 2019), both for the Grasshopper platform. The K-Means clustering in the Owl tool has been tested in experimentation studies for this research. Those tools are general and not specifically targeted to achieve shape clustering based on shape description and shape difference analysis.

Apart from clustering, another approach to identify shape difference is CAD-based geometric comparison. As part of the preparatory experiments for this work, testing of the industrial tool “Kubotek3D Compare” was conducted. It is one of the available commercial software surveyed by Brière.-Côté et al. (2012). The software accepts CAD-based models,

performs a pair-comparison of the 3D CAD models to find discrepancies between the two models, and it highlights the discrepant elements. It is often used for mechanical products, however, when tested with two Revit models of single-family house design, the tool successfully found the different elements. One of its drawbacks is that it doesn't provide a shape difference score between the compared models, which is needed for our clustering method.

Given this preliminary background, the lack of established shape clustering methods in form finding remains an unsolved problem. Therefore, this study was targeted to introduce a new SC-KM method that improves form finding and can be integrated into generative design systems. To describe the developed clustering method, there is a need to first introduce the methods and studies most relevant to this research. The order used to describe those methods in the following subsections was also carried out throughout the development of the GDS prototype and the clustering method described in Section 3 in the sequence of (1) parametric form generation and initiating a generative design system, (2) design optimization (performance evaluation and optimization): the use of evolutionary algorithms for multi-objective optimization, (3) shape clustering: shape description and shape different calculation, the Hungarian algorithm, and the K-Medoids Algorithm.

2.1. Form Finding

Form finding is a process, in which a set of design forms are generated and subjected to evaluation for either performance optimization or examination of their aesthetic qualities, in order to determine the optimum designs or preferred design aesthetics. The process can be started from arbitrary parametrization or specification of the design form (Barnes, 1999). Prior to the use of computational design tools for form finding, architects conducted analog geometric analysis experiments to determine the final designs (Piker, 2013). In the current implementation

of computational form finding, a generative tool, or a search mechanism such as Evolutionary Algorithms is often employed, coupled up with the evaluation of certain design criteria.

In this study, the term *form finding* refers to the entire process, consisting of the subdivided processes of (1) parametric form generation, (2) design optimization, and (3) shape clustering and design presentation. Yet, the overall process did not always include evaluation and optimization (such as in Test-case 1 and 2) in which form-finding was abridged to parametric form generation and shape clustering.

2.1.1. Parametric Form Generation

Parametric modeling is associated with geometry modeling when rules are applied to govern geometrical or other relations within design morphing for surface or structure modification (Holzer, 2015). Practically, parametric modeling is applied during the conceptual design phase to test a range of design options, for intuitive design explorations (Holzer, 2015). It allows for variation in design using variables and constraints in model elements and relations and enables generative explorations of design options that automatically update and change according to performance change (Aish & Woodbury, 2005). Designers work in parametric design systems on two levels: 1) defining schemata and constraints and 2) searching for best instances within a defined schema (Aish & Woodbury, 2005). Parametric design tools provide rapid iterations, often to achieve aesthetic form evaluation and to respond to environment-related requirements of energy savings, maximized daylight performance or/and respond to other criteria. This makes parametric design methods significant to achieve performance-based design objectives by aiding in the generation of multiple, discrete solutions (Caplan, 2011).

2.1.2. Parametric Generative Design Systems

Use of parametric modeling enables a designer to transition from designing one solution to designing a system that generates multiple solutions (Stocking, 2009). These systems can be called “Generative Design Systems” or GDSs. They can be simply defined as systems that are capable of producing potential design candidates for a particular design problem (Mitchell, 1975). In such systems, the generative components are parametric modeling and programming (Aish, 2003; Shea et al., 2005). In another definition, the components of a generative system involve performance simulation and the search mechanism (Caldas, 2001). In using generative methods, the computer becomes a “design generator” or a “collaborative partner” in the design process, capable of generating design alternatives, when rigorously defining design parameters and constraints (Shea et al., 2005). In this study, the term GDS refers primarily to the system composed of a parametric definition of an initial model and a generative component. In such systems, parametric variation and transformation rules are employed (Celani & Eduardo Verzola Vaz, 2012). This generative component can be simply a manual change of the model parameters leading to produce a set of design alternatives or a generative tool with random population and search mechanism for fitter solutions such as the Multi-Objective Evolutionary Algorithm described in Section 2.2.2.

2.2. Design Evaluation and Optimization

In the case of pursuing evaluation and optimization (in Test-case 3 of Section 4), the process consists of two coupled components: building performance simulation and multi-objective optimization. Those building performance criteria may include construction costs (Radford & Gero, 1987), energy consumption considering Heating, Ventilation, and Air

Conditioning (HVAC) systems (Zhang et al., 2006), and daylight measures (Torres & Sakamoto, 2007). The following subsections describe the two components.

2.2.1. Building Performance Simulation

Building simulation has emerged through energy reduction demand and sustainable practices (Clarke, 2001). Simulation of building energy performance has become essential to meet the challenges of expected high performing buildings (Samuelson et al., 2016). The obstacle to achieving energy efficiency is related to ineffective decision making, especially in the early design stages (Clarke, 2001). Energy analysis has been primarily done too late in the design process, while the most effective decisions can be made earlier in the process (Samuelson et al., 2016). Therefore, energy modeling has to be performed in the early design stage to make better decisions (Anderson, 2016).

Simulation of daylighting performance is highly related to the recent escalated focus on energy efficiency and environmentally-conscious building design (Elghazi et al., 2014; Lagios et al., 2010). In such a performance-based design approach, there is a current need to integrate platforms that run simultaneous environmental simulations of daylight analysis and energy simulation (Roudsari et al., 2013). Therefore, daylight analysis has been targeted as a performance objective in addition to energy simulation in Test-case 3 of this work as samples of design performance evaluation sought for optimization.

2.2.2. Design Optimization

In decision-making, it is often required to simultaneously consider several criteria or objectives during the design process, thus, multi-criteria or Multi-Objective Optimization (MOO) is approached (Hauglustaine & Azar, 2001; Radford & Gero, 1987; Wang et al., 2005). MOO leads to finding optimal solutions for any problem with multiple objective functions and results

in an optimal set of solutions compared to one solution in single-objective optimization (Collette & Siarry, 2013). Mathematically, optimization refers to the process of finding the best solution from a set of different alternatives (Nguyen et al., 2014). In terms of building performance, optimization does not require finding the absolute optimal solution to a problem, as it could be unfeasible for particular problems or for certain simulation programs; rather, optimization indicates an iterative process of improvement using simulation tools to achieve sub-optimal solutions. Thus, it is usually accepted in building performance simulation to use the term optimization to indicate an automated process based on mathematical optimization and numerical simulation (Nguyen et al., 2014).

There are two approaches to solve MOO problems: 1) combine the multiple objectives into a single (composite) objective by determining a weight for each objective and using the weighted sum method, and 2) determine a Pareto optimal solution set, which is a representative set of solutions that are non-dominated by each other (Konak et al., 2006). When a change to a parameter value of a solution leads to an improvement in one objective without making the other objective worse off, it is considered a Pareto improvement; when no Pareto improvements could be made, the solutions are called Pareto optimal (Radford & Gero, 1987; Yan et al., 2015). (Figure 2-1) depicts an abstracted two-dimensional performance search space that includes two objective functions with the dominated solutions (orange) shown in the grey area and the non-dominated or Pareto solutions (green) represented along the Pareto front. The figure is an adaptation of the Pareto ranking method developed by (Fonseca & Fleming, 1993). The ultimate aim of a MOO process is to find the Pareto optimal solutions set or Pareto Front (Konak et al., 2006).

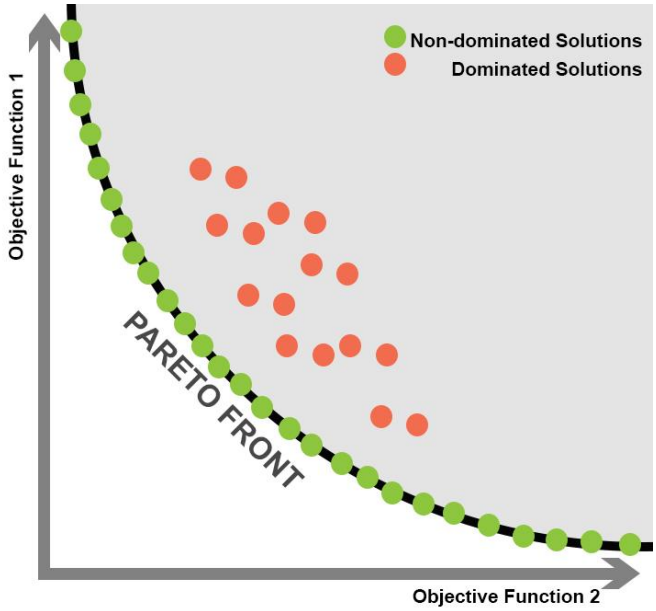

Figure 2-1. 2D Search space with two objective functions and the Pareto front curve with its non-dominated solutions in green, and the feasible area in grey, with the dominated solutions in orange. By convention, the minimums of Object Function 1 and 2 (origin in the coordinate system) are considered optimal.

Evolutionary Algorithms (EAs), biologically-inspired algorithms and considered a branch of computer science (Yi et al., 2012), are commonly used to solve MOO problems. An EA process begins with a population, often initiated randomly (Konak et al., 2006). Each candidate in the population is tested against fitness functions to determine whether it is an attractive candidate. The population is culled to include only attractive candidates, which are then combined to produce the next generation population. This new population is then used as the initial population in the EA process and the process repeats until satisfactory solutions are found (Yousif et al., 2018). The operations of EAs exploit the characteristics of good solutions according to different objectives and create new non-dominated solutions in parts of the Pareto front; such characteristics make EAs the most attractive approach to MOO problems (Konak et al., 2006). A Multi-Objective Evolutionary Algorithm (MOEA) employs Pareto optimality to assist the optimization of all the objectives (Fonseca & Fleming, 1993). Since it is a commonly

used method, and a promising generative tool, in this work, an MOEA-based tool was tested and aimed to be used for performance evaluation and optimization in the application of the overall form-finding process in Test-case 3.

2.2.3. Issues in Current Generative and Design Optimization Systems

The methods and tools used for optimization and integrated into generative systems, are promising approaches for the architectural design process, yet they are not always performed in a formalized way (Wortmann & Nannicini, 2017). A significant drawback in generative protocols is that "they may produce an excessive number of solutions for a human to cope with" (Rodrigues et al., 2017, p. 48), which can be overwhelming for designers, in terms of decision-making and interaction with the system. Collectively, available generative design systems lack articulation mechanisms of the design shapes produced in the process (Brown & Mueller, 2019). An example of a large set of solutions in design optimization systems is the Pareto front illustrated in (Figure 2-2) that shows hundreds of data points as Pareto optimal solutions which satisfy three objectives: cost, weight, and performance (Kalvelagen, 2015). More importantly, for designers, it is not feasible to rate the design solutions based on their performance and select the best ones for unclear and subjective problems such as evaluating the designs' form characteristics; alternatively, it would be advantageous to organize the produced designs into groups based on common features (Rodrigues et al., 2017). Yet, organizing the design solutions produced in generative schemes, based on their shape characteristics, is still under-addressed in research.

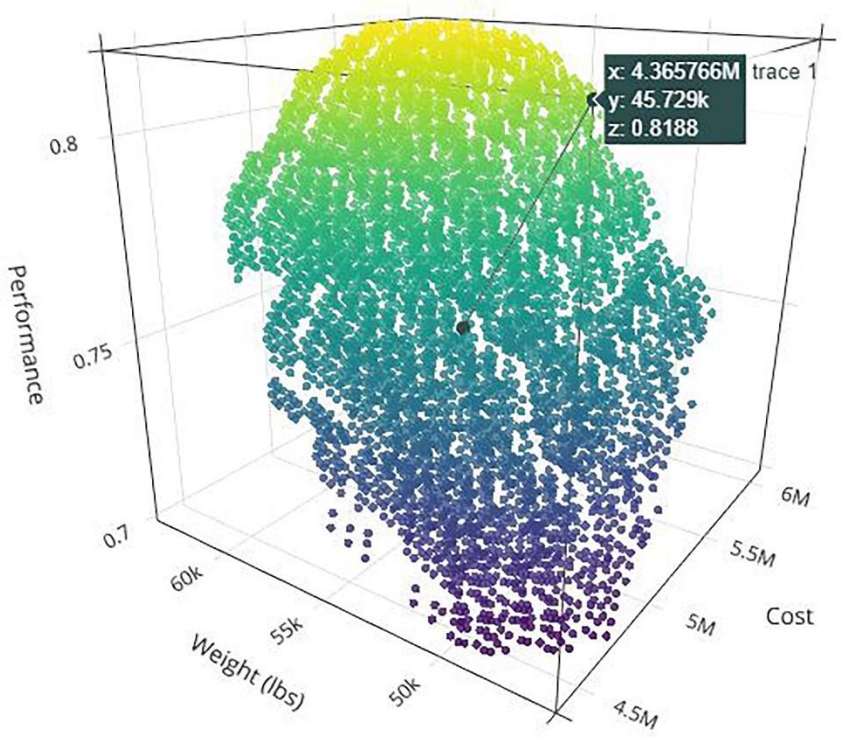

Figure 2-2. The test-data solution set of Pareto Optimization with three objective functions; each point is an optimal solution. Reprinted with permission from (Kalvelagen, 2015).

One of the relevant approaches that have been investigated for this research, and used inside the MOEA-based generative tools is genetic diversity. In such methods, given that none of the Pareto front solutions is better than the other ones in satisfying all the objectives, the goal in searching for fitter solutions becomes to find as many Pareto solutions as possible. This may lead to premature convergence of the population set towards certain areas in the solution space (Toffolo & Benini, 2003). Consequently, there is a need to maintain diversity during the search mechanism, to find truly diverse Pareto solutions. Except for the Strength Pareto method of (Zitzler and Thiele, 1999), existing MOEAs generally use two distinct methods to solve the problem of early convergence towards the Pareto-optimal set and to maintain genetic diversity. The first method is an approach to assign individual candidates a fitness value other than their

performance value, and the other method corrects the assignments to assure diversity through introducing artificial parameters that need to be determined a priori in unknown fitness objectives (Toffolo & Benini, 2003). In these approaches, the selection issue is mainly addressed towards the Pareto-optimal set, and its early convergence effects are reduced by the introduction of diversity-preserving mechanisms (Toffolo & Benini, 2003). However, this genetic diversity is problematic in terms of the shape characteristics as described in the following subsection.

2.2.4. The Problem with Genetic Diversity in Evolutionary Algorithms

Evolutionary Algorithms, often used by designers in optimization studies, includes a genetic diversity method. However; this genetic diversity does not necessarily lead to a diverse set in terms of the shapes of the generated design set. According to Brown & Mueller (2019), EAs operate a crowding distance to make sure that the solutions in the objective space are scattered and representative along the Pareto front (Deb et al., 2002). The concept of distance-based diversity measure relies on the fact that an individual that is distant from all the others has more chances, when mating, to produce offspring in regions of the search space not covered by the current population (Toffolo & Benini, 2003). Such a distance metric can be calculated mathematically using the (normalized) Euclidean distance in the objective function space, and the measure of diversity of an individual is the sum of its distances from all the other individuals (Toffolo & Benini, 2003; Yousif et al., 2017). The issue of this stance-based genetic diversity in the objective space is explained by Brown & Mueller (2019) as follows:

> *"This distance is only considered in objective space and does not directly translate to diverse solutions in the design space, meaning that designs with different performances could still look the same and not be diverse enough for explorative design. Many optimization techniques are similarly limited by focusing only on differences in objective function values."*

This entails that there is a need to find another method to achieve diversity in optimization methods in generative design. It would be promising to find approaches that lead to highly diverse design shapes that represent other similar ones. Searching for an articulation mechanism that is capable of so doing has led to investigating and experimenting with shape comparison and clustering methods, particularly the representative object-based clustering algorithm of K-Medoids described in Subsection 2.3.4.

2.2.5. Interactive Evolutionary Computation

The area of Human-Computer Interaction or (HCI) began in the 1980s when the interaction was an issue worth of research (MacKenzie, 2012). HCI is an extensive research field often associated with advancing software, interfaces, or processes that people use to develop a new or improved interaction (MacKenzie, 2012). The challenge or the complexity in HCI lies within the human factor, due to variabilities in humans while computers, by comparison, are simple. The interaction happens when the user performs a certain task using the computer and according to elements such as the control-display relationships, in addition to other elements (MacKenzie, 2012). In his article: *"Designing Interaction, not Interfaces"*, Beaudouin-Lafon argues that the emphasis in advancing computers has been on the interface while a shift in focus is needed to target designing successful interaction (2004).

A relevant area to HCI is Interactive Evolutionary Computation (IEC) (Harding & Brandt-Olsen, 2018). In terms of parametric generative design, the process of programming and the inclusion of search mechanisms and evolutionary methods in an interactive mode should be guided to mediate and adapt to the user's interaction (Harding & Brandt-Olsen, 2018). The important question becomes "how can a designer better interfere with computational heuristics and understand the search struggle" (Harding & Brandt-Olsen, 2018, p. 145). IEC has become

important for designers. While heuristic search algorithms use objective functions to evaluate designs at each iteration, interactive selection replaces this by designers' participation, who are required to make explicit their design intents and constantly change and interfere in the evolutionary process (Harding & Brandt-Olsen, 2018).

In this research, facilitating IEC is focused on two modes of interaction. The first mode is hybrid, in which the shape clustering method is used intermediately in the search process where the evolutionary search can be stopped and the SC-KM method is applied. Next, the organized design set or selected subsets can be used for further search in an iterative back and forth mode, alternative between design optimization and clustering. This way, the clustering method facilitates the designers' interaction in the interference in evaluating the emerged designs in generative and optimization systems in terms of the geometric characteristics of those designs. To do so, according to Harding & Brandt-Olsen, whereas large populations are favorable for generative and objective-based evolutionary computation, smaller populations are often used for interactive evolution, thus at each generation (iteration), the population can be reduced in size to achieve such interaction (2018). This mode is still under investigation in our study. In another mode, the SC-KM can be applied post to design optimization or after terminating the search mechanism, as demonstrated in this research case studies. In both ways, the clustering allows for management of the large population and leads to an in-depth examination of the organized and reduced design set, and the methods and tools for this clustering are explained in the following subsections.

2.3. Shape clustering and Design Presentation

The focus of this research project is to develop a new articulation method to find geometric dissimilitude amongst the produced design shapes, and to articulate (cluster) the designs

according to this dissimilitude measure, to be implemented into the form-finding process. Accordingly, the complete method of shape clustering consists of sub-tasks to (1) implement a shape description/representation method and find and compute a shape geometric difference, and (2) employ and perform a clustering method that is based on the computed shape difference. Ultimately, the resulting clustered shapes represent the articulated set, visualized and presented to the designers for facilitating their examination of designs. The following subsections explain and introduce the most influential methods and algorithms used for the developed shape clustering method.

2.3.1. Shape Description

Shape description, or representation, is a method concerned with defining and retrieving effective and perceptually important shape characteristics, based on either shape boundary or/and interior features (Zhang & Lu, 2004). Multiple shape descriptors have been designed and researched, and often assessed by how accurately they describe and retrieve similar shapes from the analyzed set (Zhang & Lu, 2004). Such studies that define shape representation techniques include a shape boundary descriptor, e.g. (Joshi & Srivastava, 2003), or other shape descriptors such as the point distance, turning function, grid-based model, and tangent distance descriptors in (Rodrigues et al., 2017)'s work. Another approach to shape representation is to find the common structure that captures the structure information for a set of shapes based on a skeleton graph, it is called Common Structure Skeleton Graph (CSSG) (Shen et al., 2013).

According to (Zhang & Lu, 2004), certain characteristics should be considered when describing shapes, identified as follows:

- It is advantageous if shape descriptors have a hierarchal (coarse to fine) representation feature, leading to a high level of matching efficiency. When

matching occurs at a coarse level, it leads to the elimination of a large set of similar/dissimilar shapes, while other shapes can be further matched at a finer level.

- It is favorable for shape descriptors to be applied to a wide range of shape types, instead of performing well only for certain types of shapes.

In classifying shape description techniques, two main categories can be identified: contour-based, and region-based; both categories can be subdivided into structural and global-based sub-classes, and each sub-class contains multiple approaches (Zhang & Lu, 2004). The global class of region-based representations includes a grid-based description. The grid-based shape description method has been further investigated for this study because it satisfies the desired characteristics of well-performing shape descriptors described above.

- **Grid-based Shape Description**

An approach to grid-based descriptions has been proposed by (Lu & Sajjanhar, 1999) and implemented in studies e.g. (ChakrabartiOrtega-BinderbergerPorkaew & Mehrotra, 2000; Huang et al., 1997; Safar et al., 2000). In grid-based shape description, a grid of cells is overlaid on the shape under analysis; the grid is scanned from the left to the right corner, in a top to bottom direction resulting in a bitmap; it is a binary assignment with the cells overlapped by the shape assigned 1, and those not overlapped by the shape given 0 (Zhang & Lu, 2004). Afterward, the shape can be described as a binary vector, and the Hamming distance is used to calculate the similarity between two shapes (Zhang & Lu, 2004). The Hamming distance for two strings of equal length represents the number of locations at which the corresponding symbols are dissimilar (Norouzi et al., 2012). \It is relevant to note here that reviewing the grid-based description method in (Rodrigues et al., 2017)’s work, the overlaid grid has been scanned

according to (Sajjanhar & Lu, 1997)'s work, from left to right, top to bottom, considering the grid-cells' center-points for correspondence, leading to a vector-based code as a matrix.

An example of grid-based shape representations is depicted in (Figure 2-3). To prepare shapes in the grid-based method for scanning, they need to be normalized, to accommodate other transformation operations of scaling, translation, or/and rotation; however, mirrored and flipped shapes are considered different (Zhang & Lu, 2004). A significant task is to scale the shapes under examination to a fixed bounding rectangle, moved to the upper left corner, and preferably rotated to get the primary shape horizontal axis (Zhang & Lu, 2004).

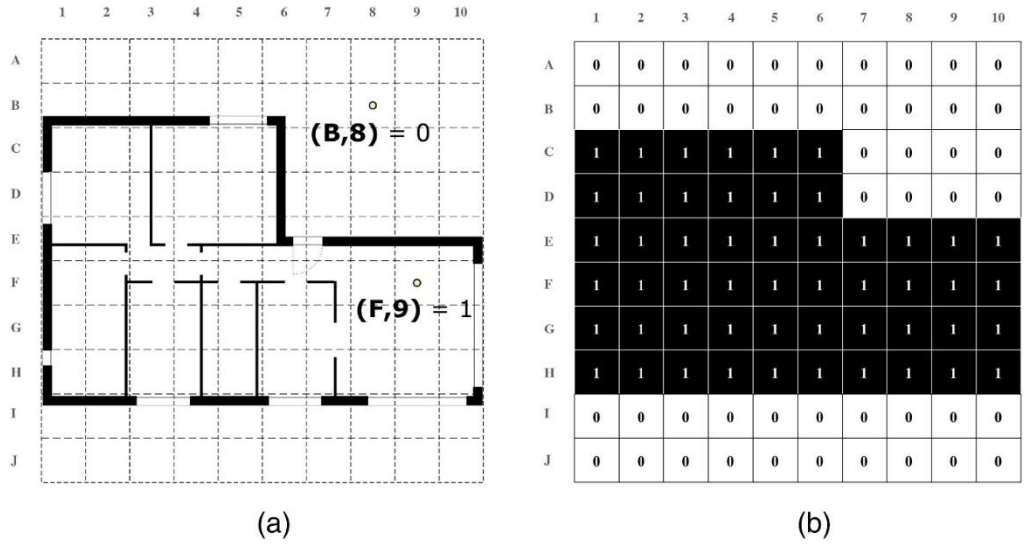

Figure 2-3. An example of grid-based shape description. Left: a grid overlaid on an architectural floor plan; right: a matrix of 0s (cells outside the shape) and 1s (cells inside the shape). Reprinted from Automation in Construction, 80, Rodrigues et al., Clustering of Architectural Floor Plans: A Comparison of Shape Representations, Pages 48-65, Copyright 4585091348148, with permission from Elsevier.

The main advantages of the grid-based method include its simplicity in representation, its characteristic to satisfy and agree with an intuitive visual examination, as well as its agreement with the shape coding in MPEG-4, which makes the grid-based descriptor attractive for shape analysis (Zhang & Lu, 2004). The term MPEG-4 refers to a method of defining compression of visual digital data that has been introduced and designated as a standard for a group of video

(and audio) coding formats and related technology, agreed upon by the ISO/IEC Moving Picture Experts Group (MPEG) (Wiegand et al., 2003).

However, the main drawback of this grid-based method is the major-axis based rotation normalization which makes it sensitive to noise (Zhang & Lu, 2004). Improvements to grid-based shape descriptions have been proposed by studies, including the work of (ChakrabartiOrtega-BinderbergerPorkaewZuo et al., 2000) that utilizes an adaptive resolution representation acquired by implementing quadtree decomposition on the grid (bitmap) of the shape. Quadtree grid decomposition is a method used to describe a class of hierarchical data structures with common property based on the recursive decomposition of shapes (Samet, 1984).

In response to the potentials of the grid-based shape description, and the possibility to improve the method, it was selected to be carried out for defining shape representation in this work. After determining the shape description method, there was a need to define a shape difference analysis measure, in terms of geometric correspondences. Such a shape difference retrieval method has also to be employed successfully to attain accurate shape clustering, as explained next.

2.3.2. Shape Difference Measure

In clustering shapes, geometric analysis of the shapes is often conducted, in order to identify the shape characteristics (Joshi & Srivastava, 2003). The reason for pursuing a method to define shape difference in this study was due to the fact, supported by research, that a quantified definition of shapes is difficult but a quantified definition of shape difference is possible. In using the grid-based shape description, a particular set of procedures need to be carried out for shape comparison and shape difference retrieval. In grid-based shape description, after applying transformation procedures for scanning the shapes and retrieving the binary

number of 0s and 1s (i.e. indexing), shapes are subjected to a similarity/dissimilarity measure (Sajjanhar & Lu, 1997). The difference between two shapes can be computed as the number of cells in the overlaid grids that are overlapped by one shape but not the other; an important criterion here is to normalize the number of cells for the overlaid grids prior to indexing and comparison for accuracy reasons (Sajjanhar & Lu, 1997). In the case of a different number of cells between the compared shapes, also means different decimal number, the length of this binary number has to be fixed for all shapes and this can be done by adding trailing zeros to the binary number (Sajjanhar & Lu, 1997).

As a common measure in grid-based shape description, the Hamming distance calculation is used; in considering two binary strings or codes of equal length, the number of non-corresponding positions between the two is the Hamming distance (Robinson, 2008). In other words, it is a similarity/difference measure (Norouzi et al., 2012) of the minimum number of changes required to match one code to the other, or the minimum number of errors that can transform one code into the other (Robinson, 2008). The method is used by (Sajjanhar & Lu, 1997) with a query or a reference shape, against which all the shapes under analysis are compared. The similarity measure for shapes is calculated by sequential comparison of the shapes' binary numbers against the query shape; the sequence of the 1s (for the cells of the grid wholly covered by the shape), and 0s (for the cells outside the shape). To measure the difference between two shapes, the number of cells in the grids that are overlaid by one shape and not the other shape is computed, and this calculation is intuitively accurate as the maximum matched shapes are those with minimum distance from the query shape (Sajjanhar & Lu, 1997).

Distance-based methods are fundamental to design the clustering method and can take advantage of optimization techniques (Han et al., 2011); optimization techniques refer here to

calculating the distance of non-overlapping cells in the most efficient way. Importantly, in an early attempt at developing the shape difference method in the published work (Yousif et al., 2017), I used a similar method to the Hamming distance considering only the number of the non-overlapping cells as a measure that indicates shape difference for a given set of shapes. Yet, in further developing the method, it became clear that the number of non-overlapping cells is not always the accurate difference measure of shape comparison. Rather, the sum of distances between the non-overlapping cells was considered as the actual difference measure of the pair.

To obtain a reasonable distance measure for shape analysis, a problem appears in pair-wise shape comparison on the best matching of the non-overlapping cells between two shapes for similarity/dissimilarity measure. As such, another algorithm is needed to solve this assignment problem, explained in the following sub-section.

- **The Hungarian Algorithm**

Developed by Kuhn and later revised by Munkers, the Hungarian algorithm is a combinatorial optimization algorithm, often used to solve assignment problems, applicable to many fields (Korsah et al., 2007). It can be explained using the following example: when given a matrix of workers and the cost of each worker to perform a certain job, the Hungarian algorithm can find the best possible assignment of workers to jobs so that the total cost is minimized (Korsah et al., 2007) as illustrated in (Table 2-1). The best assignment of jobs per employees to get the minimum sum of costs is: (employee 1-job 2, employee 2-job 1, employee 3-job 3) leading to ($8+$9+$8=$25) as the lowest cost, while other assignment combinations would lead to a higher cost.

Table 2-1. A matrix of cost, an assignment problem example to be solved by the Hungarian algorithm. The best assignments of job per employee for the lowest cost is indicated in red numbers inside grey boxes.

	job 1	job 2	job 3
employee 1	12$	8$	10$
employee 2	9$	11$	12$
employee 3	10$	12$	8$

In this study, the Hungarian algorithm will be used to match the geometric components (cells in grid-based shapes) between each pair of compared shapes, in order to find the smallest overall sum of Euclidean cell distance between the pair, for calculating the pair's shape difference measure. In particular, the Hungarian algorithm is needed to be incorporated inside the pair-wise shape difference functionality that will be explained in Section 4.

2.3.3. Cluster Analysis

As a method, cluster analysis has been used for more than 80 years, established by Zubin (1938) and Tryon (1939) in the field of psychology, as wells as in anthropology by Driver and Kroeber (1932). Prior to the computer's invention, cluster calculations were very difficult, hence, the development of clustering methods was made possible through and after the use of computation in the 1950s (Bailey, 1994). Clustering is the process of partitioning or separating a set of data objects into multiple subsets or clusters in a way that objects within a cluster have high similarity, yet are very dissimilar to objects in other subsets (Han et al., 2011), and the objects' identities are unknown in advance (Wilks, 2011). Generally, the correct number of clusters in which the objects should be grouped is also unknown beforehand, therefore, in K-Means and K-Medoids the number of clusters is pre-determined with estimation. Primarily, it is

the degree (measure) of similarity and dissimilarity amongst the objects that determines the groups and assigns the group membership (Wilks, 2011).

To assess similarity and dissimilarity, the attribute values that describe the object are computed and this task often involves distance calculation (Han et al., 2011; Wilks, 2011). According to (Han et al., 2011), cluster analysis techniques can be classified into 4 categories: (1) partitioning methods including K-Means and K-Medoids clustering, (2) hierarchal methods of which the Ward method used in (Rodrigues et al., 2017)'s study is an example, (3) distance measure, and (4) grid-based methods. Clustering leads to discovering unknown patterns (groups) within the observed data (Han et al., 2011; Velmurugan & Santhanam, 2010; Wilks, 2011).

It is relevant here to briefly explain the Ward linkage clustering method, an agglomerative hierarchal decomposition of a given dataset, based on a minimum variance criterion or objective function to minimize the total within-cluster variance (Han et al., 2011). It begins with each object forming a separate cluster, and a successive merge among the objects with close distances occur in a hierarchal manner until all the groups are merged through multiple steps into one top-level of the hierarchy, then termination happens (Han et al., 2011; Ward Jr, 1963). The minimum variance criterion considers squared Euclidean distances between the examined objects. Overall, hierarchal methods suffer from rigidity due to the fact that once a step of merge occurs, it is irreversible, yet this also leads to comparatively less computational burden as the method does not require combinatorial consideration of the possible choices (Han et al., 2011). Primarily, the method does not support correcting erroneous decisions, and several methods have been proposed to improve hierarchal clustering (Han et al., 2011).

As a tool and a branch in statistics, clustering has been widely studied, particularly focusing on distance-based cluster analysis. K-means, K-Medoids, and other clustering methods

have been applied and became part of statistical analysis software packages (Han et al., 2011). Also, clustering methods have applications in "data mining, statistics, Machine Learning, spatial database technology, information retrieval, Web search, biology, marketing, and many other application areas." (Han et al., 2011, p. 445). For Machine Learning, in classification, since class label information is given in advance, the learning is supervised, while in clustering, there is no previous label to the cluster, and this makes clustering categorized as an unsupervised technique (Han et al., 2011).

2.3.4. The Representative Object-Based, K-Medoids Algorithm

There are certain clustering algorithms developed and investigated in research, of which K-Means and K-Medoids are considered significant (Velmurugan & Santhanam, 2010). K-Means is a simple clustering algorithm that can solve well-known clustering problems through partitioning the data set into a number of clusters (k) (Velmurugan & Santhanam, 2010). The algorithm proceeds by calculating k initial cluster centers (means) and iteratively refining them by minimizing the distance between each medoid and the objects grouped within its cluster as follows: (1) each datum (object) is assigned to its closest cluster, (2) each cluster center is updated to be the mean of its constituent data, and (3) the algorithm then converges when there is no further update in the assignment of data to clusters (Wagstaff et al., 2001).

In K-Medoids clustering, the algorithm considers a Medoid, which is the most centrally located object in each cluster, as an alternative to the mean in K-Means; since the means can be easily impacted by extreme values, this makes the K-Means algorithm sensitive to outliers and can significantly distort the partitioning (Velmurugan & Santhanam, 2010). K-Medoids clustering is more robust to outliers and more efficient compared to K-Means (Jin & Han, 2016). It is based on a Partitioning Around Medoid (PAM) method that minimizes the sum of

dissimilarities between each object and its corresponding reference or Medoid (Velmurugan & Santhanam, 2010).

To put it simply, the basic concept behind K-Medoids can be explained as follows: the algorithm selects *K* representative points to create initial clusters and then repeatedly update to find better cluster representatives (Jin & Han, 2016). Importantly, all the possible combinations of medoid (representative) and nonrepresentative points can be examined, and the resulting clustering is calculated for each pair of medoid and non-medoid. The original representative point is updated (replaced) with the new point which causes the greatest reduction in distortion function; at each iteration, the groups of best points for each cluster create the new respective medoids (Jin & Han, 2016). In (Figure 2-4), the workflow and main procedures of a typical K-Medoids algorithm are illustrated, based on (Jin & Han)'s work.

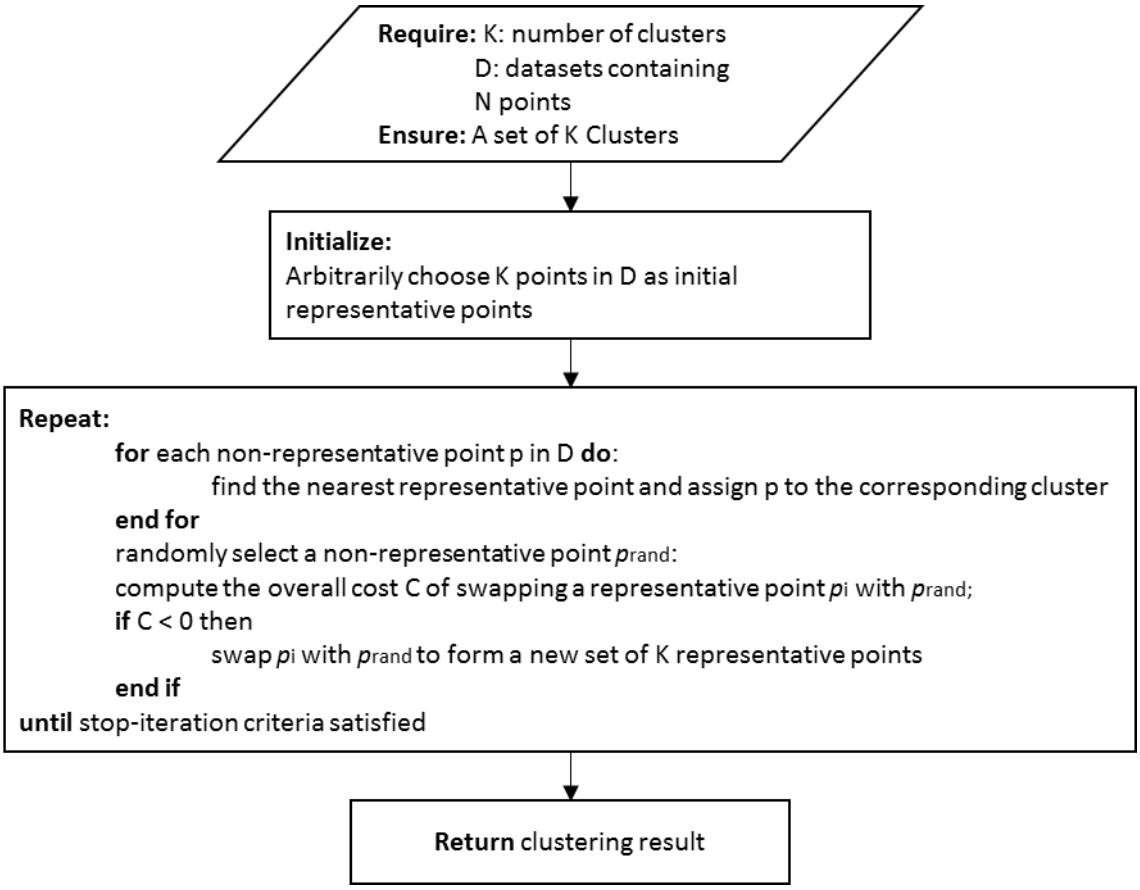

Figure 2-4. A typical K-Medoids clustering algorithm. Adapted from (Jin & Han, 2016).

Finding the Medoid is very useful for a generative design system because it is the representative of a cluster of design options and an actual design (in contrast, the Mean in K-Means is not an actual design but a calculated mean of design options). Also importantly, the K-Medoids algorithm is faster and saves computation time when compared to the K-Means, for cases of normal and uniform distributions of the given data points (Velmurugan & Santhanam, 2010).

The rationale for using K-Medoids in this research can be explained as follows: (1) the K-Medoids algorithm uses data points' dissimilarities for clustering and it minimizes the sum of general pairwise dissimilarities; the possible choice of the dissimilarity function is very rich, however, in our application we used the Euclidean distances of cells in compared pairs of shapes,

(2) the Medoids are representatives for clusters, and thus in our method they are actual design solutions, and (3) the K-Medoids algorithm doesn't need data points (or quantified shape definition, which is difficult to define), but their differences and this matches the objective in this work.

2.4. Summary of the Section

In its first part, this section offered a literature review on parametric form generation and generative design systems, methods and tools for design evaluation and optimization. Afterward, the limitations and the issues accompanied in using such methods were identified. Next, the suggested method to be incorporated into generative systems was introduced. The method constitutes a method for shape representation, a method for shape difference measure and the cluster analysis method, and the related and inner components and algorithms were also introduced.

3. RESEARCH METHODS

The research methods used in this study can be identified as mixed-methods, comprised of literature study, experimentation, prototyping, testing, and validation of findings. These methods can be explained through John Kunz's "maximum anxiety heuristic" organizing principles for building a model-based (computational) system. In Kunz's approach (1989), to develop a system, it is significant to adhere to the following protocol: *(1) identify the purposes of the system, (2) describe the representation or model of the domain, (3) specify the reasoning to analyze the model, (4) create an interface, and (5) test the model validity*. These five tasks have to be developed simultaneously in order to be perceived at an equal level of maturity at each phase throughout the process of system development, and this approach is called the "maximum anxiety heuristic" (Kunz, 1989, p. 1). To explain the "maximum anxiety heuristic" principle, Kunz (1989) defines it as an organizing rule to direct the developer of a system to focus on these tasks in a way that the attention should be on each task until some other tasks appear to be ill-defined or cause greater anxiety. The advantage of using this approach is that it provides a direct and effective method for guiding the system development, through explicitly addressing the issues of each developed activity/task so that it helps to reveal and explain the other tasks' issues (Kunz, 1989).

In this study, the system framework was developed using Kunz's method, following his heuristic. The five processes of system development can be explained as follows (Kunz, 1989):

1. *Identify the purposes of the system:* In developing a knowledge-based system, there should be a clear purpose of that system, from two perspectives, the perspective of the expert/developer and of the user/designer. For a computational system, it is a good design

goal to target users with good general domain knowledge and basic computation skills. The purposes here refer to the objectives of the system, implemented as its top-level capabilities that can be performed and achieved by the user of the system. It is important to differentiate these purposes from the tasks that the system can achieve; the goals of developing the system are to potentially achieve those tasks.

2. *Describe the representation or model of the domain:* In this activity, the researcher has to identify and describe the domain where the research problem exists, and the overall concepts or methods used in the domain and their potentials and limitations as described by experts. Also, in domain representation, it is helpful to identify the specific existing examples of the concepts that can be found in a modeled system that attempts to solve the research problem. The concepts here can be viewed as the actions or functions described in the system and their corresponding objectives.
3. *Specify the reasoning to analyze the model:* Here, the system's developer has to identify the system's behavior, the criteria by which the problems have to be analyzed, and the decisions to be taken for each problem in the system. It is essential to define the functionalities and the criteria that would be applied to evaluate those functionalities. Certain decision-making criteria can outline implementations in a specific discipline, such as developing algorithms for solving computational issues. The system's behavior includes procedures and illustrates the structure or workflow of the system. Reasoning has to accompany the activities or functions performed by the system, providing guidelines on the attributes that have to accompany those functions.
4. *Create an interface:* Knowledge systems often include user interfaces as platforms of data and destinations of the results. The primary purpose of the user interface is to

communicate the system to the user, collect and receive data, and convey the results. The user interface is not necessarily required to be a stand-alone application; it is aimed to present and justify the reasons for the assumptions and decisions made when developing the system.

5. *Test (create procedures to test the model validity):* Any developed system must be tested throughout its development to verify and assure its validity and to decide if changes are needed when problems occur. Testing includes defining criteria for choosing particular test-cases, and ideally for selecting a standard procedure verifying the accuracy of the cases being tested. When a standard procedure is carried out to test multiple cases, it is important to find the expected behavior of the system at each test-case and to identify the procedures to compare the output of the system of all cases.

Following Kunz's heuristic, the processes carried out in this study for developing the system are illustrated in (Figure 3-1) and can be each described as following subsections.

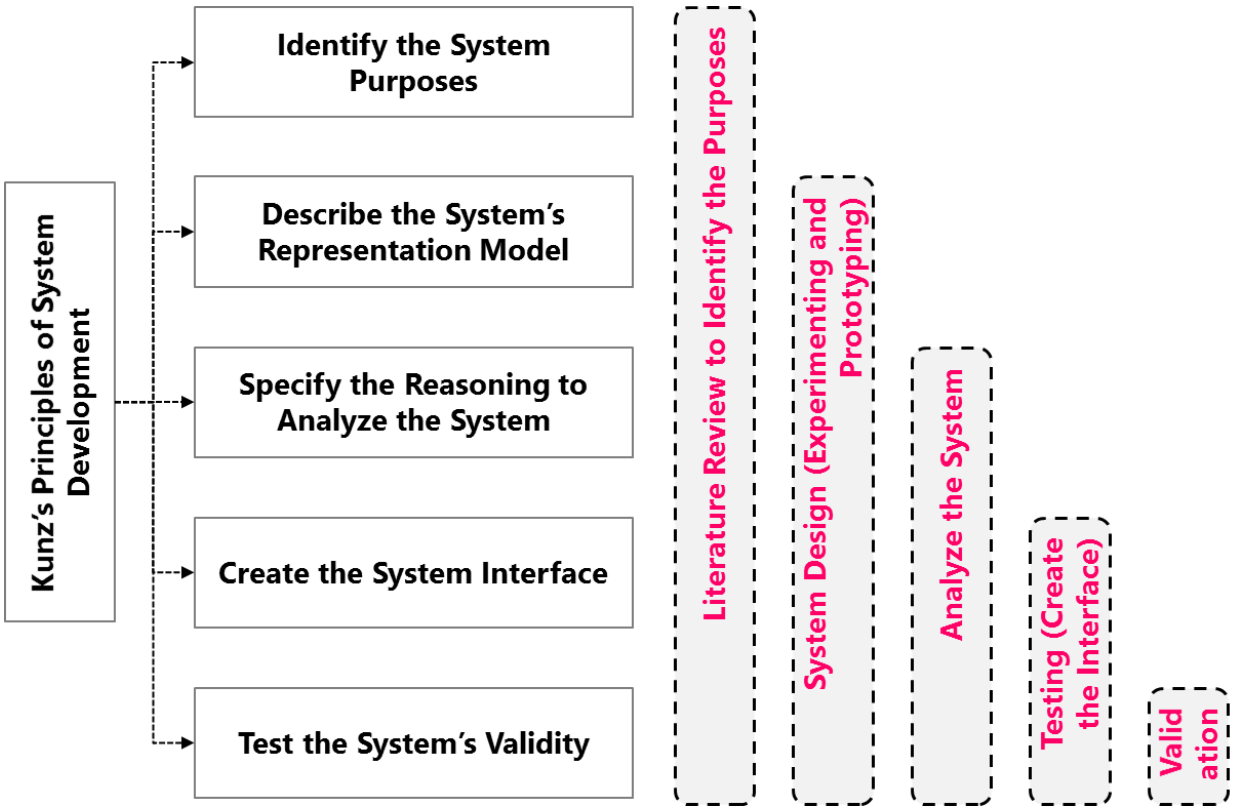

Figure 3-1. The Research methods used for the system development, following Kunz' Principles.

3.1. Literature Review to Identify the System's Purposes

To identify the research problems, and the purposes of the system to solve those problems, an extensive literature study was continuously carried out throughout the five tasks, focusing on the most recent relevant work. Analyzing existing studies, the emphasis was on potentials and limitations of available articulation methods in form finding of generative systems, if there are any. A list of criteria has been utilized to determine the gaps and problems of existing methods, and was categorized as the following:

- *The Domain of Generative Systems:* The domain of our system is identified as computational generative systems such as parametric modeling systems in which hundreds, thousands or a higher number of design options are generated.

- *Users of Generative Systems:* The users here are designers, the audience who uses generative systems to create and explore a wide range of design alternatives, and often possess advanced design knowledge, yet basic to moderate computational skills. Therefore, the system developed in this work is to be introduced to such an audience and has to be modeled to be run successfully by those designers.
- *Problem-Specification in Form Finding:* In form finding, designers face an issue of an overwhelming number of design alternatives that emerge when running the system, as identified by multiple studies, e.g. (Brown & Mueller, 2019; Rodrigues et al., 2017). As a consequence, the designers find it difficult to control the system when too many solutions evolve making it unfeasible to examine the solutions' formal qualities and to interact with the system.
- *Lack of Available Articulation Methods:* Analyzing literature, the lack of articulation methods in form finding remains an unsolved issue. Therefore, the system's functions and tasks have been determined to lead to solve this problem and to achieve a successful articulation of the design options in terms of geometric correspondences.
- *Use of Shape Clustering Methods:* The purposes of building the system in this work are to develop and incorporate a new shape clustering method in form finding. It is important to note that shape clustering methods are not commonly used in generative design or optimization systems.

These criteria helped shape the system's purposes and suggested the functions needed inside the system, particularly the need to develop an original method for clustering to support successful form finding.

3.2. System Design (Experimenting and Prototyping)

To represent the design process, a system's framework that includes sufficient expressive capability to capture the characteristics of the ideas that support the process can be used (Gero, 1990). Schemas or system prototypes are often used to represent design knowledge and demonstrate the framework (Gero, 1990). Design prototypes are conceptual schemas for representing a family of abstracted groups of components derived from similar design cases, and behave similarly to the design process, with the a priori knowledge required for the design situation, all combined into one schema (Gero, 1990). The use of design prototypes can be described as "matching a cognitive view of a process model of design" (Gero, 1990).

The use of prototyping allows for separating architectural design knowledge from the computational processes that operate upon that knowledge; such representation is usually effective as an interpreter that articulates between the structure or the syntax of design, and the function or the semantics of a design (Gero, 1990). This design articulation is advantageous in producing designs as well as in analyzing and evaluating those designs (Gero, 1990). The aim of using software tool prototypes is to meet the expectations of a designer who uses computational processes for producing designs as it provides a framework that assists in both routine and nonroutine design processes (Gero, 1990).

In this study, to achieve the suggested system's purposes, prototyping became essential to demonstrate the functionalities of the system and to serve as an apparatus by which applications are tested and data are collected. The system's tasks were formulated and connected together to create a fully working prototype of a generative system that performs all the tasks seamlessly and with reasonable computational time. The prototype comprises routine, and nonroutine (including innovative) processes, as described by (Coyne et al., 1987). A routine process can be described

as a well-defined state space of potential designs, in which all the variables and the knowledge to calculate their values are based on existing prototypes (Coyne et al., 1987). An innovative process, differently, can be defined as a "design prototype-instance adaptation" with modifications of some of the knowledge regarding applicable values of the variables. A nonroutine, creative process involves introducing new variables into the prototype, leading to produce a new prototype (Coyne et al., 1987). Following this categorization, in developing the prototype of this study, three processes have been pursued of which two are considered routine and one is innovative; the parametric form generation and design optimization are routine processes, and shape clustering is the innovative process, all illustrated in (Figure 3-2).

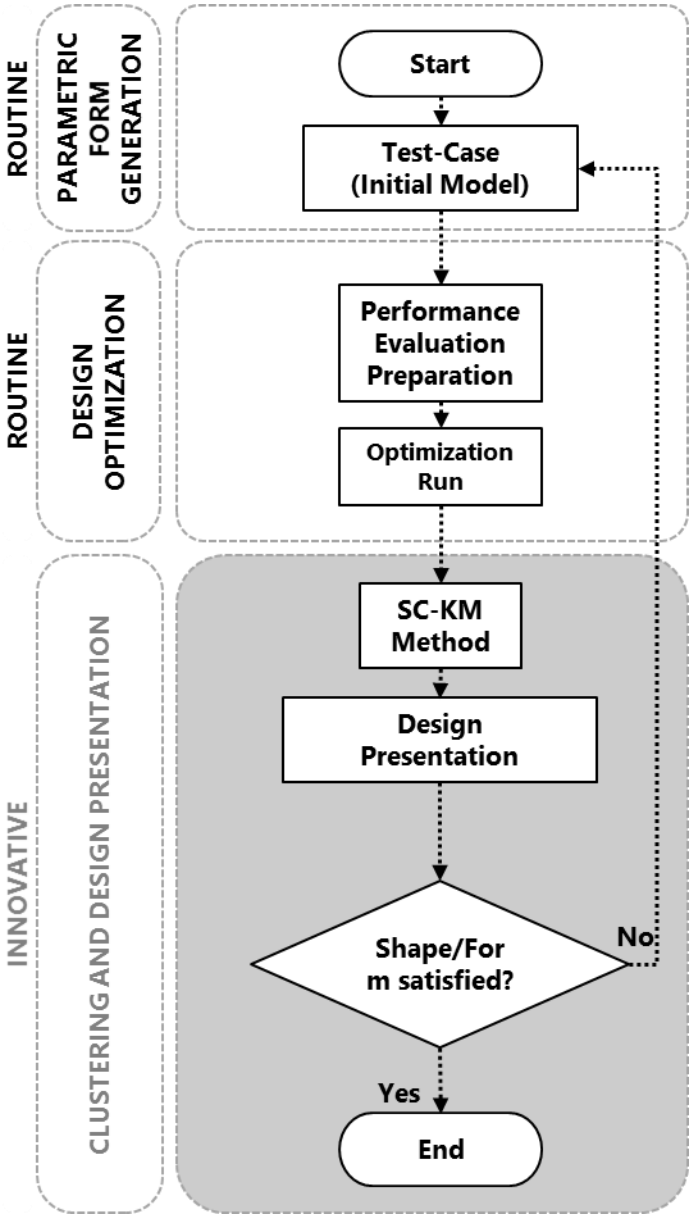

Figure 3-2. Workflow of the proposed system.

The three processes constitute the prototypical protocol explained in the following subsections.

3.2.1. Process 1: Parametric Form Generation: (Routine)

According to (Coyne et al., 1987) 's categorization, this process is routine-based, as it comprises a typically-used activity of parametric modeling to set-up the initial mass model. To

create the parametric model, an architectural design project is modeled and parametrized. The initial parametric model setup involves establishing the constraints and variables that control the model and allow for a variety of alternatives to emerge.

3.2.2. Process 2: Design Optimization (Routine)

This is a routine process as well, where an optimization tool is used, coupled with building performance evaluation tools. The process involves the use of a generative tool that allows for the model parameters to change and captures and evaluates each resulting design options at every iteration. Before running the tool, the parametric model and its parameters are first prepared for energy and daylight performance evaluation (as in Test-case 3). This requires adding physical properties to the mass model, such as adding windows and assigning materials to all building components. In the pursuit of this approach, a Pareto optimization method is targeted using an MOEA tool for evaluating and retrieving efficient design solutions, fitness functions have to be determined, with fitness values. Once the optimization run starts, a random population of solutions are generated in the solution space, and the run of generations leads to fitter solutions which will eventually form the Pareto front.

3.2.3. Process 3: The SC-KM Method (Innovative)

The innovative process in this study is the process of developing a new method for clustering the design options' shapes, and the incorporation of the developed method into the generative system. The set of design alternatives retrieved from the generative process are used as the input for the shape clustering method. The clustering method required formulating a set of algorithms to perform the following functions:

i. Pair-wise comparison for finding geometric correspondences: This activity requires a comparison of all design alternatives to compare and analyze their geometric

correspondences. For all the design solutions (shapes), a pair-wise comparison is targeted in which the compared shapes are overlapped for all possible cases in order to find a "Shape Difference Score" for the pair. The algorithm utilizes the difference of shapes, instead of the shapes themselves. As a result, a "Shape Difference Score Matrix" is constructed, a cross-reference matrix of difference values for all the shapes compared. This method differs from the typical grid-based shape comparison explained in Section 2. In a typical grid-based shape description, the vectors/codes of the whole overlaid grids are considered, and calculation of shape difference is often performed using the Hamming distance approach. In our method, scanning the cells inside the shapes is performed within a pair-wise shape comparison, directed to achieve exhaustive search for the overlap case where the shape difference is minimum. This exhaustive search requires another algorithm for the optimum assignment of the non-overlapping cells so that the sum of their Euclidean distances is the minimum, which necessitated the incorporation of the Hungarian algorithm.

ii. K-Medoids clustering: The distance-based "Shape Difference Score Matrix" is used as an input to the K-Medoids algorithm that clusters the solutions into groups of similar shapes, and identifies a medoid, a representative solution for each cluster. The clustering algorithmic set was applied to multiple test samples of shapes as will be explained in Section 4.

3.3. Specify the Reasoning to Analyze the System (Identify the Objectives)

In designing the system, each function (process) was pursued to achieve its expected objective, based on collected knowledge from studying the literature, and based on

experimenting with each process separately. The objectives were determined upfront, to address and solve the research problems, defined as follows:

i. the capability of the system to initiate similar/different shapes, or a building mass model with customized parameters and constraints to create a wide range of shapes.
ii. a generative component should be pursued to allow for parametric change to occur to produce a set of design options.
iii. the capability of the developed clustering method to cluster the generated shapes of design options into clusters of similar shapes, and to find the representative shape of each cluster.
iv. the system has to be easily implanted by designers, with adaptability to different design problems.

The design of the system was changed and improved in the prototype development process, according to the issues faced, and to the modifications needed to assure that the system behaves correctly, as expected. To achieve the above objectives, the system's functionalities were decided to be the following:

i. a parametric modeling functionality, with different test-cases to allow for parametric modifications
ii. the parametric model should be changed in a generative manner to provide a rapid generation of design options, within possibly (optional) an evaluation-based search mechanism
iii. the selected solutions should be articulated and clustered according to their geometric differences. This required a cross-reference geometric difference finding amongst all the

compared solutions. The clustering method can incorporate and benefit from the K-Medoids algorithm to cluster the solutions into groups of similarities, and find the representative shape of each group.

3.4. Testing (Create the System Interface)

To communicate the system, the interface for the prototype has to be a programming-based platform because the prototype is comprised of several connected algorithms. Also, the platform has to be compatible with the determined functions of parametric modeling, design optimization, and clustering algorithms. Preferably, the platform can accept external input, store and display data, and capable of representing architectural geometric models. Therefore, a visual programming platform (Grasshopper/Rhino) was selected as the platform; however, other platforms such as (Dynamo/Revit) can be used alternatively.

The primary reason for favoring Grasshopper/Rhino is its advancement and compatibility with multiple plugins and tools such as the simulation tools (Honeybee and Ladybug), the multi-objective evolutionary algorithm (Octopus), and more importantly the programming tool (GH_CPython) that that allows for incorporation of numerical and scientific Python libraries in order to develop and perform the clustering and related algorithms. The resulting interface of the developed system is a framework of a written and customized set of algorithms and nodes inside the Grasshopper platform that can be adapted to other test-cases and design problems.

For testing the prototype, three experimental cases were conducted. In the first test-case an abstracted model, a grid-based layout that is capable of generating similar and different geometric shapes, was used to develop the shape clustering method. In the second carried out test-case, and upon inspecting published studies on generative systems, a decision was made to

apply the developed SC-KM method to the shapes of Rodrigues et al. (2017)'s study, for testing the clustering method and comparison. The third test-case was based on modeling a parametric spatial configuration that represents zones and spaces of the building and included the design optimization process.

3.5. Validation

Collectively, this study is rational, empiricist, and logical; thus validation was based on the results of the tests and evaluations done through comparing the processes of the prototype with corresponding studies (Sargent & Balci, 2017). As mentioned, the developed system was applied to three different test-cases to demonstrate and test the behavior and functionalities of its components, and its usefulness. For validation, verification processes were carried out internally and externally. Since the system framework was developed using two routine processes, and one innovative process, validating the routine tasks was simplified to verification and comparison of the results since those routine tasks are based on inherently validated components and methods (Phillips et al., 2002) such as parametric modeling, and the optimization method. The innovative method of clustering needed to be thoroughly validated. As such, overall, the validation procedures can be explained as follows:

a) For the parametric from generation process, verification tests were made by flexing the model, changing its parameter to assure the update of the other related parameters.
b) For design optimization, validation was pursued in testing the results of optimization through manual checking and calculating the resulting values.
c) In terms of validating the clustering method, two approaches were pursued: internally (qualitative), visual examination perceptual coherence of the clustering results, and

externally (quantitative), comparing the results to reference clusterings of an existing study and conducting clustering evaluation measure procedures.

In the three applications of the systems (the three test-cases explained in the next section); validation of the clustering method was performed as the following:

i. In the first test-case, the clustering results were evaluated perceptually and compared against predetermined reference clusters.
ii. For the second test-case, the clustering method was applied to a sample of shapes from the study of (Rodrigues et al., 2017) after modeling those shapes and using them as input to our clustering method. In addition, a comparison and evaluation study of the resulted clusterings was conducted using clustering evaluation techniques and measures.
iii. In the third test-case, validation was targeted to test the application of the clustering method into the optimization process. Also, the clustering results were visually analyzed according to their perceptual coherence and the existence of a dominant shape in each cluster subset.

3.6. Summary of the Section

The section describes the methods used to carry out this research. The development protocol of the proposed system followed a framework of computational model development principles suggested by Kunz (1989) and often followed in the computational model-based inquiry. The literature review offered in Section 2 revealed a gap in generative design research resembled in the lack of organizational strategies. In response, developing the new system was targeted to solve the problem and achieve the objectives of developing a new shape clustering method incorporated into the system. This necessitated a demonstration of a fully working

prototype with the determined functionalities. The system development had to culminate in an illustrated interface, a package of fully working algorithms to achieve the determined shape articulation tasks and the integration of the method into a generative system. In addition, testing the shape clustering method is necessary to verify that the components are working successfully as expected.

4. TEST-CASES AND VALIDATION[*]

Three test-cases were developed as applications to the developed prototype, to demonstrate its framework, explore its functionalities, and to test and evaluate those functionalities. For the first two test-cases, the processes of parametric modeling, and the clustering method were carried out with the absence of the design optimization process. In Test-case 3, design optimization was followed, achieved by using the MOEA tool, leading to a fully working prototype. In every test-case application, the following phases and tasks were conducted and detailed in the subsequent subsections: introduction to the experiment, developing the test-case experiment, retrieving the experimental outcome, evaluating of the experimental outcome, and conducting the validation procedures.

The platform used for all the test-cases was Grasshopper/Rhino, a visual algorithmic environment, often used for parametric exploration and is compatible with other plugins and tools. For parametric modeling, in addition to the available functions and nodes inside Grasshopper, there was a need to write customized algorithms in some test-cases such as for creating the grid-based layout in the first test-case. In developing the SC-KM, the programming tool (GH_CPython) which is a plugin that allows for using modules and scientific Python libraries to be incorporated into Grasshopper. In addition, the typical Iron-Python language was used to write other programs, and sometimes C# language was used for specific algorithms. In the process of generative exploration and performance evaluation in Test-case 3, the energy and

[*] Part of this section is reprinted with permission from "Incorporating Form Diversity into Architectural Design Optimization" by Yousif, S., Yan, W., & Culp, C. (2017). Paper presented at the ACADIA 2017: DISCIPLINES & DISRUPTION [Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), MIT, Cambridge, MA.

daylight simulation tools (Honeybee and Ladybug), and the MOEA-based Octopus tool were used. Octopus is a multi-objective evolutionary algorithm that provides efficient design exploration to achieve high-performance designs. A more detailed explanation of the algorithmic tools and methods used is offered in the following subsections with elaboration on each test-case pursued.

The specifications of the PC used for the three test-cases are as follows: The Processor is Intel® Core™ i7 8086K (6-Core/12-Thread, 12MB Cache), the Operating System is Windows 10 Pro 64-bit English, and the Video Card is Dual NVIDIA® GeForce® GTX 1080 Ti graphics with 11GB GDDR5X each.

4.1. Test-case Experiment 1

In this experiment, the prototype was applied to a grid-based test-case, with the focus on two main processes: (1) the generation of a large dataset of shapes with similarities and differences, and (2) the development and testing of the clustering method with its algorithms and components. Preceding to this test-case, multiple experiments were carried out that include the process of generative exploration and performance evaluation in addition to the early experiment of creating a form diversity articulation algorithm, as explained in the published paper (Yousif et al., 2017). This test-case was carried out in the following phases: programming a parametric setup of a grid-based layout to generate a dataset of shapes, developing a clustering algorithmic definition, testing and evaluating the clustering results, and conducting validation procedures. The following sub-sections describe these tasks in detail.

4.1.1. Introduction to the Test-case Experiment

This experiment was designed to apply the prototype to a test-case in which, the two processes of generative parametric modeling and development of the clustering method were

carried out and tested. Throughout its development, the major challenges faced were the functionalities of the developed clustering method, and its credibility (validity). Therefore, in explaining this test-case, the focus is on the newly developed clustering method and its algorithms. The processes of developing test-case 1 can be illustrated in (Figure 4-1) and explained in the following subsections.

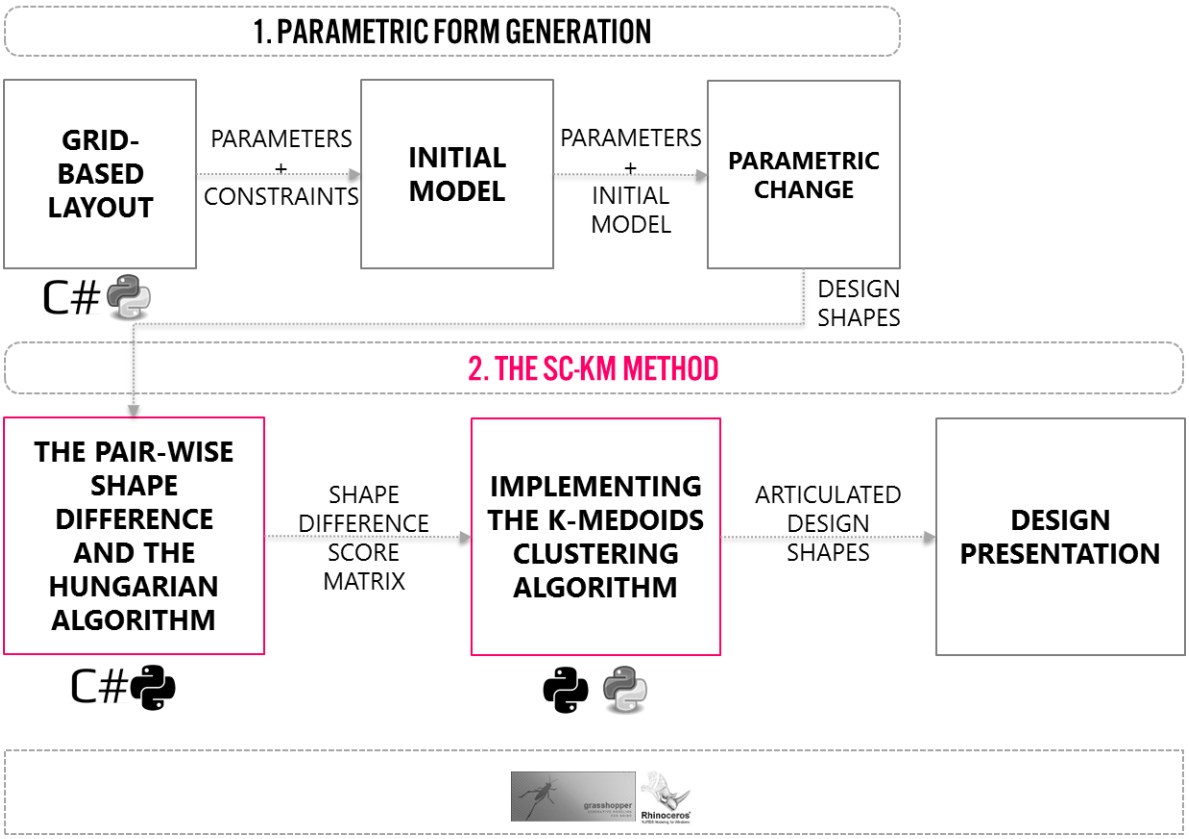

Figure 4-1. Workflow and tools of Test-case 1 with the SC-KM method as a major component.

4.1.2. Parametric Form Generation

The set of shapes, in this case, was created using a generative algorithmic layout, a grid-based setup. This algorithmic layout combines the use of available nodes in Grasshopper for parametric modeling and specifically formulated algorithms. The layout setup is based on a configuration of 10 modular cells, each represents a building space, allocated to a 2D grid

composition. The cells can be allocated flexibly, allowing for modification possibility in the solid/void relation, where solid is the existence of a cell and the void is its absence. The 2D grid contains 11 x 11 units, with each measuring 10 x 10 m. Each allocated cell is 10 m high, represents an open volume that can contain multiple floors. Inside the 121 unit-grid, only 10 cells are parametrized and allocated on the grid, leading to 121! / (121-10)! or (4.5913309e+20) of possible configurations to be generated. This large number of design alternatives requires an aggregation mechanism, in order for users to compare the generated designs, and this is the reason for conducting this study as described in the introduction section. The allocation of cells was done using an algorithm, based on an agglomerative combinatory, a cell-by-cell approach. The first cell (Cell 1), was located in the grid center (x=6, y=6), and the other 9 cells were appended subsequently (Yousif et al., 2017). This cell-allocation method is illustrated in (Figure 4-2).

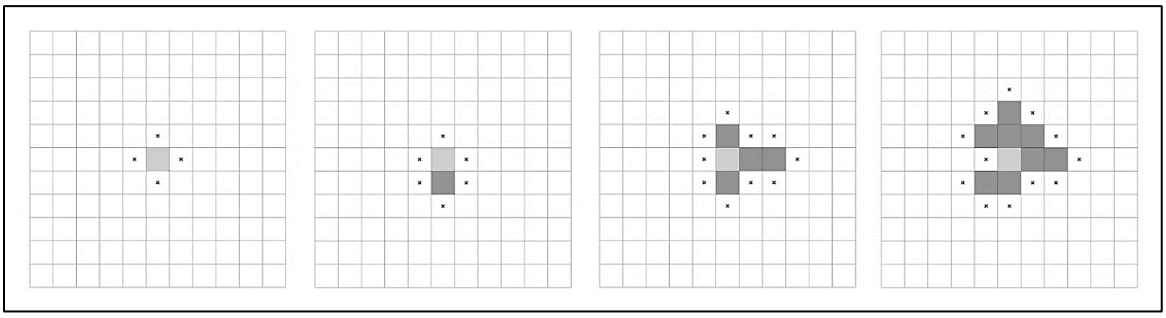

Figure 4-2. From left to right, the process of generating the mass model based on adjacency, non-overlap, and boundary-detection constraints. The dots represent feasible locations for the next appended unit.

Adding each cell was parametrically-driven, through coding customized algorithmic definitions to limit the possible configurations to three constraints:

a) Adjacency constraint: each cell must be adjacent to at least one side of the previous cell(s)

b) Non-overlap constraint: a cell cannot be located on an occupied cell

c) Boundary-detection constraint: the added cell cannot be appended (or moved when changing parameters) outside the grid boundary.

Changing the 9 location parameters except for the first cell (each parameter controls the changing location of one cell), the adjacency relations among those units change, based on repositioning, which changes the shape configuration. Examples of the shapes resulting from this grid-based pattern are shown in (Figure 4-3). The rationale for using this pattern was due to its simplicity yet capability to allow possibly for a diverse/similar set of shapes, and its appropriateness to test the clustering algorithm (Yousif & Yan, 2018), as will be explained later in this test-case. The grid-based pattern can also approximate more complex forms characterized by curvatures or angles, using a high-resolution grid. In addition, the application of the clustering algorithmic definition to this grid-based descriptor will be feasible as the clustering will consider shape difference based on the center points of the grid cells. The layout is invariant to scaling with all shapes of the same area, yet variant to rotation and reflection.

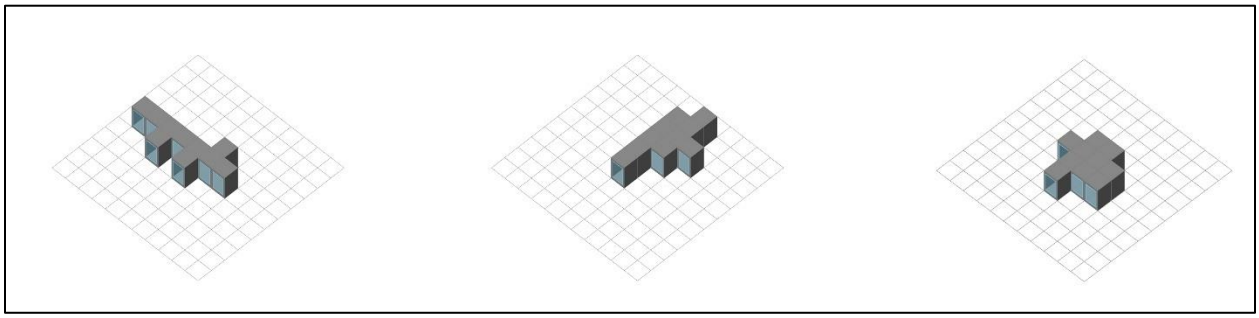

Figure 4-3. Examples of shapes generated from the grid-based pattern.

4.1.3. Developing the Shape Clustering Method

An algorithmic set was developed to achieve shape clustering. This clustering set represents the complete version of the algorithms that perform shape clustering, with the implementation of the K-Medoids and other algorithms, applied to a set of shapes generated in

this case through parametric changes. For formulating the clustering algorithms, applying and testing the clustering method, a sample of shapes was needed. In this case, the grid-based layout was changed manually and purposefully to generate three perceptually identified clusters. This was done to produce twenty shapes comprised of three groups; each group contains shapes with only one cell-location changed within the group, as illustrated in (Figure 4-4), a top-view of the sample of shapes. In this sample set, one group of seven vertically linear shapes, another of seven horizontally linear shapes, and the third group of six zig-zag (meandering) shapes were created. The shapes were intentionally shuffled and randomized for verification purposes (Yousif & Yan, 2019).

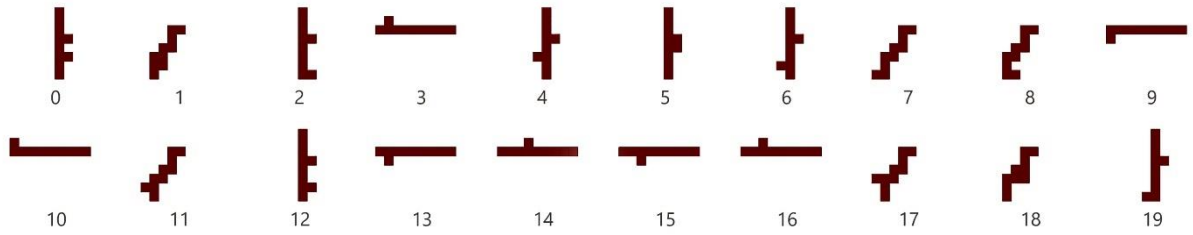

Figure 4-4. A sample of shapes used to apply and test the clustering method.

The clustering method was developed using two main algorithms: The Pair-wise Shape Difference and the Hungarian Algorithm, and the K-Medoids Algorithm, as explained in the following sub-sections.

4.1.3.1. The Pair-wise Shape Difference and the Hungarian Algorithm

To explain the importance of pair-wise comparison, it is essential to note that in formulating this algorithm, the objective was to compare all the sample shapes in an agglomerative pair-wise method, pair by pair. In each pair-comparison, the pair is overlapped, moving, for example, each cell of Shape 0 over all the ten cells of Shape 1 of (Figure 4-4), leading to hundred cases of

overlap, of which 4 cases are shown in (Figure 4-5). The rationale here is that the optimum pair overlap should be considered out of all the possible overlaps, for retrieving the actual geometric difference between the two shapes. This assumption is based on operations often applied to shape comparison studies, where certain procedures are applied to the compared shapes before determining their differences. In such shape comparison analysis, there is a need to align the compared shapes, using transformation operations such as scaling, translation, to best find their similarities and differences (Funkhouser et al., 2005). We utilized the alignment approach for overlapping the shapes to compare them. Readily, all the shapes used in this test-case are of similar area and did not require a scaling procedure. In terms of rotation and reflection, the decision made here was that the pair-wise comparison is variant to rotated and reflected shapes, for example, a T-Shape and a rotated T-Shape represent two different shapes.

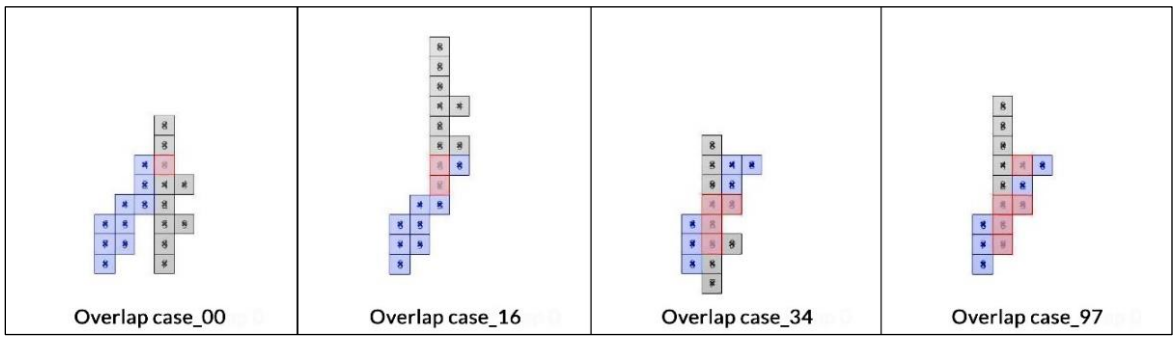

Figure 4-5. Four selected of the 100 overlap cases of the compared pair (Shape 0 in grey, Shape 1 in blue, and the overlapping cells are in red).

Using the GH_CPython plugin, with its capabilities to use Python scientific libraries, the algorithm was developed. In each overlap case, the non-overlapping cells between the two shapes are subjected to a Euclidean distance measure, considering their center-points. The reason for the 100 cases of translation (overlap) is to check all possible shape overlap cases and to find the smallest sum of Euclidean distances between cells in the pair. This exhaustive search for the best overlap leads to a problem of cell assignment. The optimum assignment is to determine

what non-overlapping cells of shape 0 that would be best assigned to the non-overlapping cells of shape 1 for a minimum sum of distances, for which we will utilize the Hungarian algorithm to resolve.

The GH_CPython-based algorithmic node (Pairwise Shape Difference and the Hungarian Algorithm), represented in (Figure 4-6), was formulated using three parts. First, preliminary functions were created for defining: a vector for translation (move), coordinates checking, and Euclidean distance calculation. The second part of the written algorithm is the function: “Shape Difference Calculation for a Pair of Shapes” for calculating the distance-based difference for a pair of shapes. The third component is the “Get the Shape Difference Score Matrix” function to retrieve the “Shape Difference Score Matrix”, to be used for the K-Medoids algorithm.

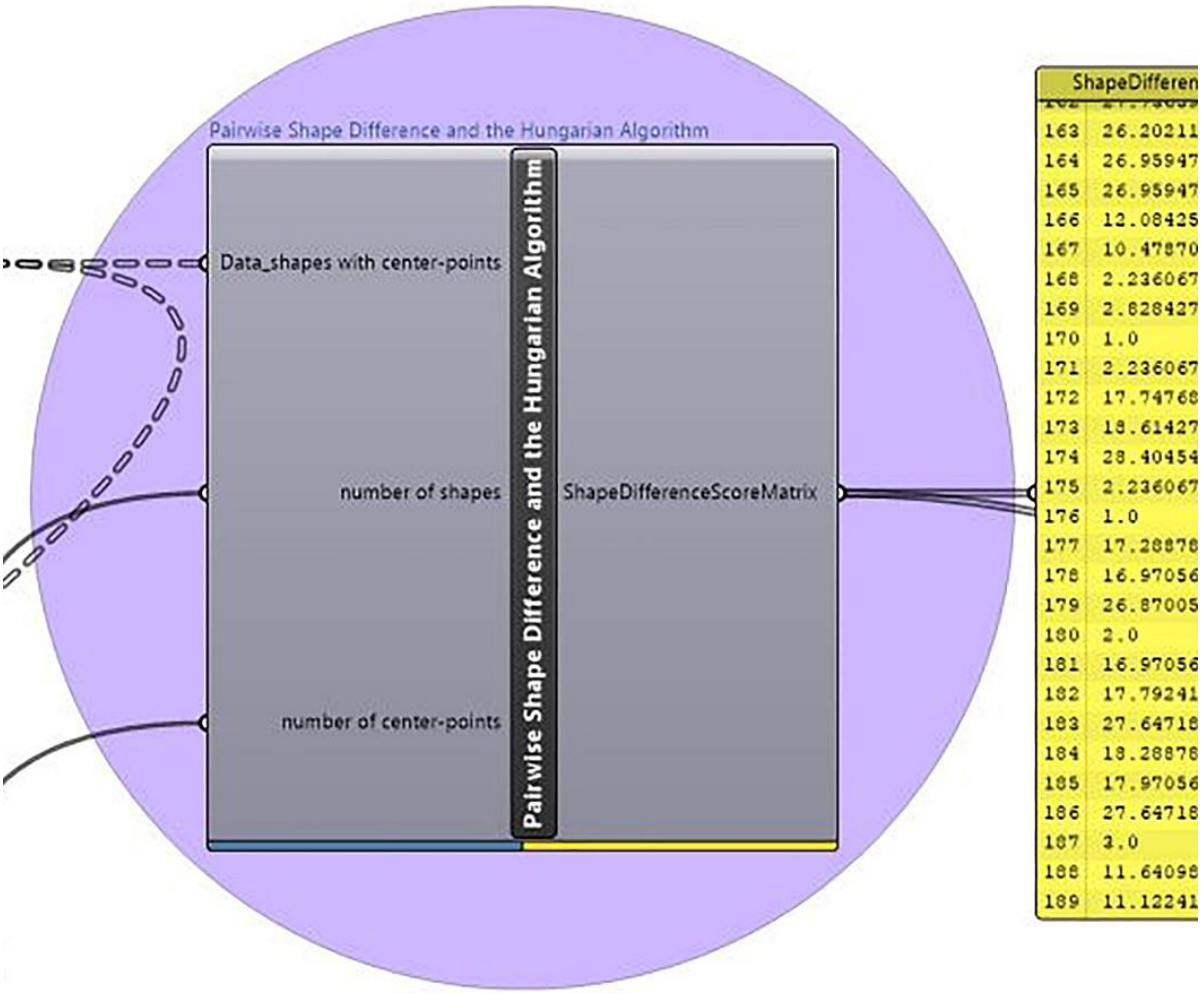

Figure 4-6. The GH_CPython-based Grasshopper node of the Pair-wise Shape Difference algorithm, with its input and output.

Those three parts can be explained as follows:

I. Preliminary Functions:

The three following functions are called in the two main successive functions.

- Function A: Defines a translation (move function) that moves a shape to overlap with another shape.
- Function B: Defines a checking function that checks if two shapes are on the same coordinates (x, y, z).

- Function C: Defines a distance function that performs a Euclidean distance calculation between two points.

II. Function 1: Shape Difference Calculation for a Pair of Shapes

As an input to this function, the main important input variable is a data tree (a list of lists) that represents the list of shapes, each with its ten cells' center-points. In this function, a "Shape Difference Score" is defined as a pair-wise shape comparison result, which is, across all overlapping cases, the smallest sum of distances for the best matching non-overlapping cells. In every overlap case, to achieve the best matching of the non-overlapping cells, the Hungarian algorithm was implemented to compute the optimum assignment of matching cells between each pair of shapes so that the sum of Euclidian distances between the non-overlapping cells of the pair is minimized. As an example, in Overlap Case 97 in (Figure 4-7), the optimum assignment of the non-overlapping cells (1-5) of Shape 0, to the non-overlapping cells (a-e) of Shape 1, which has been shaded in (Table 4-1), was needed.

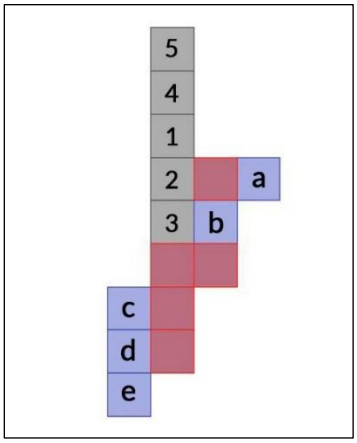

Figure 4-7. Overlap case 97 of the pair-wise comparison of two overlapping sample shapes (Shape 0 in grey and Shape 1 in blue), with five cells overlapping (in red).

Table 4-1. Matrix for distance calculations for center-points' coordinates (a-e), and (1-5). Border-outlined boxes are for a random assignment and shaded boxes for the optimum assignment by the Hungarian algorithm.

		a	b	c	d	e
		(5, 5)	**(4, 4)**	**(2, 2)**	**(2, 1)**	**(2, 0)**
1	**(3, 6)**	2.24	2.24	4.12	5.10	6.08
2	**(3, 5)**	2.00	1.41	3.16	4.12	5.10
3	**(3, 4)**	2.24	1.00	2.24	3.16	4.12
4	**(3, 7)**	2.83	3.16	5.10	6.08	7.07
5	**(3, 8)**	3.61	4.12	6.08	7.07	8.06

In Overlap case 97, depicted in (Figure 4-7), an assignment makes a one-to-one match between the non-overlapping cells of shape 0 and shape 1. Thus, the number of all possible assignments is 5! (factorial of 5), from which we need to find the best assignment. For instance, to understand the need of the best assignment, we have the following two possible assignments: one is random, which results in the sum of Euclidean distances between the 5 pairs of cells (center points) to be 21.91, while the optimum assignment selected by the Hungarian algorithm is 19.13, the minimum sum of distances possible. (Note that the 19.13 may or may not be the overall minimum for all 100 overlapping cases.)

$$\boldsymbol{sum\ of\ distances\ (random\ assignment)} = (\boldsymbol{a},\mathbf{3}) + (\boldsymbol{b},\mathbf{2}) + (\boldsymbol{c},\mathbf{1}) + (\boldsymbol{d},\mathbf{4}) + (\boldsymbol{e},\mathbf{5}) = \mathbf{21.91}$$

$$\boldsymbol{sum\ of\ distances\ (optimum\ assignment)} = (\boldsymbol{a},\mathbf{5}) + (\boldsymbol{b},\mathbf{4}) + (\boldsymbol{c},\mathbf{1}) + (\boldsymbol{d},\mathbf{2}) + (\boldsymbol{e},\mathbf{3}) = \mathbf{19.13}$$

After finding the minimum sum of distances in each overlap case, 100 shape difference values are retrieved of which the minimum value is selected to represent the actual difference between the pair. The algorithm for calculating the "Shape Difference Score" of the two shapes can be expressed as follows:

```
Shape Difference Score = min ((min sum of distances of overlap case 1),
                              (min sum of distances of overlap case 2),
                               ……
                               )
```

As a result, across all the 100 overlap cases of the pair, the minimum sum of distances is 9.30 for Overlap cases: 34, 41, 52, and 95 (identical overlaps). It is important to note that a repetition of overlap was expected (but they could be eliminated to make the algorithm more efficient).

III. Function 2: Get the Shape Difference Score Matrix

For all the paired shapes, the algorithm constructs a "Shape Difference Score Matrix": a cross-reference matrix of values that represents the Shape Difference between each two compared shapes of the twenty sample shapes illustrated in (Table 4-2). In (Figure 4-6), the "Shape Difference Score Matrix" (as a list of lists in Grasshopper) is illustrated as an output of the (Pair-wise Shape Difference and the Hungarian Algorithm) node.

Table 4-2. A triangle of the symmetric "Shape Difference Score Matrix" of Shapes 0-19 in the explained sample.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	19
0		9.30	2.00	27.14	2.00	1.00	2.24	10.71	8.89	27.91	27.91	11.12	1.00	27.14	25.55	26.37	26.37	11.54	9.48	2.83
1			10.67	20.38	7.89	9.06	8.71	2.24	1.00	20.72	21.38	2.00	9.89	19.94	18.38	19.16	19.38	2.24	1.41	9.71
2				28.39	2.83	3.00	2.24	11.54	9.89	29.17	29.16	12.08	1.00	28.39	26.87	27.62	27.62	12.72	11.48	2.00
3					27.10	26.51	27.74	20.52	21.21	2.24	1.00	19.88	27.74	2.00	2.00	2.24	1.00	19.16	18.97	28.40
4						2.24	1.00	8.89	7.06	27.88	27.87	9.30	2.24	27.10	25.55	26.32	26.32	9.89	9.30	2.00
5							2.83	10.47	8.65	27.28	27.28	10.89	2.00	26.51	24.91	25.73	25.73	11.30	8.48	3.61
6								9.30	7.89	28.51	28.51	9.89	2.00	27.74	26.21	26.96	26.96	10.67	10.12	1.00
7									2.00	19.80	21.33	1.00	11.12	19.16	18.52	18.52	19.52	2.00	3.61	9.89
8										21.33	22.21	2.24	9.30	20.56	19.21	19.80	20.21	2.83	2.24	8.89
9											2.00	19.16	28.51	1.00	3.61	2.00	2.83	18.52	19.35	29.15
10												20.56	28.51	2.24	3.00	2.83	2.00	19.80	19.97	29.16
11													11.54	18.38	17.88	17.75	18.88	1.00	3.16	10.71
12														27.74	26.20	26.96	26.96	12.08	10.48	2.24
13															2.83	1.00	2.24	17.75	18.61	28.40
14																2.24	1.00	17.29	16.97	26.87
15																	2.00	16.97	17.79	27.65
16																		18.29	17.97	27.65
17																			3.00	11.64
18																				11.12
19																				

4.1.3.2. Implementing the K-Medoids Algorithm

The next task was to implement the K-Medoids algorithm, with additional modifications to apply it to the sample shapes, using their "Shape Difference Score Matrix". Conceptually speaking, in K-Medoids clustering, the initial medoids or representative objects are selected randomly, then, iterations of replacing the medoid with other (non-medoid) objects occurs to improve the clustering (Han et al., 2011). Importantly, all the possible replacements are tested in an iterative process that continues until the clustering accuracy cannot be further improved through replacement; the accuracy or quality of clustering is often calculated as a cost function of the average difference between an object and its cluster representative (Han et al., 2011).

To explain its functionality in details, a typical K-Medoids algorithm performs the Partitioning Around Medoids (PAM) method (Velmurugan & Santhanam, 2010) as follows:

- **Input:**
  K: The number of clusters
  D: The matrix of data containing n objects (shapes)
- **Method:**
  Random initiation of cluster medoids: first, the algorithm arbitrarily selects medoids in D as the initial representative objects. Next, the algorithm assigns each remaining object to the cluster with its nearest medoid.

  Repeat, update of cluster medoids:
  While the total cost (sum of distances of points to their medoid) of

  the assignment decreases:

  For each medoid *m*, for each non-medoid data point *o*:

  1. Swap *m* and *o*, assign each data point to the closest medoid, recompute the cost
  2. If the total cost increased, undo the swap

- **Output:** A set of k clusters that minimizes the total cost.

The specific K-Medoids implementation is a NumPy/SciPy function. In our implementation of the algorithm, it called the function inside the GH_CPython coding tool,

which enables the importation of NumPy/SciPy. The reason for selecting this implementation of K-Medoids is due to its validated method and robustness to applications, designed particularly to consider distance matrices and cluster them based on user-defined difference values (Bauckhage, 2015).

According to Bauckhage (2015), the distance matrix-based K-medoids clustering method solely relies on distances between data points (or shapes), therefore, the distance matrix becomes the most important component of the algorithm. The "Shape Difference Score Matrix" was already retrieved from the shape difference algorithm described in the previous section. The input data of the K-Medoids becomes the following: the number of shapes, the "Shape Difference Score Matrix" (as a list of lists), the number of clusters, and the number of iterations of the K-Medoids as illustrated in (Figure 4-8).

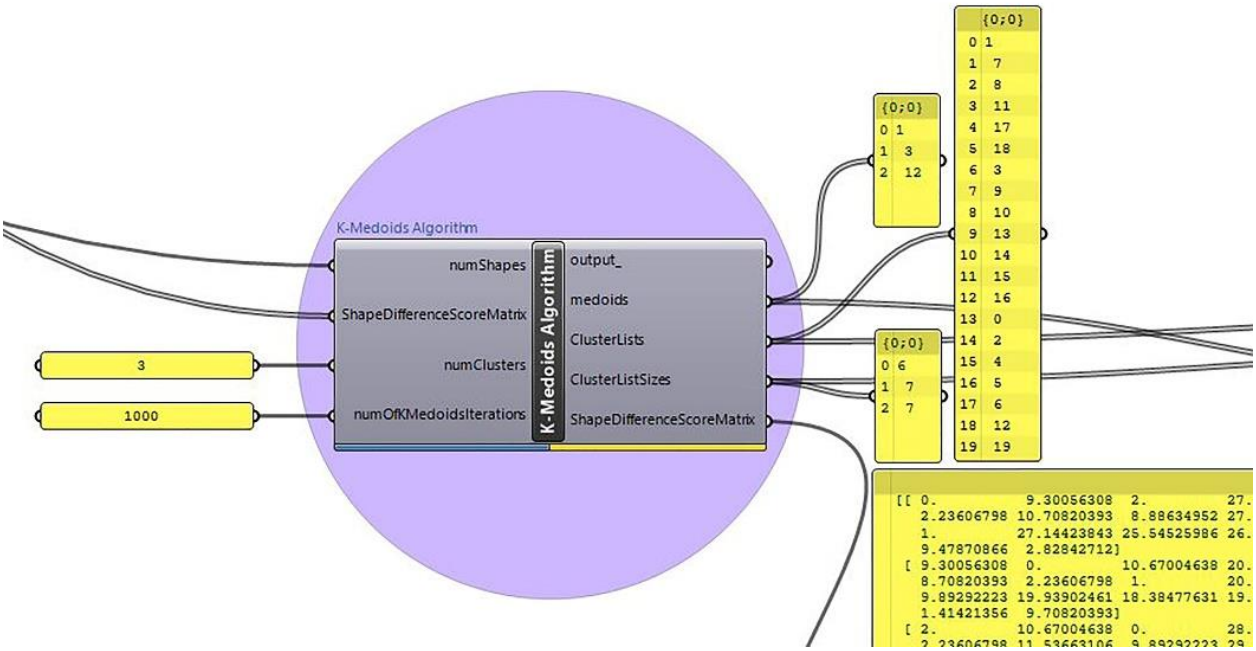

Figure 4-8. The Grasshopper node of the K-Medoids Algorithm with its input and output.

The outputs of the K-Medoids are primarily: the list of clusters of shapes and the medoids. The outputs have been articulated as lists of clusters (each cluster with their shape IDs), the medoids' indices (IDs), and the cluster size (number of shapes in each cluster) as depicted in (Figure 4-8).

4.1.4. Experimental Clustering Outcome

Applying the described clustering algorithmic set to the above-illustrated sample of twenty shapes, the results showed correct clustering of shapes with the given numbers of clusters (3) and identified the medoid of each cluster (Figure 4-9). Further, more scenarios were tested and the results were perceived accurate too (Figure 4-10). To analyze the outcome of this clustering method, it is important to note that the sample cases used were intentionally created in advance to have three clusters each by manually typifying shapes from the grid-based layout to create three groups of perceptually similar shapes. Perceptual refers here to the visual identification of shape similarity/difference. Therefore, the samples were already clustered as reference clusters. For instance, in (Figure 4-9), the selected sample of shapes includes a group of seven horizontally linear shapes (in cyan color) with only one cell changing in location, making the change of shape minimal. The method resulted in accurately identified clusters. Despite that in the bottom of (Figure 4-10), the shapes become more complex to differentiate, the clusters were correctly identified by the clustering method.

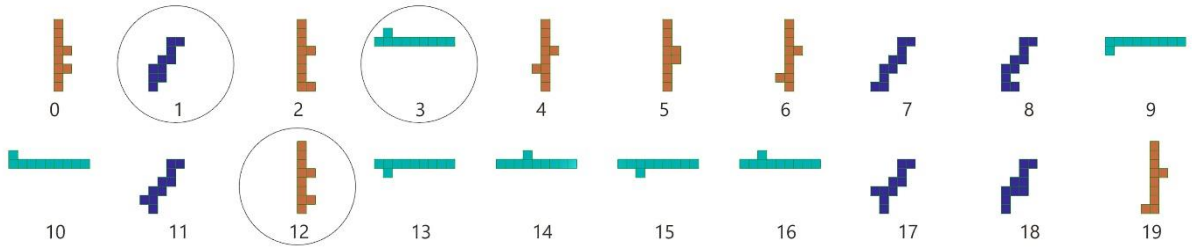

Figure 4-9. The output of the developed K-Medoids-based shape clustering; three groups of shapes are automatically clustered and represented in three colors, with the Medoids in a darker tone for each group.

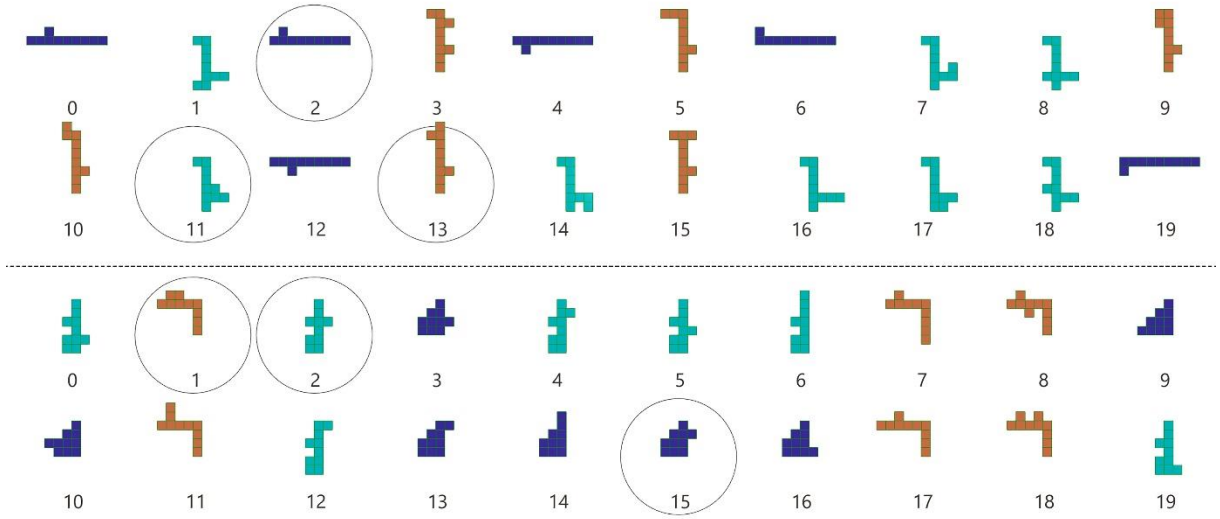

Figure 4-10. The correct clustering results: the same colored shapes indicate that they are in the same cluster in two additional test cases (above and below), with the Medoids circled, and in a darker shade for each group.

4.1.5. Validation Study

For verification purposes, before clustering each sample, the shapes were shuffled randomly in location to make sure the algorithmic set worked regardless of the order of shapes in terms of locations. In addition to the presented samples, more other tests were conducted, applying the algorithmic set to multiple samples of two or three clusters, and the algorithm identified the clusters as expected.

As a measure to determine the accuracy of the clusterings, the results were compared to the reference clustering; the algorithmic set is expected to cluster the similar typified shapes in

one cluster, given the same number of clusters determined in the reference clustering such as 3 in the illustrated samples. Also, perceptual coherence was considered a measure for clustering accuracy; a cluster is considered coherent if it presents a dominant shape that appears at a high number of times in the cluster (Rodrigues et al., 2017). In the results of the explained method, for all the tested samples, the results showed a successful outcome of the clustering method to determine dominant shapes in a cluster, and correct clusters compared to the intended reference clusters. However, the next validation procedure was thought to be external, comparing the results of clustering to other studies, and this was implemented in the second test-case, explained in 4.2.

4.1.6. Discussion and Conclusions

The development of the shape clustering method was demonstrated in this experiment through formulating two functionalities: (1) using grid-based shape description and conducting pair-wise comparison for computing a shape difference score, and (2) implementation of the K-Medoids clustering. The test-case showed how the method can be applied to cluster samples of grid-based shapes and how it is applicable to other samples, through changing the input variables. The resulting clusters showed successful articulation of the shapes, retrieving the similar clusters and representing each cluster with its medoid. It is important to note here that the sample shapes do not necessarily represent building shapes, yet, the sample sets are grid-based abstract shapes with low resolution (only 10 cells), created for the purpose of developing and testing the method. Therefore, in the next test-cases, other shape samples are utilized.

One of the important discussion points for the K-Medoids clustering algorithm used is its characteristic of randomness, embedded in the algorithm to initiate the medoids at the first step. However, in running the algorithm multiple times using the above samples, the results were

always consistent and converged to the same results. This random initiation can be impacted by the number of shapes. For a small sample of shapes, and a high number of iterations, it is expected that the results will converge to the optimum assignment of items to their medoids. This was proved in this test-case, since for all the tested samples, running the clustering algorithm multiple times, the same clusterings were retrieved. In particular, as a number of iterations for the K-Medoids algorithm in this experiment, 1000 was used.

4.2. Test-case Experiment 2

To test the prototype, a new experiment was conducted in which the SC-KM was further developed. The objectives of pursuing this experiment were threefold: (1) to apply the prototype to a new sample of shapes, architectural building layouts, (2) to add an algorithm to convert the boundary-based building shapes into grid-based in order to apply the clustering method, and (3) to apply clustering evaluation metrics to test the results of the clustering method in comparison with the results of an existing study for external validation. The workflow, processes, and tools of this experiment are illustrated in (Figure 4-11).

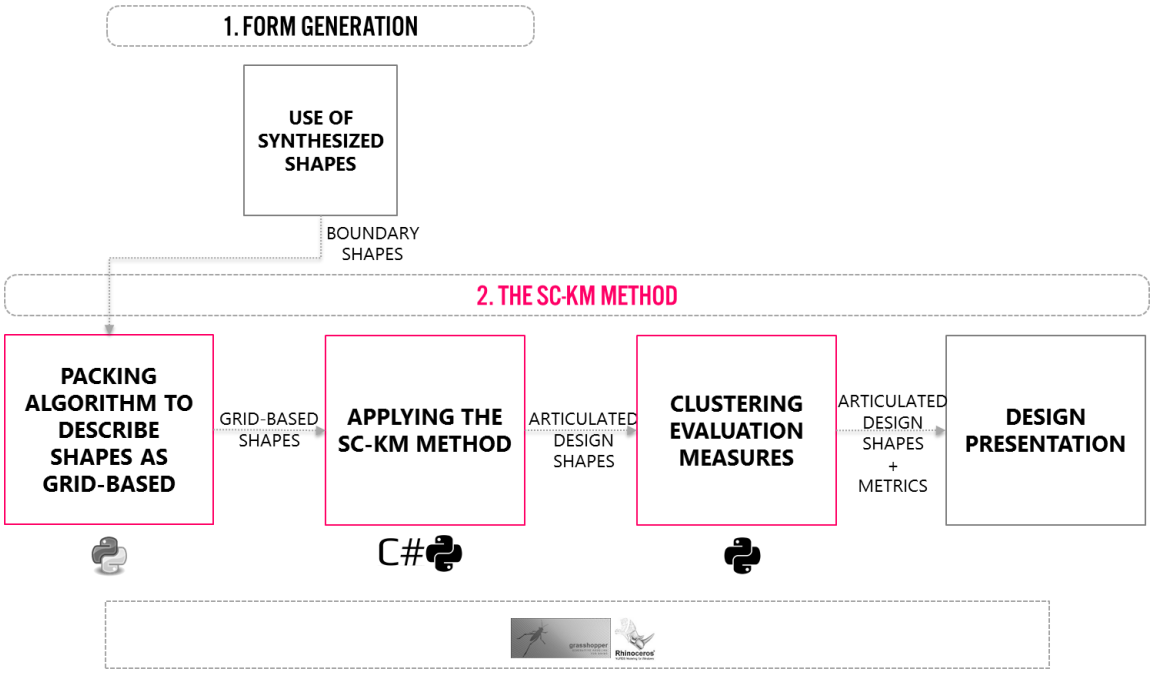

Figure 4-11. The workflow of Test-case 2 with the incorporation of the packing algorithm and the clustering evaluation metrics.

4.2.1. Introduction to the Test-case Experiment

In this test-case, a set of 72 shapes were remodeled based on the shapes used in the study of (Rodrigues et al., 2017). The shapes represent designs of architectural floor plans of a three-bedroom single-family house, generated through a hybrid computation scheme combining an Evolutionary Program for Space Allocation Program (EPSAP) along with a Stochastic Hill Climbing approach in a two-stage method (Rodrigues et al., 2013). The EPSAP produced design alternatives of multi-story floor plans based on parametric non-rigid, and non-fixed circulation elements that emerge throughout the search interacting with the other spaces (Rodrigues et al., 2013). This generative design process initializes a random population of arbitrarily arranged rectangles of which each rectangle represents space, and the search proceeds with the evaluation of each design options against defined objectives. Those objectives include connectivity amongst the spaces, the proximity of those spaces, spatial dimensions and area for minimum side and

minimum floor footage, compactness of the floor arrangement, space's overflow in regards to the boundary, and fenestration dimensions and their orientation (Rodrigues et al., 2013). The search mechanism produced a large set of alternative floor plans, and using certain requirements, the EPSAP algorithm generated 72 design alternatives that represent the improved fitness values to satisfy the objectives out of 576 design candidates (Rodrigues et al., 2017). Next, the resulted set of 72 design alternatives was chosen as the sample for shape representation and clustering studies.

In preparing for clustering the generated set, (Rodrigues et al., 2017) have conducted an online survey asking design experts to group the 72-floor plans, considering the main design features such as shape, and interior configuration arrangement (Sousa-Rodrigues et al., 2015). Analyzing the survey responses, a conclusion was drawn that human experts are inconsistent in clustering the sample, sometimes grouping them according to outline shapes, and other times by their interior configurations. For example, one clustering results in grouping dissimilar floor plans, such as grouping floor plan A' (the first raw of rectangle shapes in Figure 4-12) in the same group as floor plan B' (the first raw of L-Shapes). Also, the subjects in the study considered floor plan B' as a design of the same interior arrangement of floor plan C', leading to grouping A', B', and C' together. For that reason, the results of the survey were dismissed and considered inaccurate; alternatively, a reference clustering was utilized, determined by typifying and labeling the 72 designs (from A', first raw to I', last raw) as depicted in (Figure 4-12). This typifying approach was done by the authors, considering the shapes' characteristics and their interior arrangements of spaces (Rodrigues et al., 2017). For comparison purposes, in our use of the 72 shapes, the reference clustering was used for evaluating our clustering results, following (Rodrigues et al.)'s use of this clustering set as a reference.

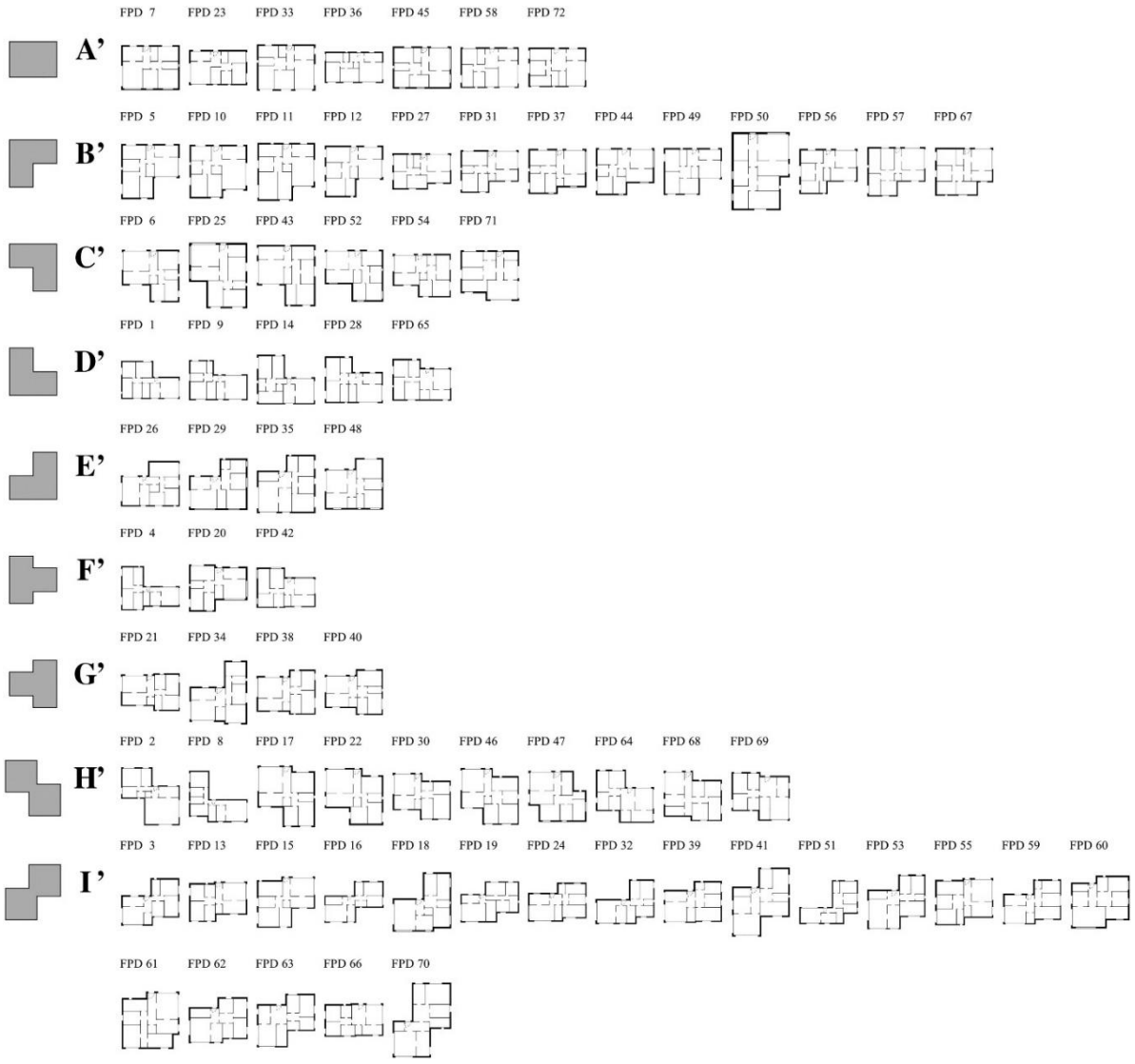

Figure 4-12. The 72 shapes used for test-case 2 as a reference clustering. Reprinted from Automation in Construction, 80, Rodrigues et al., Clustering of Architectural Floor Plans: A Comparison of Shape Representations, Pages 48-65, Copyright 4585091348148, with permission from Elsevier.

The method used for describing and clustering this synthesized dataset of shapes is invariant to scaling, yet variant to rotation and reflection. Examining the 72 shapes, it can be noticed that four cluster subsets belong to the L-Shape family, two clusters belong to the T-Shape, and two clusters are of the Z-Shapes. One can argue that the rotated and reflected L-Shapes, for instance, are in fact one big L-Shape family and need to be clustered all together. This may apply to the other rotated/reflected, similar or/and identical shape families. However,

the choice to have a variance to reflection and rotation in this set of shapes has been rationalized by (Rodrigues et al., 2017, p. 50) as follows:

> *"…Despite floor plans being generated on a blank canvas, human experts continue to have a notion of north-south and east-west framework, thus a rotated or a reflected floor plan is considered as an alternative design. Buildings have a strong relation with their environment and their form depends on the surrounding buildings, landscape, solar orientation, and so on. However, because there are no visual references around each floor plan, translation does not affect the human perception of that shape. As a result, rotation and reflection were considered as features that influence the clustering result."*

4.2.2. Modeling the Synthetic Dataset of Shapes

Modeling the 72 shapes in Grasshopper/Rhino was done for converting the shapes into a grid-based dataset for the purpose of applying our clustering method as will be explained in the following section. The conversion was performed considering the outline shape characteristics similar to (Rodrigues et al.)'s grid-based description. In modeling, the 72 shapes were abstracted to mass models and needed to be prepared for the clustering method. The clustering method described in Test-case 1 considers shape difference as the calculation of the distances between geometric cells or spaces' center-points, in a pair-wise manner. Thus, the method requires a grid-based shape descriptor, with an equal number of cells' center-points across all shapes to be compared in order for the clustering algorithmic set to function properly. This has led to a need for an approach to pack cells inside the 72 shapes to convert them to grid-based. The packing technique is explained in the next subsection.

4.2.2.1. The Bin-packing Algorithm

In packing problems, often, for a defined set of numbers and a determined bin capacity, the question is to assign each number to a bin so that the sum of numbers for each bin does

exceed the bin capacity (Korf, 2002). The solutions to bin-packing problems abound. The optimum solution to bin-packing issues is to use the least possible number of bins; one of the best existing bin-packing algorithms is the one of (Martello & Toth, 1990a, 1990b) in which a method is proposed to pack a set of different-sized items into a minimum number of identical bins (Korf, 2002).

Different from the typical bin-packing algorithm, in our packing technique, the items to be packed are unified in size, and the bins (the shapes to be packed) are variant in size and characteristics. Therefore, a modified packing algorithm has been developed for the purpose of packing the 72 shapes. This algorithm was formulated by the author utilizing Grasshopper/Rhino nodes, in addition to writing a program in Python to perform the packing. The algorithm consists of the following functions:

i. **Scaling and translation operation:** since each of the 72 shapes has a different area, all shapes were scaled to the same area, utilizing a formula to assure that the scale ratio is adaptive and leads to the same number of area units. The number of packed area units (number of cells) can be changed parametrically to allow for a range of units to be packed. Also, each shape was surrounded by a bounding box and translated (moved) so that the upper left corner is at the origin point (x=0, y=0).

ii. **Packing with an array of cells:** a simple Python code was written to create an array of units (each unit is 1x1). Next, a check-node was used to separate the cells' center-points that are inside the shapes (in red), and those outside the shape (in green) as depicted in (Figure 4-13).

iii. **Normalizing the list of contained cells' center-points:** The list of center-points that are contained in the shape was retrieved and subjected to analysis to check their numbers. All shapes had to be packed with the same number of cells, which was not always the case. Therefore, the resulting number of packed cells was evaluated against the predefined number of units to be packed, and if the contained cells are more than the defined number, the excessive cells will be deleted. When the packed cell number was smaller, the difference will be added to the list of cells. The decision for reducing the excessive cells was made in deleting the inner cells, while the addition was made by duplicating the last cells and moving them inside the shape, in a vector of (x = -0.5, y=0), without affecting the shape outline.

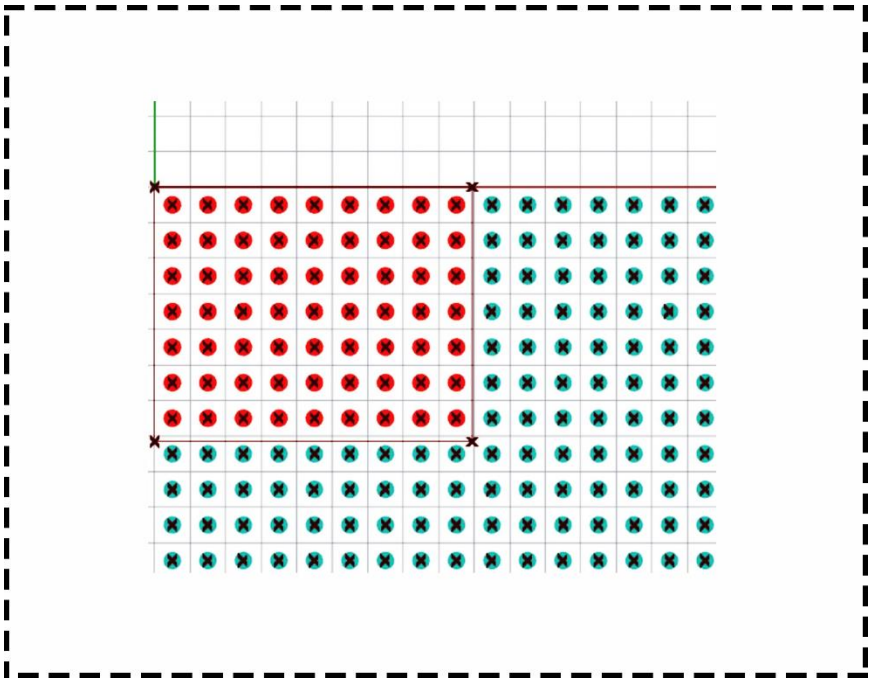

Figure 4-13. A visualization of a packed rectangle-shape A'-0 with contained cells' center-points in red, and external cells in green.

The 72 shapes were first packed with 36 cells in the first scenario, and for comparison purposes, another packing procedure, Scenario 2, was performed using 64 cell-packing, as illustrated in (Figure 4-14). The reason for increasing the number of packed cells is to test the impact of two scenarios on the clustering results. Yet, it is not expected that the higher resolution (the higher number of packed cells), would necessarily lead to better clustering performance, in comparison with the reference set. That is because the reference set was not clustered algorithmically, rather, was a result of typifying the shapes according to their typological characteristics.

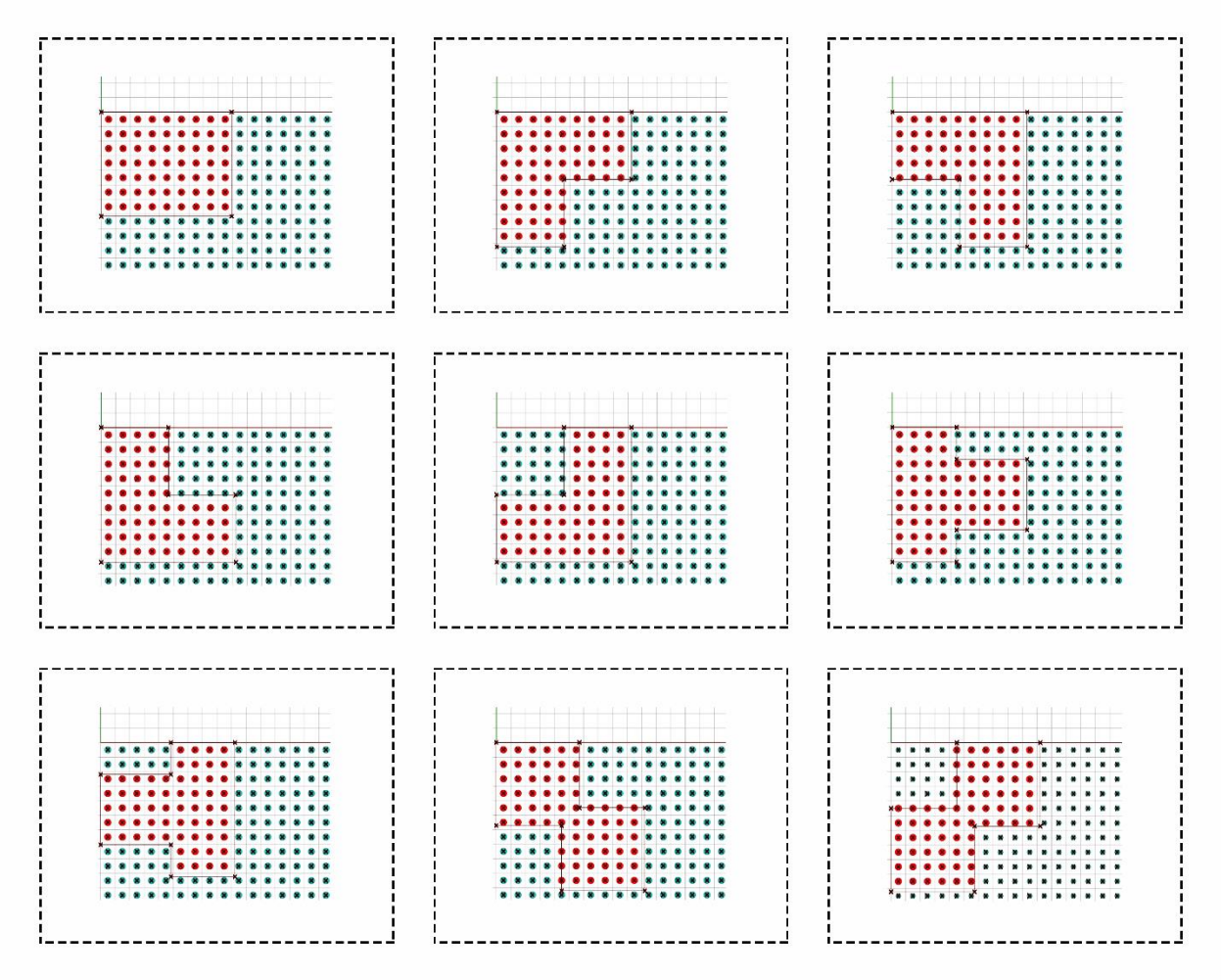

Figure 4-14. The 9 cluster representative shapes of the reference clustering set, subjected to the packing algorithm with 64 cells.

4.2.3. Applying the Clustering Method

Since two packing experiments were pursued, after the packing task, two calculation procedures of the "Shape Difference Score Matrix" for both experiment cases of 36 cells-packing and 64 cells-packing were conducted. The higher numbers of cells, compared to the 10 cells in Test-case 1, required extended computation time and necessitated the integration of a new plugin, an algorithm called *Batch-run*. Developed by Ramsden (2015) for Grasshopper, the Batch-run component enables the run of parametric change in an iterative process, changing one

value of a parameter at a time while keeping all the connected parameters' values constant. Connecting two parameters' sliders that represent the 72 shapes, each slider of (0-71) indices, for the cross-reference shape difference calculation, this calculation was performed for the *Pairwise Shape Difference and the Hungarian Algorithm* definition. The shape difference calculation has led to (72*71/2=2556) calculation steps.

As such, calculation of the shape difference for the 72 shapes was done in an agglomerative manner using the Batch-run component, and in each step, one case of pair-wise comparison is calculated for finding the shape difference value. Thus, a recorder component was needed to store every output shape difference value obtained from the pair-wise comparison. For Scenario 1, in each step of the Batch-run, the compared pair of shapes was overlapped in (36*36=1296) cases of overlap for every single pair-wise comparison of which the computation time needed was 10 seconds. Therefore, the total cases of 2556 comparisons required 25560 seconds, or 7 hours and 6 minutes. This was performed using an Intel® Core™ i7 8086K (6-Core/12-Thread, 12MB Cache) processor, and a video card of Dual NVIDIA® GeForce® GTX 1080 Ti graphics with 11GB GDDR5X each.

Using the same PC, in the second scenario, the 64 cell-packing, each pair-wise shape difference calculation took 80 seconds, leading to an overall calculation of the "Shape Difference Score Matrix" of (80*2556=204480) seconds, or 56 hours and 48 minutes. The extended computation time was expected due to the Hungarian Algorithm runs in all 64*64=4096 overlapping cases to find the smallest sum of cell distances, at each pair-wise shape comparison. Yet, an improvement can be made to reduce the computation time needed, as suggested in Section 5. Next, the cross-reference-based Shape Difference Score Matrix was constructed and

stored (recorded), first for the 36 cell-packing, and next for the 64 cell-packing. (Detailed explanation of Shape Difference Score and Matrix calculations can be found in Test-case 1.)

Similar to Test-case 1, the "Shape Difference Score Matrix" in both scenarios becomes the main input for the K-Medoids Algorithm. In addition, the variable of "Number of Clusters" was kept 9, for the purpose of comparison to the reference clusterings and the grid-based clustering result of (Rodrigues et al., 2017). In terms of computation time, the K-Medoids clustering algorithm performed relatively similar to Test-case 1, in less than 5 seconds, the algorithm organizes the set and performs the clustering results successfully. This computation time applies to the second scenario of 64-packed-cells as well.

For both scenarios, the outputs of the K-Medoids algorithm were: the nested list of clusters (C) with their shapes' IDs and the IDs of the medoids (M). Articulating the outputs and visualizing them were done next, using a color-coding to each cluster, and a darker tone for the medoid of each cluster. The clustering results for the two scenarios were compared to two clustering sets: (1) the reference clustering, and (2) the grid-based descriptor in (Rodrigues et al., 2017)'s study as described in the following subsections.

4.2.4. Experimental Clustering Outcome

To visualize and discuss the clustering results, it is important to note that in our algorithmic clustering, the shape type or label is unknown a priori, as it is unsupervised clustering. This means that the reference clustering set was not used for labeling or classifying the dataset; rather, the reference was only used for result comparison post to running the clustering algorithmic definition. Despite that the reference clusterings do not necessarily represent the most accurate clustering data or ground truth as it was performed subjectively and some shapes can be re-clustered differently, comparing our results to the reference set was

pursued implementing and testing clustering evaluation measures, explained the following subsection. Another method used to evaluate the clustering results was perceptual coherence or the presence of a dominant shape in each cluster.

4.2.4.1. Clustering Results of the 36 Cell-Packing-Scenario

The outcome of the clustering method, using 36 packed cells for the 72 shapes is illustrated in the below image of (Figure 4-15) compared to the reference set above. For discussing the resulting cluster subsets, labeling was added to the reference set with each subset labeled numerically as depicted in the top left corner above the shapes in (Figure 4-15). In addition, each reference cluster was given a color.

Our resulting clusters were given labels from A to I, similar to the reference set A' to I'. However, this labeling does not mean that the order of the resulting clusters follows the order of the reference set since it does not necessarily matter for the results to match the order of the reference set, as will be explained in the Subsection 4.2.5. What is important for accurate clustering results is the presence of a high number shapes that belong to the family (typology) of the dominant shape (Rodrigues et al., 2017) regardless of how the clusterings' order corresponds with the reference set.

As can be noticed in (Figure 4-15), the number of shapes per cluster varies from 3 for Cluster B to 16 for Cluster I. The cluster with the highest number of dominant shapes was cluster I with 11 mirrored L-shapes. Importantly, the clustering results show 7 unique dominant groups, represented in the medoids of the 7 clusters (A, B, D, E, G, H, I), and two clusters are repeated (Cluster C can be considered as a repeated dominant shape similar to Cluster B despite the different proportions, and Cluster F is similar to Cluster E with differences). In terms of perceptual coherence of the clusterings, almost every cluster has a dominant shape represented

by the medoid (darker tone) in each group; however, outliers do exist, when considering the reference clustering. The outliers can be identified as the shapes with different labels and colors from the dominant label and shape. It is noticeable that in several cases, two or more typological dominant shapes are present, except Cluster H with one dominant shape (the mirrored Z-Shape) and no outliers.

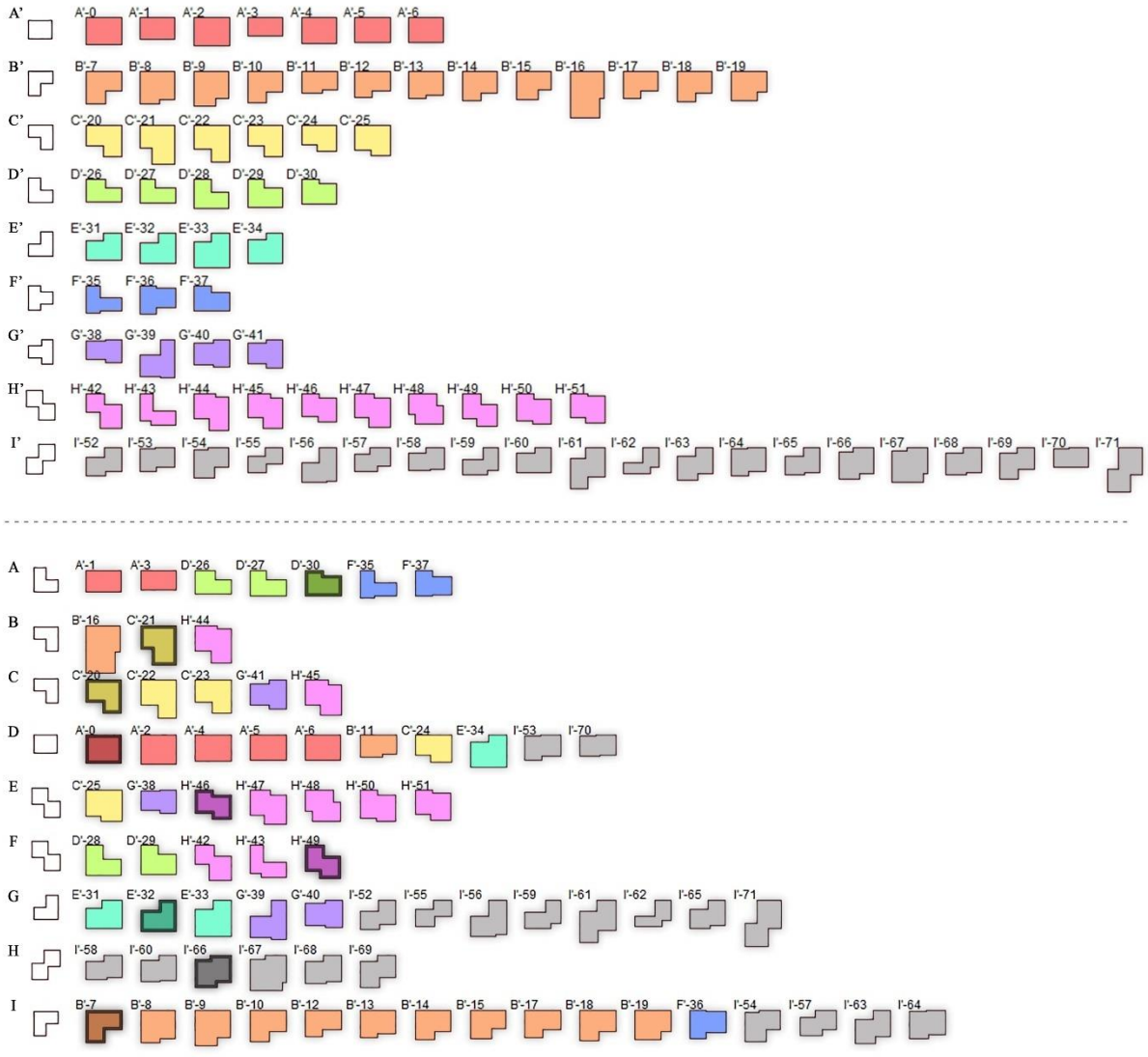

Figure 4-15. Above: the reference set; below: the clustering results of the 72 shapes using 36 cells packing, and the medoid of each cluster represented in a darker tone.

To explain the clusters with outliers, it is relevant to note that the clustering algorithmic definition calculates the shape difference and performs a region-based clustering, according to the Euclidean distance computation, regardless of the typified and boundary characteristics, whether it is an L-shape or a rectangle. In Cluster A, the L-shape (D'-30) was identified as the medoid, making 3 dominant shapes of D'-26, D'-27 and the medoid. The outliers can be divided into two groups, the two rectangles: A'-1, and A'-3, and the two L/rotated left (T-Shapes in the reference set) of F'-35 and F'-37. A user may assert that the actual outliers are only the two rectangles. To explain such behavior of clustering the rectangles with the dominant L-Shapes, it can be noticed that those L-Shapes and the rectangles have similar proportions, and the L/ rotated left T-Shapes have some slightly projected parts outside the L shapes.

Such importance of proportions is noticeable in Cluster B as well with the Shapes B'-16, C'-21, and H'-44 clustered together. The cluster dominant shape is the top-right L-Shape, represented in the medoid C'-21, and it is a very similar shape to H'-44. The possible outlier can be B'-15 with a different direction in the projected part that makes it a top-left L-Shape. However, what is in common amongst the three shapes is the width-to-height proportions which make them significantly different in proportions compared to other clusters. Obvious is that the height/width ratio is large.

This aspect of proportion is important in grid-based shape description, and it applies to other clusters; shapes with similar proportions of height to width tend to be clustered together. Also, it is significant to note that the shapes illustrated in (Figure 4-15) are visualized exactly as they appear in the reference set (of different areas), while in fact, they were subject to procedures of scaling, packing, and normalizing the number of cells in the bin-packing as illustrated in

(Figure 4-13). The reason for showing them as they are in the reference clustering is to facilitate the visual comparison of the results to the reference set.

Cluster C, which is similar to Cluster B yet with different proportions, has an evident perceptual coherence with dominant top-right L-Shapes (or similar to L-Shapes), and the medoid C'-20 is the representative shape. Two other L-Shapes of C'-22, and C'-23 are included in the cluster, with relative outliers of G'-41 and H'-45. Particularly, the Shape G'-41 is almost a top-right L-Shape with a marginal extrusion of its upper right corner which makes its membership to this cluster reasonable. Possibly, the reason for including Shape H'-45 into this group is due to its similar proportions to the other shapes and that it represents a small variation to the dominant top-right L-Shape.

Cluster D attains coherence, represented by the dominant rectangular shape, with the inclusion of proportionally similar non-rectangular shapes that make them almost rectangles. The medoid of Shape A'-0 makes it a dominantly rectangle cluster, along with Shapes A'-2, A'-4, A'-5, A'-6. In the case of Shapes B'-11, I'-53, and I'-70, they are almost rectangles with minimal projections outside an inscribed rectangle in each. Shapes C'-24, and E'-34 can be considered outliers, yet they can be considered of similar height/width ratios to the rectangles.

Examining the rest of clusters: E, G, H, and I, coherence was maintained with few possible outliers. It is noticeable that Cluster F is similar to Cluster E in shape dominance, yet with different proportions. Cluster F is less coherent with two dominant shapes included, the L-Shape and Z-Shape. Memberships of some of the outliers to their clusters can be attributed to the number of 36 cell-packing that makes some few differences in cells marginal. It is expected that packing the shapes with more cells will enhance the subsets' perceptual coherence. Another reason for the existence of outliers is that the reference set was clustered differently, with focus

on shape typologies and space distributions inside each building, which makes it challenging to compare to this method's results.

An important point of discussion in regards to the clustering results is that there can be a different result, each time running the K-Medoids clustering algorithm; that is the algorithm has a random function in initiating the medoids (Bauckhage, 2015). Such randomness will be explained in the following subsection of validation. Despite this characteristic, examining each clustering result, overall, proportions are important to determine the clusterings. In using grid-based description, it is sensitive to variation in proportions since it relies on the cells contained in the shape and lower height of a rectangle compared to a higher rectangle represent two different shapes (Rodrigues et al., 2017); it is primarily a region-based and does not consider shape boundaries.

4.2.4.2. Clustering Results of the 64 Cell-Packing-Scenario

In the scenario of 64 cell-packed shapes, the outcome of the clustering method is illustrated in the lower part of (Figure 4-16), in relation to the reference set above. As in the case of 36 cell-packing, the clusters were illustrated and color-coded according to the same colors and labeling used for the reference clustering. The number of shapes per cluster varies from 5 for cluster F and I to 14 for cluster D. Overall, the 9 clustering sets show 7 unique dominant shapes, signified by the 7 medoids of the clusters (A, C, D, E, F, G, H), while Cluster B can be considered similar to Cluster A with a dominant rectangle shape, and Cluster I is related to Cluster H with a dominant mirrored or reflected Z-Shape. It is noticeable in the resulting sets that overall, perceptual coherence has been relatively improved from the results of the first scenario with a slightly higher number of dominant shapes and somewhat higher accuracy measure, as will be explained in the Validation subsection. Dominant shapes were completely attained in

Clusters F and H with no outliers. Other clusters are perceptually coherent, yet include outliers that belong to other dominant shapes, particularly in Clusters (A and B) with three or more typified outlier shapes within each cluster, while the remaining sets have 2 or less typified outliers.

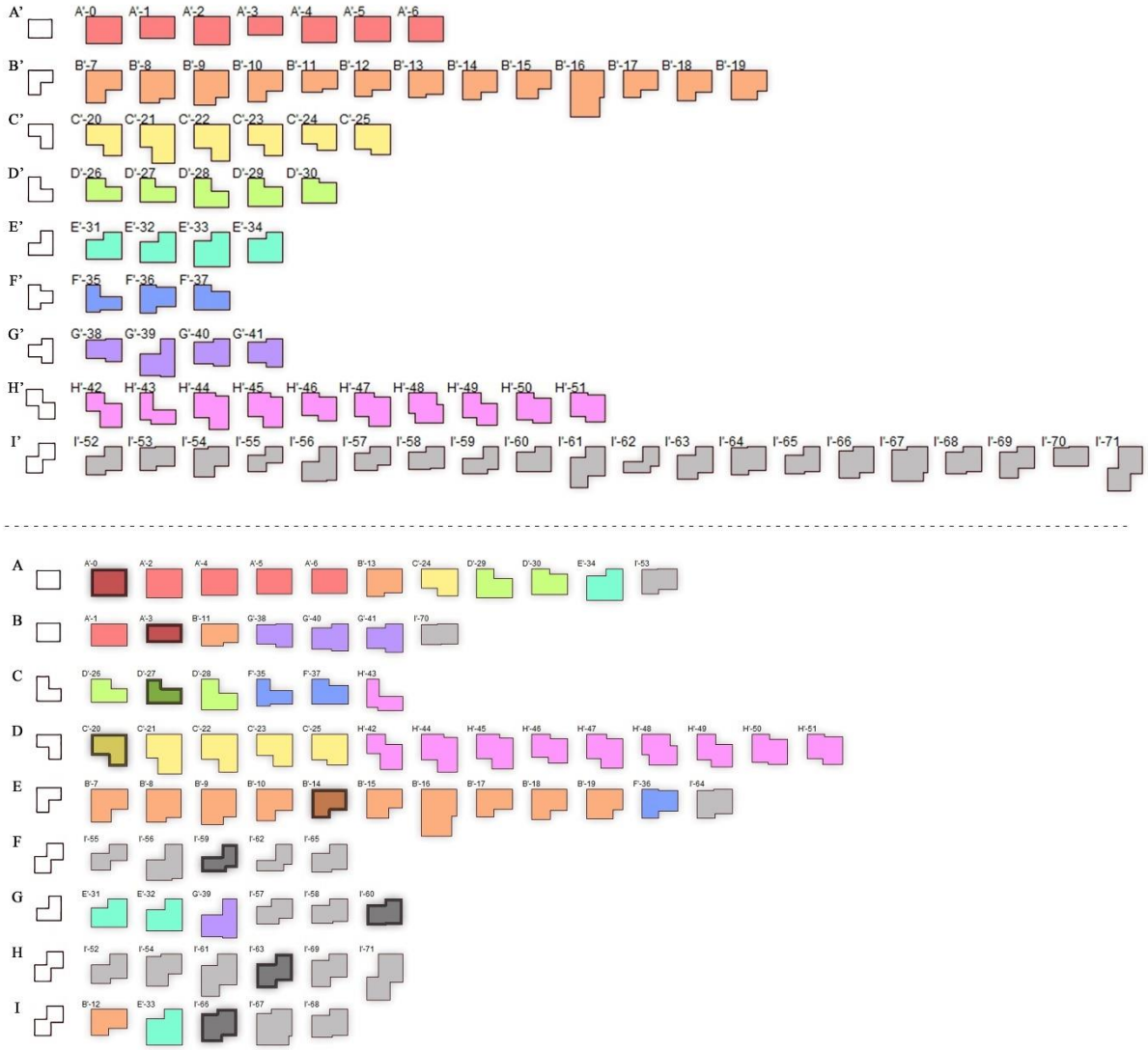

Figure 4-16. The clustering results of the 72 shapes using 64 cells packing, and the medoid of each cluster represented in a darker tone.

To explain the clusters with multiple typified shapes of outliers, such as Cluster A, the dominant shape represented in the medoid, the rectangle, is of particular proportions that are similar to the L-shapes clustered within this group. In the case of Cluster B, although it can be considered as another rectangle-dominated shape because of the rectangle medoid, the proportions are different here with less height to width ratio in comparison with Cluster A. In the clusters (C, D, E, G, and I), it can be noticed that L-Shapes were clustered with Z-Shapes and other shapes of similar direction or/and proportions.

In Group A, the five rectangular shapes (A'-0, A'-2, A'-4, A'-5, A'-6) have been accurately clustered together, yet outliers of (B'-13, C'-24, D'-30, E'-34, I'-53) are included, and inscribe similarly proportioned rectangle inside with some projected parts, while D'-29 can be regarded as the only obvious outlier with a relatively bigger projected portion out of the rectangle and represents an actual L-Shape. The same applies to Cluster B with rectangle shapes (A'-1, A'-3) and related shapes with similar proportions and slightly projected parts out of the rectangle-inscriptions of (B'-11, G'-38, G'-40, G'-41, I'-70). Group C shows evident perceptual coherence with a dominant L-Shape (D'-26, D'-27, D'-28) and an almost L-Shape of (F'-37), while the shapes (F'-35 and H'-43) have slightly projected parts to the L-Shape inscription. Group D combines two families of shapes, the dominant top-right L-Shape and the Z-Shape with similar direction to the L-Shape. Cluster E is the largest coherent set with 10 top-left L-Shapes and very similar shapes of (F'-36 and I'-54) as possible outliers. Cluster G represents another case of a group with two dominant shapes (a mirrored L-Shape and mirrored Z-Shape) with similar direction. Clusters F and H, as mentioned, contain fully attained coherence with no outliers. In Cluster I, the mirrored Z-Shapes (I'-66, I'-67, I'-68) are accurately clustered, while

Shape B'-12 shares similarities in terms of the lower projected part being similar to the mirrored Z-Shapes, while Shape E'-33 signifies similarities of the upper projected part to those Z-Shapes.

4.2.5. Validation Study

The clustering results of the two scenarios have been subjected to validation studies with quantified analysis of the clustering accuracy as explained in the next subsection.

4.2.5.1. Validation Study of Clustering the 36 Cell-Packing-Scenario

To validate the clustering result of 36-cell-packing, it was first compared to the reference clustering of (Rodrigues et al., 2017). To compare the resulted clustering set to the reference set, a clustering accuracy calculation was utilized. The most common method to compute the accuracy of resulting clustering data is to calculate the percentage of the data that has been correctly clustered with reference data (Story & Congalton, 1986). This calculation is often done using an error or a confusion matrix that can be represented as a table of the clustered data as columns, and the reference data as the rows of the matrix (Story & Congalton, 1986) or vice versa. Considering the Confusion Matrices (a) and (b) in (Table 4-3), Matrix (a) represents the comparison of the clustering results of the grid-based (GB) descriptor of (Rodrigues et al., 2017) to the reference clustering, and Matrix (b) is the comparison of the clustering results of this test-case using the 36-packed cells, against the same reference clustering.

It is important to emphasize here that both the shape description method and the clustering method of this work are different from Rodrigues et al.'s work. In their shape description method, Rodrigues et al. have used the grid's binary vector as a matrix with each matrix contains the corresponding values of the overlaid grid, while in our grid-based shape comparison, an exhaustive search for an optimum overlap enabled by the Hungarian algorithm in the pair-wise shape comparison was pursued. The other difference is the clustering method; in

their work, the Ward linkage clustering method was used for grouping the shapes, while we used K-Medoids clustering. Those two differences can explain the reason for different clustering results.

The two matrices, (4-3-a) and (4-3-b), depict how the shapes in Clusters (A to I) in the grid-based descriptor in Rodrigues et al. (2017)'s study and our clustering sets have been dispersed in relation to (A' to I') in the reference clustering set respectively. In our clustering results in Matrix (4-3-b), for instance, in the column of Cluster A, 2 shapes correspond with the A', 3 shapes belong to Cluster D', and 2 belong to Cluster F'. Similarly, all the other columns were organized, via scattering the shapes according to their reference clusters. The highest value in each column has been shaded in grey. For the accuracy calculation that will be explained next, the shaded boxes are considered for the SUM function.

Table 4-3. Confusion Matrix (4-3-a) for comparing clustering results of the Grid-based descriptor of (adapted from Rodrigues et al., 2017) to the reference clustering, and Matrix (4-3-b) for comparing the test-case results to the reference.

Results of clustering using a grid-based descriptor (Rodrigues et al.) compared to the reference set

		Clustering Results (Fixed aspect ratio)								
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
Reference Clustering	**A'**	3					4			
	B'	1			3			8	1	
	C'					2			1	3
	D'	2	3							
	E'			2	1					1
	F'	1	1					1		
	G'	1			1					2
	H'		4			3				3
	I'	2		8	5			3	2	

(4-3-a)

Results of clustering using 36 cell-packing compared to the reference set

		Clustering Results (36-cell-packing)								
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
Reference Clustering	**A'**	2			5					
	B'		1		1					11
	C'		1	3	1	1				
	D'	3					2			
	E'				1			3		
	F'	2			2					1
	G'			1		1		2		
	H'		1	1		5	3			
	I'							8	6	4

(4-3-b)

The methods used for evaluating the clustering accuracy in their study were also pursued here for comparison, using the same two metrics: accuracy, and the Rand Index. One method to calculate the overall level of accuracy in confusion matrices is performed by dividing the sum of the highest values in each column in the above matrices over the total number of the sample data. This accuracy measure of each cluster was computed using the following formula:

$$\boldsymbol{Accuracy} = \boldsymbol{SUM}\ (\boldsymbol{highest\ value\ of\ each\ column})/\boldsymbol{SUM}\ (\boldsymbol{matrix})$$

$$\boldsymbol{Accuracy\ in\ Matrix}\ (\boldsymbol{4-3-a}) = \boldsymbol{40/72} = \boldsymbol{55.5}\%$$

$$\boldsymbol{Accuracy\ in\ Matrix}\ (\boldsymbol{4-3-b}) = \boldsymbol{45/72} = \boldsymbol{62.5}\%$$

According to this calculation, the average accuracy of all clusters' accuracies of Confusion Matrix (4-3-b) is 62.5%, in contrast to 55.5% of accuracy for the Rodrigues et al. (2017)'s grid-based clustering results with fixed aspect ratio illustrated in Matrix (4-3-a). In their study, Rodrigues et al. have used two scenarios, one with a fixed aspect ratio, and another with a non-fixed aspect ratio for each descriptor, including the grid-based. Our comparison study was carried out in relation to the fixed aspect ratio of Rodrigues et al.'s work since in our bin-packing algorithm, the aspect ratio was also maintained.

Further, other metrics were pursued evaluating the clustering results of which the Rand Index was considered important. Developed by Rand in statistics, the index is particular for measuring data clustering by calculating the similarity between two sets of clusterings (Rand, 1971). For Rand (1971), evaluation of a clustering method requires either comparing its results to standard results or to another result. For two clustering sets that need comparison such as $X=\{X_1, \ldots, X_n\}$ *and* $Y=\{Y_1, \ldots, Y_n\}$ calculating the Rand Index is performed as follows: (Rand, 1971).

$$\boldsymbol{Rand\ Index} = (\boldsymbol{TP} + \boldsymbol{TN}) / (\boldsymbol{TP} + \boldsymbol{FP} + \boldsymbol{FN} + \boldsymbol{TN})$$

Where:

TP: True Positives, the number of pairs of items in X that are in the same subsets in Y

TN: True Negatives, the number of pairs of items that are in different subsets in X and in different subsets in Y

FP: False Positives, the number of pairs of items that are in the same subsets in X and in different subsets in Y

FN: False Negatives, the number of pairs of items that are in different subsets in X and in the same subsets in Y

In addition to the Rand Index, other metrics were calculated for additional comparisons. One of the metrics that is often used for clustering evaluation is *Precision* which can also be called the *confidence value* that "denotes the proportion of predicted positive cases that are correctly real positives" (Powers, 2011, p. 2). Another important measure for assessing clustering is the *Recall* or *Sensitivity* metric which is used to identify the rate of real positive items that are correctly predicted positive; the measure considers the ratio of the True Positives over the total amount of items that are True Positives and False Negatives (Powers, 2011). Calculating those two measures facilitates the retrieval of one more metric called the *F1-Score* or *F-measure* which is the harmonic average of both measures Precision and Recall (Powers, 2011). When F1-Score reaches its maximum value at 1, this means perfect precision and recall are attained (Sasaki, 2007). The three metrics are calculated as follows: (Powers, 2011).

$$\boldsymbol{Precision} = (\boldsymbol{TP}) \,/\, (\boldsymbol{TP} + \boldsymbol{FP})$$
$$\boldsymbol{Recall} = (\boldsymbol{TP}) \,/\, (\boldsymbol{TP} + \boldsymbol{FN})$$
$$\boldsymbol{F1} = (\boldsymbol{2.0} * \boldsymbol{Precision} * \boldsymbol{Recall}) \,/\, (\boldsymbol{Precision} + \boldsymbol{Recall})$$

Calculating the four measures, the Rand Index, Precision, Recall, and F1-Score was performed through implementing the Python program-based algorithm of (Tom, 2014) that has been developed according to Manning et al.'s work (2008). Examining the values of those

measures in Table 4-4, Matrix (4-3-b) of our clustering results using the 36 cell-packing-scenario gives higher values with the Rand Index of 0.85, Precision of 0.44, Recall of 0.41, and F1-Score of 0.42, compared to 0.82, 0.37, 0.28, and 0.32 respectively for Matrix (4-3-a) of the (Rodrigues et al., 2017)'s grid-based results.

Table 4-4. Results of the metrics for the clustering evaluation comparing the two confusion matrices discussed above, (4-3-a) and (4-3-b).

Clustering Metrics of Confusion Matrix (4-3-a)	Clustering Metrics of Confusion Matrix (4-3-b)
TP: 105, FP: 182, TN: 2000, FN: 269 **Rand Index: 0.823552** **Precision: 0.365854** **Recall: 0.280749** **F1: 0.317700**	**TP: 141, FP: 182, TN: 2030, FN: 203** **Rand Index: 0.849374** **Precision: 0.436533** **Recall: 0.409884** **F1: 0.422789**

4.2.5.2. Randomness in Clustering the 36 Cell-Packing-Scenario

It is important to signify that our results (45/72) have been retrieved in one instance of running the K-Medoids clustering method, with 1000 iterations, and when re-running the algorithm, different clustering results emerge. For 8 cases of different clustering results retrieved from running the algorithm 8 times, the range of accuracy was between (42-45/72). Two examples of those emerged clustering results are illustrated in (Figure 4-17) with accuracy measures of (43/72=59.7%) for the upper clustering, and (45/72=62.5%) for the lower clustering, respectively. Overall, the clustering results may not be of significantly higher accuracy compared to Rodrigues et al.'s results because the reference set was determined subjectively, and through typifying the shapes by human examiners. The higher accuracy result was not the exact target of our clustering method, but the comparison helps us understand whether our clustering results are reasonable and comparable to the reference.

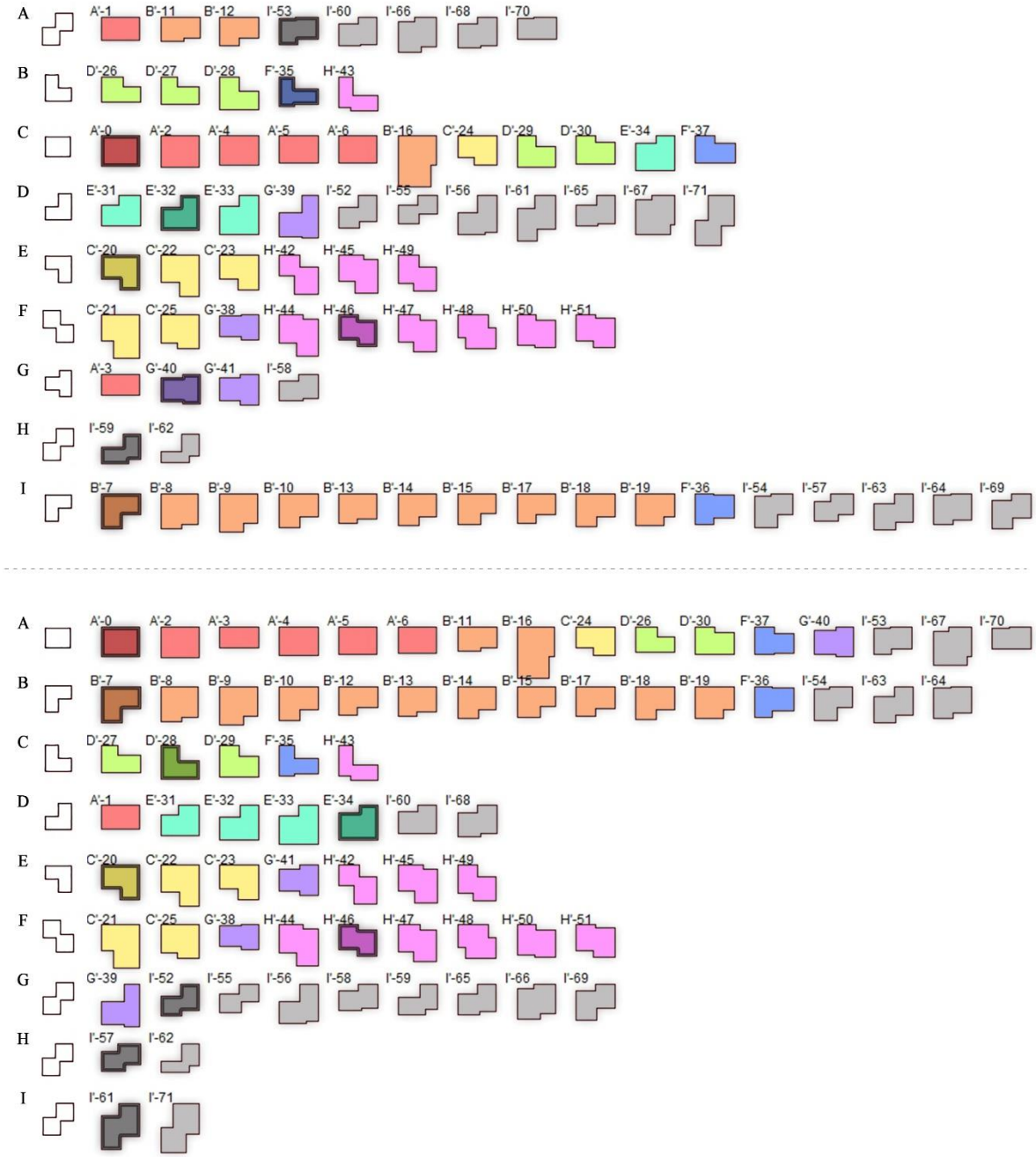

Figure 4-17. Two sample clustering results of the 36 cell-packing-scenario that emerge from running the K-Medoids clustering algorithm twice.

This behavior of retrieving different clustering results at each time running the clustering method can be attributed to the randomization used in the K-Medoids clustering (Bauckhage,

2015). In implementing the K-Medoids algorithm of Bauckhage (2015), when importing the Numpy library, the random function was called and utilized. In particular, randomness was used when initializing the medoids, which is performed early in the program (Bauckhage, 2015). Yet, despite the initial randomness in selecting the representative objects, the search in K-Medoids exploits all the possible combinations of representatives (medoids) and non-representatives (non-medoids) which are analyzed (Jin & Han, 2016) . It has been noticed that after certain repeated clustering runs (5-10 runs) the clustering results start to repeat in a random order. This means that the clustering results can be reproducible by the users of the method yet with multiple tests.

This impact of the random function in the K-Medoids algorithm on retrieving different clustering results running the algorithm multiple times and despite using a high number of iterations, such as 1,000,000, is expected. The behavior can be explained as follows. In initiating the medoids, for 9 clusters, the possible options represent a huge number calculated as (72x71x70x69x68x67x66x65x64, or 72! /63! = 3.0885807e+16). Even when using 1,000,000 iterations of K-Medoids clustering, it is still a fraction of the possible options of medoids. It is the complexity caused by the number of shapes, and the number of clusters that is impactful, and not the actual shape difference values since the algorithm is straightforward in calculating distances. However, despite that the randomness leads to different results, all the results are reasonable and consistent, with no substantial difference (Han et al., 2011).

4.2.5.3. Validation Study of Clustering the 64 Cell-Packing-Scenario

Applying the same validation method to the second scenario of 64 cell-packing has led to different clustering results, thus, different clustering evaluation measure values. In comparing the clustering results of the 64 cells-packing to the reference set in (Rodrigues et al., 2017)’s work, the results are illustrated in the Confusion Matrix (4-5-a) in Table 4-5.

Table 4-5. Confusion Matrix (4-5-a) for comparing the clustering results of the test-case results with 64 cell-packing to the reference clustering, and Matrix (4-5-b) for comparing the test-case results of 36 cell- packing to the 64 cell-packing.

Results of clustering using 64 cell-packing compared to the reference set

		Clustering Results (64-cell-packing)								
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
Reference Clustering	**A'**	5	2							
	B'	1	1			10				1
	C'	1			5					
	D'	2		3						
	E'	1						2		1
	F'			2		1				
	G'		3					1		
	H'			1	9					
	I'	1	1			1	5	3	6	3

(4-5-a)

Results of clustering using 36 cell-packing compared to the clustering results using 64 cell-packing

		Clustering Results (36-cell-packing)								
		A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
Clustering Results 64 cell-packing	**A**	1			8		1			1
	B	2			2	1		1		
	C	4					2			
	D		2	4			2			
	E		1			6				11
	F							5		
	G			1				3	2	1
	H							3	1	2
	I							1	3	1

(4-5-b)

Calculating the accuracy measure of Matrix (4-5-a) was done using the same equation considering the highest number in each column as follows:

$$\boldsymbol{Accuracy\ in\ Matrix\ (4-5-a)\ =\ 47/72\ =\ 65.2\%}$$

Another comparison study was pursued, considering the resulting clusterings of the 64 cells-scenario as a reference set, and comparing the 36 cells-scenario to it as represented in (Figure 4-18). The objective of this comparison was to additionally test the clustering results, and to internally examine the two different results of Scenario 1 (36 cell-packing) and Scenario 2 (64 cell-packing), with no consideration of the reference set of the (Rodrigues et al., 2017)'s research. Articulating the clustering results of Scenario 1 in reference to Scenario 2 is illustrated in Matrix (4-5-b), leading to an accuracy value of 62.5%, calculated as follows:

$$\boldsymbol{Accuracy\ in\ Matrix\ (4-5-b)\ =\ 45/72\ =\ 62.5\%}$$

Figure 4-18. Above: the clustering subsets of the 64 cell-packing-scenario re-illustrated as a reference set; below: the clustering results of the 36 cell-packing-scenario re-illustrated according to the 64 cells with new color-coding.

This comparison offered insight into the reasonable results of comparing two clusters internally, without the use of a reference set. The assertion here is that when comparing two

clustering results, using one as reference, the metrics show that the results are comparable and reasonable. Such comparison informs that the algorithm performs consistently with different resolutions.

In regards to the other clustering evaluation measures, Matrix (4-5-a) of 64 cell-packing compared with the reference set has led to values of Rand Index of 0.85, somewhat higher Precision with 0.48, a lower Recall measure of 0.38, and a marginally lower F1-Score of 0.42 in relation to the clustering evaluation measures of Matrix (4-3-b). When calculating the four measures for Matrix (4-5-b), this comparison shows the results of 0.85 for the Rand Index, 0.42 Precision, 0.42 Recall, and 0.42 F1-Score as shown in Table 4-6. In comparing the metrics of Matrix (4-5-b) to Matrix (4-5-a), the Rand Index and Recall values are slightly higher, while the Precision and F1 measures are somewhat lower in (4-5-b). In analyzing the results of changing the reference set in Matrix (4-5-b), the metrics were somewhat similar, with marginal differences, reasonable in each case.

Table 4-6. Results of the metrics for the clustering evaluation comparing the two confusion matrices discussed above, (4-5-a) and (4-5-b).

Clustering Metrics of Confusion Matrix (4-5-a)	Clustering Metrics of Confusion Matrix (4-5-b)
TP: 142, FP: 156, TN: 2026, FN: 232 **Rand Index: 0.848200** **Precision: 0.476510** **Recall: 0.379679** **F1: 0.422619**	**TP: 136, FP: 187, TN: 2047, FN: 186** **Rand Index: 0.854069** **Precision: 0.421053** **Recall: 0.422360** **F1: 0.421705**

4.2.5.4. Randomness in Clustering the 64 Cell-Packing-Scenario

Similar to the first scenario, in this scenario of 64 cell-packing, the K-Medoids clustering gives more than one result when re-running the algorithm, leading to a range of accuracy measures (42-47/72) for 8 results retrieved. Two of those results are depicted in (Figure 4-19), with accuracy measures of (45/72=62.5%) for the upper clustering, and (43/72=59.7%) for the lower clustering, respectively. Noticeable is the similarities of the cluster subsets between the two results, and in some cases like Clusters D and I in the top image, and Clusters C and E in the bottom, the subsets are identical. This leads to the assertion that despite randomness, the results can converge to the optimum assignment of shapes to their medoids, in grouping the clusters.

Figure 4-19. Two sample clustering results of the 64 cell-packing-scenario that emerge from running the K-Medoids clustering algorithm twice.

4.2.6. Discussion and Conclusions

In this test-case, the SC-KM method was further improved, incorporating a packing algorithm to assign a grid-based description to the selected shapes. More importantly, the pursuit of clustering evaluation and external validation was conducted in this experiment. In this external validation, the accuracy measure and all the 4 clustering measures used show higher values than the compared study for the two carried out scenarios, as illustrated in the cases shown in the tables. However, due to K-Medoids algorithm's randomness, some clustering can be lower. One reason for the differences between the results of this SC-KM method and the compared method can be the pair-wise shape comparison that yielded an impact to the improved results. In addition, the difference between the K-Medoids clustering used here and the Ward linkage used by Rodrigues et al. can be the reason to lead to different results. It is expected that the K-Medoids and partitioning clustering methods, in general, perform better in comparison with other methods (Jayanti et al., 2009).

In terms of comparing the two scenarios, it is important to note that despite the higher resolution, it was not predicted that the results can be improved from 32 cell-packing to 64 cell-packing due to the use of the reference set which is not the ground truth. The reference set does not necessarily represent the best clustering since it was created by human examiners typifying the shapes into subsets of typological characteristics without the use of an algorithmic clustering (Rodrigues et al., 2017). Overall, perceptual coherence of the clusterings can be considered satisfactory, with the existence of outliers.

In terms of the impact of randomness, it has been noticed that it does not necessarily lead to a substantial change in clustering accuracy; the results are reasonable in every run of the clustering algorithm. Important to mention here are the studies that aimed to replace the random

selection of initial medoids, e.g. (Gullo et al., 2008) in which a refined strategy of Kaufmann & Rousseeuw (Kaufmann & Rousseeuw, 1987) was used instead of random initiation of the medoids. However, the replacement of randomness with another procedure did not give a considerable improvement.

4.3. Test-case Experiment 3

In order to apply the SC-KM method to various architectural shapes that could evolve in GDSs and are beyond the specific setup of Test-case 1, or the 72 sample shapes in Test-case 2, this test-case was carried out. More importantly, in this experiment, the three prototypical processes discussed in Section 3 were pursued, with the inclusion of building performance evaluation and optimization, leading to a fully working prototype that demonstrates the new system. In addition to those processes, a discussion of the clustering method applied to the optimal and near-optimal generated design solutions is offered. The following sub-sections describe those processes and tasks in detail. Subsequently, the clustering outcome is visualized and discussed. Finally, at the end of this experiment part, an overall discussion and concluding points are given.

4.3.1. Introduction to the Test-case Experiment

The prototypical processes pursued in this experiment can be briefly explained again as follows. In Process 1, the parametric form generation was carried out through programming a parametric setup of a spatial configuration to generate a dataset of building shapes with similarities and differences. Process 2, design optimization, involved conducting building performance evaluation and optimization of the design candidates. Process 3 is the application and testing of the SC-KM to the resulted optimal and near-optimal solutions with visualization

and presentation. The three processes are represented in the workflow in (Figure 4-20) with the tools used.

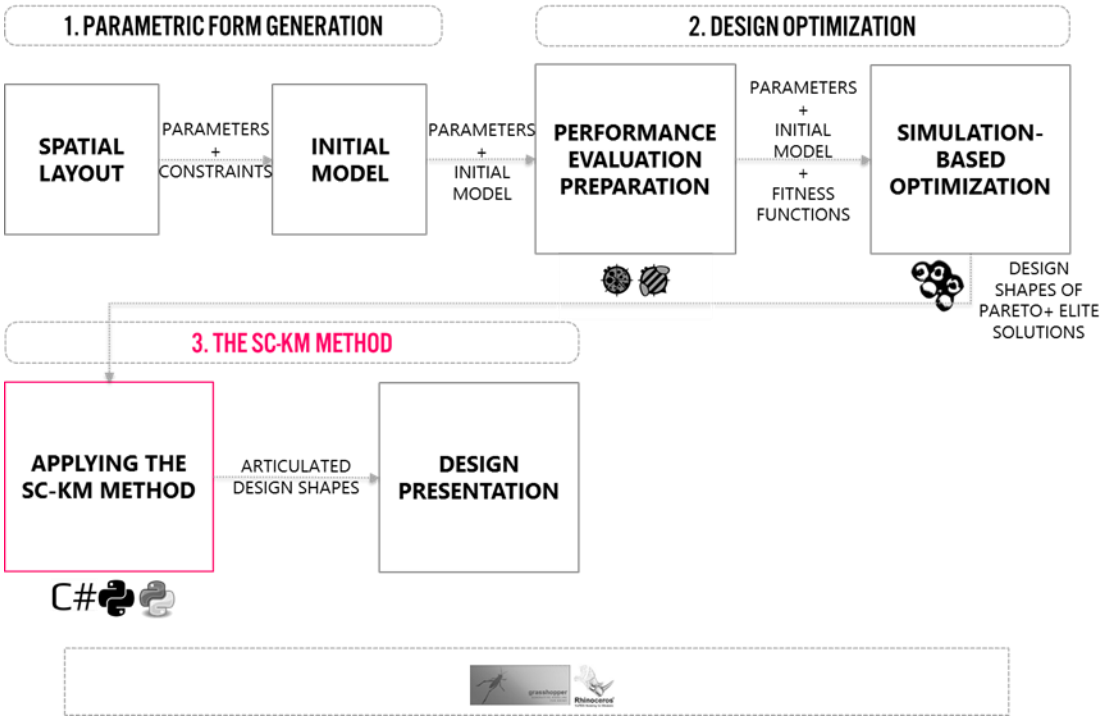

Figure 4-20. The workflow of Test-case 3 with the incorporation of the building performance evaluation and optimization process.

In creating the initial model for this test-case, the building layout was parametrically modeled as a simple abstracted arrangement of four masses or spaces, with adjacency and connection relations. Such a layout can be simple, yet in adding additional masses or zones, it is representative of a large array of spatial arrangements. The masses' dimensions were parameterized, and constrained, to allow for the building configuration to change to produce a range of design shapes to respond to the environmental performance objectives sought for simulation and optimization. In addition to the parametrized width and length dimensions of each of the four masses, the Window-Wall-Ratio (WWR) was parametrized as well. It is expected that for satisfying minimum energy, particularly the minimum cooling loads for a hot climate in the

summer (that was used as the fitness value), the building configuration will tend to be compact with the lowest surface area/volume ratio, or minimum surface area exposed to the exterior thermal conditions and heat gain, and with low WWR to avoid direct solar heat gain. However, for achieving an appropriate interior daylight illuminance, which was another determined fitness value, it is predicted that the building configuration would be fragmented (distributed) to increase the surface area/volume ratio and the possibilities to have a high WWR (Caldas, 2001).

In terms of preparing for shape comparison and clustering, the parametric building layout was determined to always contain an exact number of grid-based cells (48) for all the generated shapes, to facilitate accurate shape clustering, and thus the exact number of cells became a constraint. Overall, the layout was parameterized and constrained to prepare for testing the clustering method into a generative scheme with environmental performance evaluation and optimization.

4.3.2. Parametric Form generation

The building layout was set up to contain four adjacent masses, originating from the origin point of the coordinates (x=0, y=0) as illustrated in (Figure 4-21). Each mass is composed of 12 cells, each of which is 5 meters by 5 meters, in dimensions. A sample of the possible shapes that are generated from this layout model is shown in (Figure 4-23). The north orientation is the green axis shown in the figure, a default in the modeling software Rhino. With 12 cells in each mass, a fixed area of (25*12=300 square meters) results as the area of any mass, and (4*300=1200 square meters) becomes the total area of the building. The height was left as a fixed dimension of 10 meters, assuming that the interior spatial program inside the building is flexible to include multiple levels (Yousif et al., 2017). The layout has been governed by the following constraints:

- *A fixed number of cells*: the reason for selecting the number of cells for each mass to be always 12 is for the purpose of facilitating the grid-based descriptor's pair-wise comparison in the SC-KM method, in order to retrieve the same number of cells for all shapes, which is (12*4=48).
- *A determined range of parametric values*: for each of the four masses, the dimensions are parametrically driven by the number of cells. Each side of any mass has to be one of the factors of 12 within the range of (2, 3, 4, 6) leading to possible dimensions of (10 m, 15 m, 20 m, 30 m) respectively. The reason for avoiding factors 1 (cell) and 12 (cells) is practical, as space will be too narrow (5 meters) when using such values. Those values represent the possible options for one dimension of each rectangular mass which is the length (x), while the width (y) is dependent on the length, computed as the outcome of the following formula:

$$\boldsymbol{Number\ of\ cells\ of\ the\ width\ =\ Total\ cells\ (12)/\ Number\ of\ cells\ of\ the\ length}$$

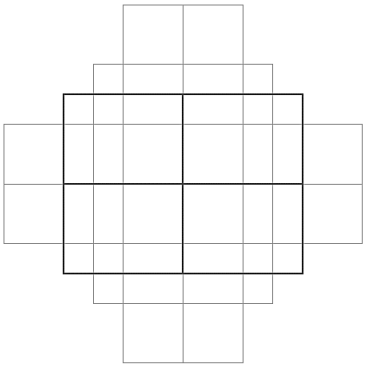

Figure 4-21. Top view of the initial setup of the layout of Test-case 3.

As such, the layout consisted of 4 explicit (independent) parameters for the length, and 4 indirectly changeable (dependent) parameters for dimensions of the width, driven by the change in length. The cells of each mass were subjected to solid union operation to convert them into

one solid volume of the rectangular mass as the cells are not needed for simulation or optimization, and they are only important for clustering purposes. After setting up the initial mass model, for each façade (north, south, east and west), windows were added according to a list of parameters of the Window-Wall-Ratio (WWR) of (0, 0.1, 0.2, 0.3, 0.4) for each of the east and west façades, and (0.4, 0.5, 0.6, 0.7, 0.8), also for each of the north and south façades. The rationale for using lower values for the east and west façades is due to the research-backed rule that for east and west facades, large glazed surfaces are not favorable for excessive thermal heat gain (Caldas, 2001), particularly for the selected hot climate of College Station, TX, with its dominated cooling loads. Also, the reason for choosing high WWRs for north and south facades is to balance the low ratios of east and west and to test what design solutions would the MOEA run evolve. Samples of the resulting parameters of the WWR are illustrated in (Figure 4-22). This setup yields 4 length parameters of 4 values for the mass model, 2 parameters of 5, and other 2 parameters of 5 ranges for the WWR leading to thousands of possible options, of which 4 possible shapes are illustrated in (Figure 4-23). The figure four columns of 4 different shapes, and three rows for different views of those shapes. For each shape, the top image is for a 3D view that depicts the 4 masses in dark grey, the middle image is a 3D view showing the shape with windows added, while the bottom image is a top-view of the shape with the cells highlighted in green grid lines. The overall specified building parameters are illustrated in (Table 4-7).

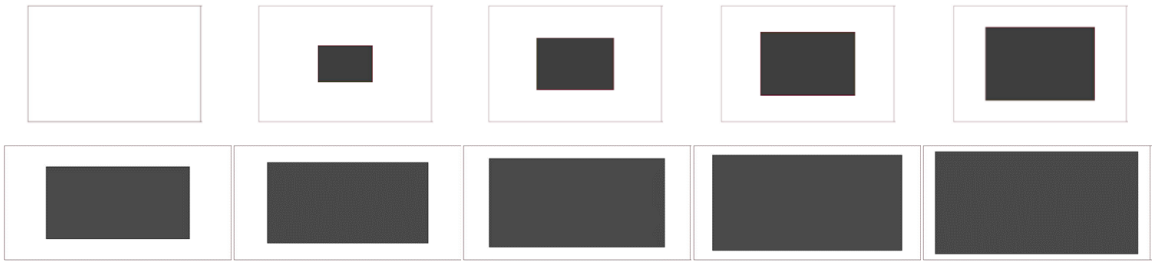

Figure 4-22. Above: Samples of exterior walls, parametrization of WWR for east and west facades, from left to right: WWR of (0, 0.1, 0.2, 0.3, 0.4) respectively. Below: Samples of exterior walls, parametrization of WWR for north and south facades, from left to right: WWR of (0.4, 0.5, 0.6, 0.7, 0.8) respectively.

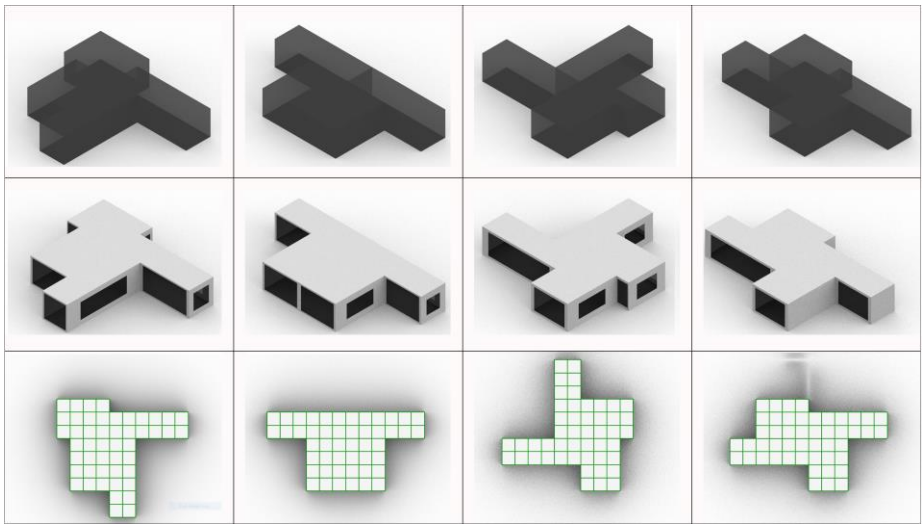

Figure 4-23. A sample of 4 possible shapes that the parametric layout of Test-case 3 produces. Above: 3D mass models with the inner cells; middle: 3D of possible fenestration added to the mass models; below: top view of the options with the inner cells highlighted.

Table 4-7. Building parameters (variables).

Geometry Variables	Dimensions, Length Limits (m)	Dimensions, Width Limits (m)	Variable Type
Mass 1 (Northeast)	(10, 15, 20, 30)	(10,15,20,30)	Discrete
Mass 2 (Northwest)	(10, 15, 20, 30)	(10,15,20,30)	Discrete
Mass 3 (Southwest)	(10, 15, 20, 30)	(10,15,20,30)	Discrete
Mass 4 (Southeast)	(10, 15, 20, 30)	(10,15,20,30)	Discrete
Window-Wall-Ratio	**WWR Range**	**Step**	**Variable Type**
North Façade	(0.4, 0.8)	0.1	Continuous
South Façade	(0.4, 0.8)	0.1	Continuous
East Façade	(0.0, 0.4)	0.1	Continuous
West Façade	(0.0, 0.4)	0.1	Continuous

It is important to note that the building program was simplified in this setup to an open office area for the 4 masses, considering an early conceptual design phase. However, in real building design, more complex program and other factors like design intents dictate the spatial layout. Often, proximity amongst the program spaces is desired. In following this simplified model setup, it is expected that designers can freely set up the initial layout considering the building program requirements, and more importantly, pack the model with cells for retrieving grid-based shapes. In the next phase, the building mass model is converted into an analytical (simulation) model assigning building materials, site conditions, and other components to describe the building and prepare for energy simulation and daylight analysis as described in the following parts.

4.3.3. Design Optimization (Performance Evaluation and Optimization)

The parametric mass definition was then prepared for simulation in several tasks. The environmental analysis was conducted using Ladybug and Honeybee simulation tools that rely on other environmental analysis engines such as EnergyPlus, DAYSIM, and OpenStudio (Roudsari et al., 2013). Octopus, the Multi-Objective Evolutionary Algorithm MOEA-based tool was utilized as the search mechanism to generate an initial population and search for fitter solutions in regards to predetermined objective functions. Octopus is based on the Strength Pareto Evolutionary Algorithm (SPEA-2), an improved elitist multi-objective evolutionary algorithm (Vierlinger & Bollinger, 2014). The two building performance metrics chosen for simulation and optimization here were the minimum cooling loads and maximum preferred daylight illuminance ratio. The preparation needed for both objectives is explained in the following subsections.

4.3.3.1. Preparation for Energy Simulation

The minimum energy consumption of the building model was considered as a metric to parametrically improve the building performance in the generative run. In particular, the minimum cooling loads for the month of July were sought to be a fitness value for optimization. The reason for using this metric is that it will be conflicting with the maximized daylight illuminance that is sought to be satisfied simultaneously, especially for the specified hot climate with dominated cooling loads. Choosing those two conflicting fitness values is to look for the best tradeoffs between the reduction of cooling loads and maximization of a preferred threshold of daylight illuminance, using the search tool, the MOEA.

The experiment uses a hypothetical project site, and for the energy simulation and daylight analysis, the TMY3 format of the EnergyPlus weather file of College Station-Easterwood Field 722445, in the climate zone 2A (hot-humid) as identified by ASHRAE, was utilized. In addition to the weather data, energy simulation requires information such as assigning thermal zones, glazing attributes, analysis period, the configuration of the HVAC systems, and other information. The primary variable used for this simulation study was the WWR, and all the other building features and thermal properties were kept as default and template as configured in the Honeybee and Ladybug nodes. The building was considered one thermal zone for all the 4 masses that constitute the described model, and the entire building has been considered for the whole building energy simulation.

The following steps show the tasks needed for setting up energy simulation:

1. **Assign Program to Zones**. In terms of the early tasks needed for preparing building energy simulation, often, zoning becomes important. The building program was

determined to be an office building, and all the 4 masses were considered as an open office area.

2. **Convert Mass to Zone**. All the 4 masses were converted to one thermal zone, using the *Convert-Mass-to-Zone* node. The node requires inputs of the geometric masses and the zone program and gives Honeybee-based thermal zones as an output.
3. **Solve Adjacencies**. The adjacency amongst the 4 zones was solved using the *Solve Adjacencies* node in order to match the zone surfaces and their constructions considering the inner adjacent walls as interior surfaces with controlled thermal conditions as explained by the author of Honeybee and Ladybug nodes, Roudsari (2018). The nodes used for Steps 1-3 are illustrated in (Figure 4-24).

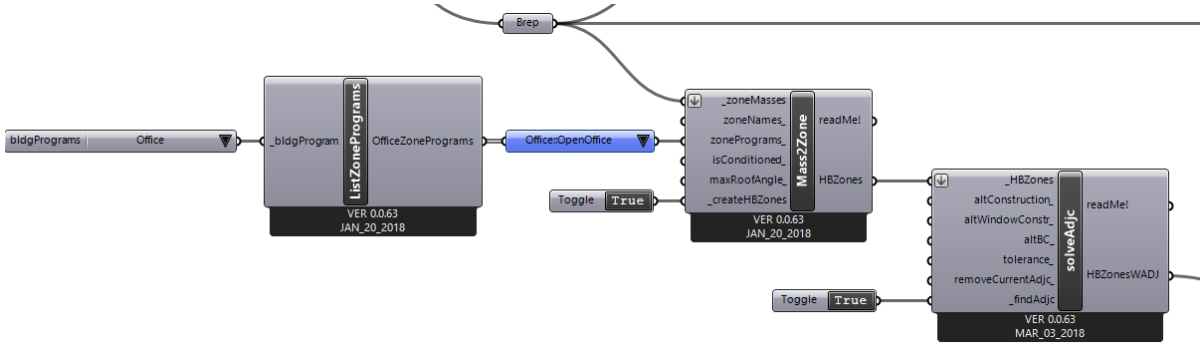

Figure 4-24. The first set of the Honeybee and Ladybug Plugins required for preparing energy simulation. From left to right: List-Zone-Programs, Mass-to-Zone, Solve-Adjacencies.

4. **Assign Glazing**. As described, the primary variable for energy simulation was determined to be the glazing ratio or WWR. Thus, for this task, the WWR variables drive the change of the building energy performance and daylight performance, in combination with the change in the layout dimensions. The list of glazing ratio parameters was then connected to the *Glazing-Creator* node, which requires the created thermal zones as well and assigns Radiance-based window materials to the

glazed surfaces. The attributes of the Radiance materials are described in the following *Preparation for Daylight Simulation* part. It is important to mention that by default, this *Glazing-Creator* node adds the glazing surface in an offset manner from the wall peripheries as one continuous surface to each wall of each façade, with respect to the façade orientation. This means that all north façade glazing surfaces, for instance, are of the same ratio. The Honeybee and Ladybug nodes for this task are illustrated in (Figure 4-25). In terms of the parameters and physical properties of the building envelope, they are shown in (Table 4-8).

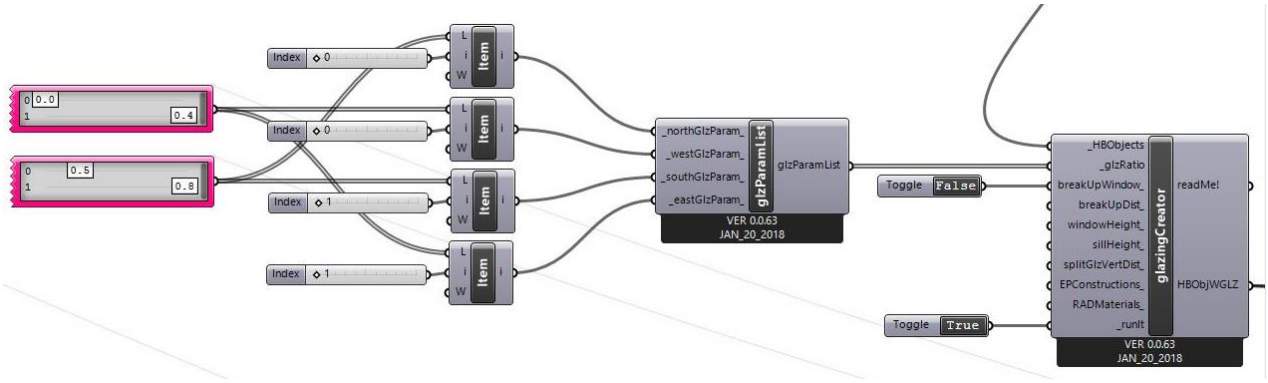

Figure 4-25. The second set of the Honeybee and Ladybug Plugins required for preparing energy simulation. From left to right: Parameters of WWR for each façade, Glazing-Parameter-List, Glazing-Creator.

Table 4-8. Model envelope thermal physical properties.

Opaque Materials	Reflectance	U-Factor with Film [W/m^{2}K]	U-Factor no Film [W/m^{2}K]
Exterior Wall	**0.3**	**0.429**	**0.459**
Interior Floor	**0.7**	**1.174**	**1.449**
Exterior Roof	**0.5**	**1.209**	**1.449**
Exterior Fenestration	**Glass U-Factor [W/m^{2}K]**	**Glass SHGC**	**Glass Visible Transmittance**
Exterior Window	**2.72**	**0.761**	**0.807**

5. **Add Simulation Time-step and Analysis Period**. In terms of energy simulation time-step, it was set up to be for the month of July with hourly simulation; thus, the analysis period is between the 1st and the 31st of July. The rationale for using this analysis period is to test the behavior of the optimization run since satisfying the minimum energy cooling loads will be challenging, particularly for the month of July.
6. **Prepare the Energy Simulation Output**. To set up the energy simulation task that will run within the MOEA optimization, the primary node for running the energy simulation used was *Export-to-OpenStudio* that takes the above-mentioned assigned attributes as inputs and generates an IDF file (EnergyPlus-based simulation file format) as the output. This IDF file refers to the *input data file* that contains the data describing the building and HVAC system to be used for simulation (EnergyPlus, 2015).
7. **Prepare the Read Output and Retrieve the Cooling Loads Value**. The resulting IDF file is then connected to the *Read-EnergyPlus-Result* node that gives results such as total thermal loads, cooling loads, heating loads, electric light, and equipment loads. The cooling loads' value was considered to be the fitness value for minimization and connected to the objectives input in the MOEA-based optimization tool, Octopus. The plugins required for steps 5-7 are depicted in (Figure 4-26).

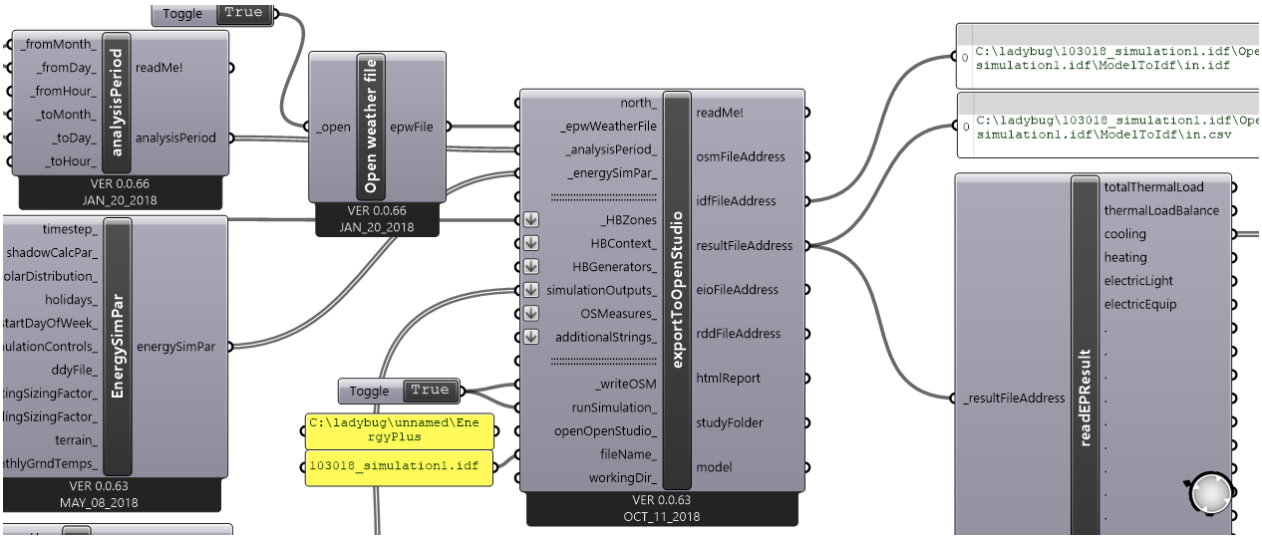

Figure 4-26. The third set of the Honeybee and Ladybug Plugins required for preparing energy simulation. From left to right: Analysis-Period, Energy-Sim-Par, Open-Weather-File, Export-to-Open-Studio, Read-EP-Result.

4.3.3.2. HVAC System Used for Simulation

The energy consumption calculation is performed by modeling the whole building simulation. The type of HVAC system used in this simulation was the Ideal Loads Air System, which is often used to study the building performance without detailed modeling of a full HVAC system (Simergy, 2013), to avoid complex input data, which can be less important in such an early design stage of this experiment. In this Ideal Loads Air System, the needed parameters are the zone controls, zone equipment configurations and the Ideal Loads system component, with no need to specify the parameters of the air loops and water loops. The system can operate with finite or infinite heating and cooling capacity, and for both cases, the designer can determine the schedules for on/off states of the heating and cooling, in addition to the outdoor air controls (Simergy, 2013).

As mentioned, the building model was simplified to a single thermal zone. The table for the HVAC components and their properties/variables for the system: Ideal Loads Air System used here can be found in the Simergy documentation (Simergy, 2013), based on the EnergyPlus

IDF Objects in the Input-Output Reference. In terms of schedules, the occupancy schedule of Medium Office Building was considered. The internal loads (people, lighting, infiltration, etc.) and heating/cooling setpoints comply with the requirements for office building specification. The compact schedules of Medium Office Building in EnergyPlus were used, and such schedules of office buildings can be found in the National Renewable Energy Laboratory (NREL) documentation (NREL, 2011).

For the energy simulation, the fitness function is calculated by hourly simulation of the whole building energy use for the month of July, and the cooling loads were retrieved and considered for minimization. The process of energy simulation using the above-mentioned tools starts with collecting the required information of geometry, building materials, and the weather data. The workflow includes generating the IDF file that can be opened and modified using the EnergyPlus engine. The modified file can then be used for further simulation tasks.

4.3.3.3. Preparation for Daylight Simulation

For preparing the daylight analysis, other tasks were required. Primarily, for daylight analysis purposes, defining building materials, generating a grid of daylight sensors, and assigning sky conditions were required, as well as other tasks for retrieving the analysis results and visualization purposes. Those tasks can be described as follows:

1. **Assign Radiance-material attributes for building components**. Materials for the interior building components: windows, walls, ceilings were customized. For windows, glass material was assigned using the Radiance-based material definition node *Rad-Glass-Material* with the transmittance values in (Table 4-9). Walls were described using the *Rad-Opaque-Material* node, which requires RGB reflectance values. Similarly, the same node was used for assigning the ceiling materials, with

higher reflectance values, as shown in (Table 4-9). This method of customizing the material properties is derived from the Radiance 5.1 Synthetic Imaging System developed by the Lawrence Berkeley National Laboratory that considers RGB radiance values to define materials (Radiance, 1997). The plugins used for assigning materials are shown in (Figure 4-27).

Table 4-9. Radiance-based building component material attributes.

Radiance Material Attributes			
	R Transmittance	**G Transmittance**	**B Transmittance**
Window Material	**0.4**	**0.4**	**0.4**
	R Reflectance	**G Reflectance**	**B Reflectance**
Wall Material	**0.75**	**0.75**	**0.75**
	R Reflectance	**G Reflectance**	**B Reflectance**
Ceiling Material	**0.9**	**0.9**	**0.9**

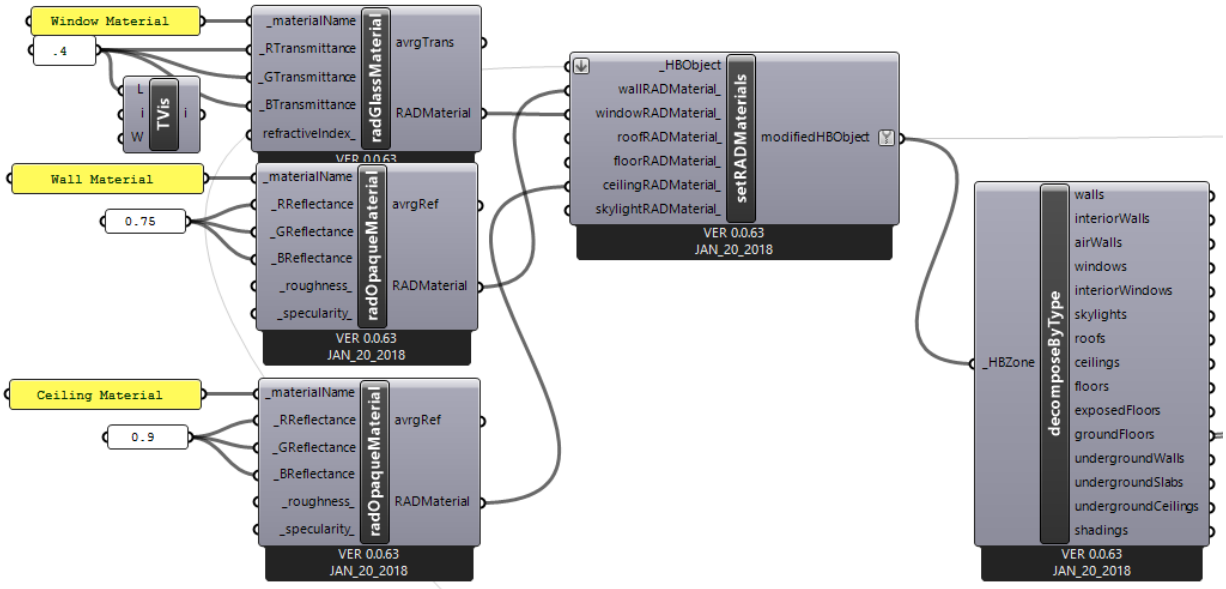

Figure 4-27. The first set of Honeybee and Ladybug Plugins required for daylight simulation. From left to right: 3 rows of Radiance-Material nodes for glass and opaque surfaces, Set-Rad-Materials, Decompose-by-Type.

2. **Create a grid of daylight sensors (Test-Points)**. The created Honeybee thermal zone was used to retrieve the parameterized surface area for adding the daylight

sensors or test-points for the daylight grid-based simulation pursued in this experiment. The zone was decomposed using *Decompose-by-Type* node, to get the building surfaces as separate components. The ground floor was retrieved from the node, and connected to a *gen-Test-Points* node, to create those test-points, which requires the grid-size and distance from the selected surface (ground floor). The grid-size or spacing used was 1 meter apart, and the distance between the location of the test-points and the ground surface was determined to be the default desk-level of 0.76 meters. The height of those test-points determines the daylight grid mesh that can be used to visualize the level of illuminance as a color-coded grid where each cell's center-point is the daylight test-point.

3. **Assign Sky Conditions**. The sky conditions are important to assign for daylight analysis. Thus, *gen-Standard-CIE-Sky* was the node used for generating the sky. This node requires the weather file, the time of the day, and the type of sky to be determined for the analysis. The daylight simulation time was chosen to be one hour at 9 am under equinox. This assumption is often pursued in such grid-based daylight simulation. The sky type used was: sunny with the sun to test the effect of heat generated by such sky conditions on the building energy performance, including the cooling loads. The nodes for Steps 2 and 3 are depicted in (Figure 4-28).

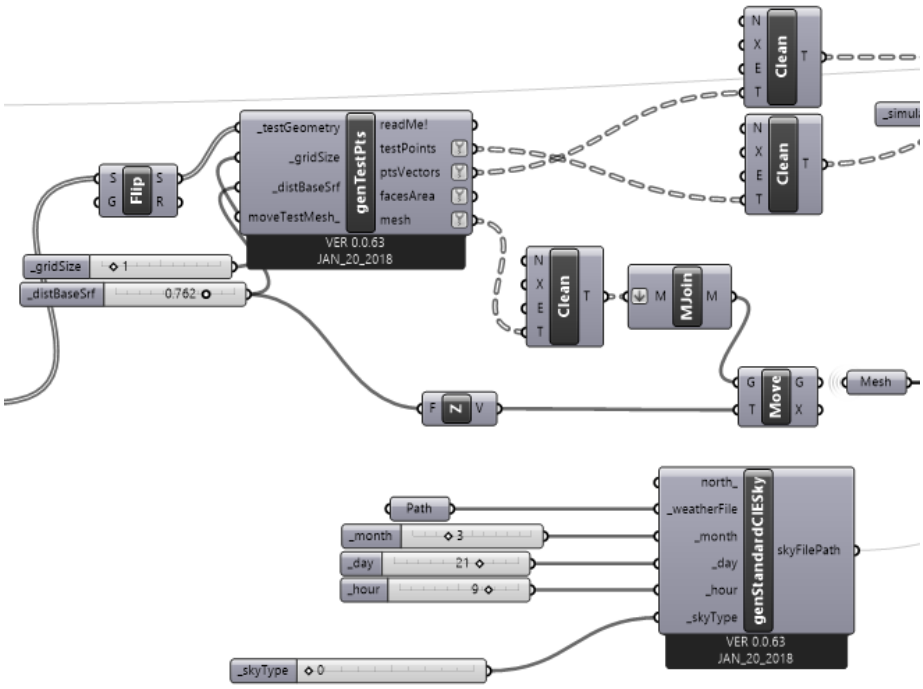

Figure 4-28. The second set of Honeybee and Ladybug Plugins required for preparing daylight simulation. From left to right: Generate-Test-Points, Gen-Standard-CIE-Sky.

4. **Add a Reflective Exterior Surface**. Another important component for accurate daylight simulation is to add an exterior reflective surface that would bounce the indirect light to the building. This was done by retrieving the mass model and creating a ground surface in front of it.
5. **Prepare the Grid-Based Illuminance Simulation**. In particular, an additional task is needed for the selected grid-based daylight simulation, which is preparing the parameters for the *Grid-Based-Simulation* node. This step requires the above-mentioned parameters and attributes of the sky, grid of test-points, their vectors pointing up, and the created mesh to illustrate the daylight illuminance values.
6. **Setup the node of *Run-Daylight-Analysis***. For this simulation, the primary node is the *Run-Daylight-Analysis*, which takes the analysis recipe, the Honeybee objects

(zones) of the building along with the exterior reflective surface, and the path and name of the file that will be written when the daylight simulation is initiated, in addition to other customizable parameters. Nodes of Tasks 5 and 6 are illustrated in (Figure 4-29).

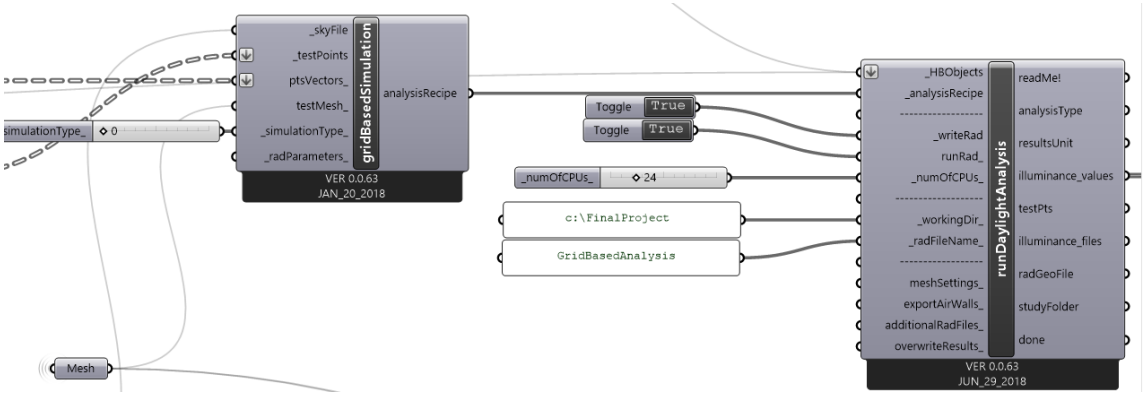

Figure 4-29. The third set of Honeybee and Ladybug Plugins required for preparing daylight simulation. From left to right: Grid-Based Simulation, Run-Daylight-Analysis.

7. **Add a Colored Mesh for Visualizing Daylight Analysis**. For visualization of the illuminance values at each test-point, a colored mesh was used. A legend can be appended to the mesh to show the illuminance values and their representative colors indicated in the colored mesh.

8. **Daylight Illuminance Values**. In terms of the daylight fitness value, at first, LEED v4 spatial Daylight Anatomy (sDA), a metric for illuminance compliance, was thought to be satisfied. This requires achieving illuminance levels (300–3000 lux) for either a ratio of 75% of floor area (2 points) or 90% (3 points) for 9 a.m. and 3 p.m. at the equinox (USGBC 2014). However, to reduce extended simulation time, the measure was simplified to grid-based daylight simulation of one hour at 9 am on

March 21st for a grid of sensors 1 meter apart. The daylight illuminance fitness function was calculated as follows (and sought to be minimized):

$$-\textbf{\textit{Illuminance Ratio}} = -(\textbf{\textit{Room area of illuminance between}}\ \mathbf{300-3000}\ \textbf{\textit{lux/Total room area}})$$

For the task of daylight simulation, the building geometry information, the opaque materials' reflectivity, and the glazing properties are collected, and for each design candidate, the grid-based daylight illuminance simulation is conducted.

Overall, the two objective functions of this experiment can be expressed as follows:

$$E_{obj} = \min(JCL)$$

$$D_{obj} = \max(DIR)$$

where

E_{obj} = Energy Performance Objective Function

D_{obj} = Daylight Performance Objective Function

JCL = July Cooling Loads

DIR = Daylight Illimunance Ratio (within the range)

4.3.3.4. Optimization Run and Discussion of Results

For the optimization run, the 8 parameters were connected to the Genotypes, the mesh of the initial mass model was linked to the Phenotypes, and the fitness function values were linked to the Objectives in Octopus. Running the tool, a random design population was initiated, and for every design option in the objective space, daylight and energy analyses were conducted. The MOEA run searches for fitter solutions; while daylight and energy performance are improved, design options evolve and change in composition, and the fenestration openings change in ratio to satisfy the two fitness values simultaneously. It is also relevant to note here that the

performance evaluation for each design candidate required 80 seconds for both energy simulation and daylight illuminance analysis. At each iteration (or generation run), a pool of 191 candidates is evaluated for 100 Pareto front and Elite solutions. Therefore, running each generation (iteration) required (240 minutes or 4 hours).

Optimization was terminated at generation 20. For demonstration and discussion purposes of the optimization run, and more importantly for applying the clustering method, the design solutions of generation 20 were used; in particular, the Pareto and Elite solutions (100 solutions) were selected to be recorded and reinstated into Grasshopper, as shown in (Figure 4-30). The Elite set refers to the best-fit solutions that are guaranteed as parent candidates in the next generations; when a new solution set emerges, it is compared to the worst of the elite set, and only if the new solution is better it will replace the worst Elite solution (Musnjak & Golub, 2004). The search space shows the Pareto and Elite solutions as red dots, and the two objective function values in the 2D space in (Figure 4-30). It can be seen from the search space that the solutions started to move towards defining a Pareto front at generation 20 although still not well defined; yet, the results were thought to be satisfactory, and the design solutions are considered worthy of discussion. Those Pareto and Elite design solutions were retrieved and reinstated from the Octopus tool interface to the Grasshopper environment with their parameters and fitness values, and geometric characteristics. As a sample, four solutions (Solution 1, 34, 67, 100) were selected, labeled and visualized for discussing the optimization results as depicted in (Figure 4-30). All 100 solutions were subjected to the application of the clustering algorithmic definition.

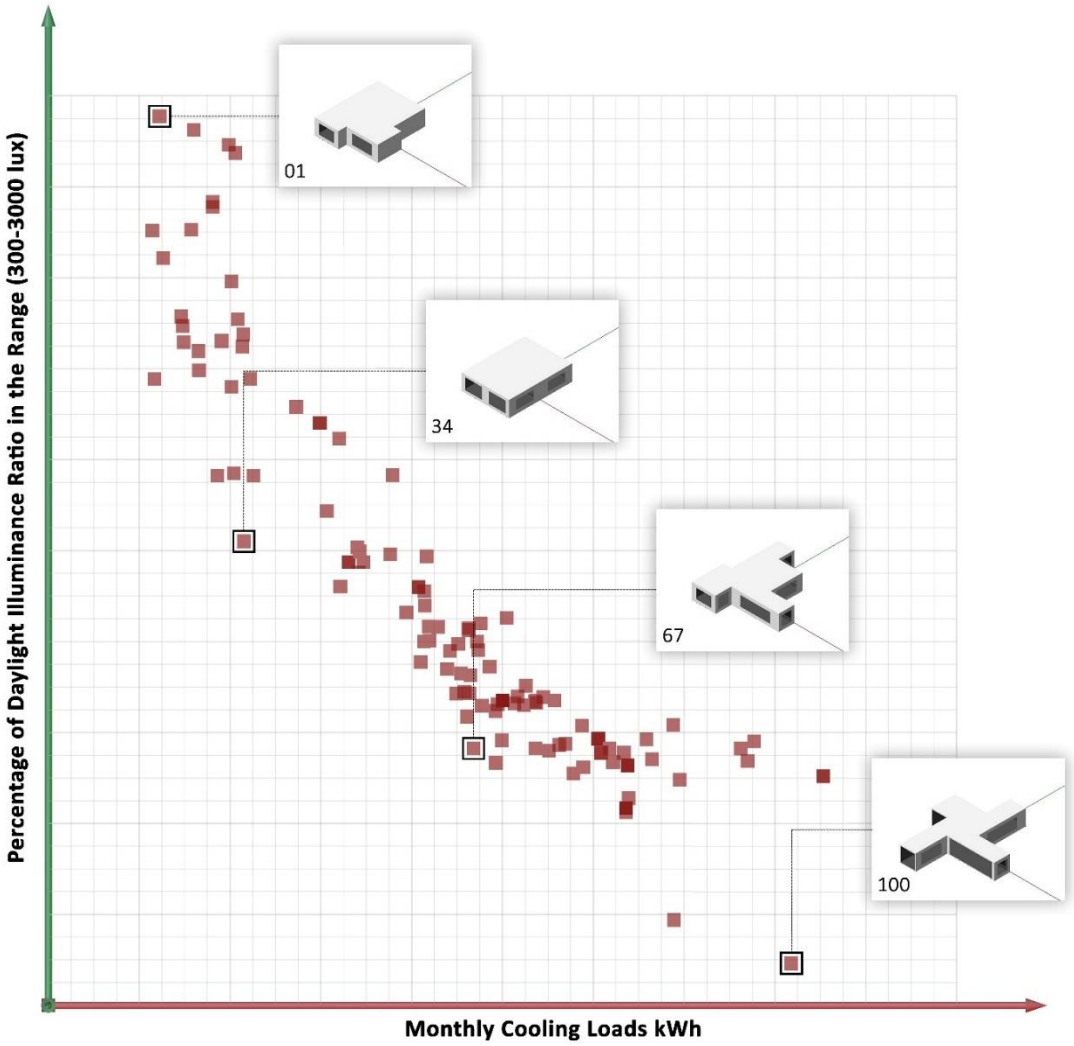

Figure 4-30. The Pareto front and Elite solutions (total 100 solutions) at generations 20, in regards to the monthly cooling loads (x-axis) and the daylight illuminance ratio (y-axis) as objective functions. Sample solution #1, #34, #67, and #100 are labeled in the figure.

The four solutions depicted in (Figure 4-30), have been enlarged and visualized in (Figure 4-31) in which for each solution from left to right the following views are shown: a top view that shows the grid-based pattern, a 3D view that illustrates the generated shape and openings, and a grid-based daylight mesh)a color-coded according to the illuminance values at each sensor). In addition, for each design, the shape ID number, the energy consumption for the July cooling loads, and the illuminance ratio value are presented in (Figure 4-31). It is important

for the discussion to state the expected behavior of the Pareto front method in MOO. The solutions with the closest distance to the minimum of each objective function, the closest to the origin point in the 2D search space shown in (Figure 4-30), will satisfy those objectives the best. For instance, Solution 1 is one of the best solutions to satisfy minimum cooling loads due to its closest distance to the origin in respect to the energy use objective function, yet it is one of the poorly performing solutions in terms of illuminance ratio because it is one of the farthest to the daylight illuminance ratio objective function's origin point.

As expected, Shape #1 in (Figure 4-31), that is located on the top left corner in (Figure 4-30), adheres to the minimum energy cooling loads with 30,936.58 kWh. However, it represents one of the lowest illuminance ratios within the range with 6%. The reason for such a behavior is that it is a compact configuration that avoids windows, with few surfaces exposed to external thermal conditions, which makes it efficient in terms of minimum cooling loads. Yet, less external surfaces mean fewer opportunities for windows, and the lack of windows on the east façade, in addition to the comparatively lower WWRs for the west facade, all causes the solution to behave poorly in satisfying the preferred illuminance ratio. Such a result illustrates how conflicting the two objectives of energy and daylight are. It is relevant to emphasize that for the illuminance ratio calculation, the values above 3000 lux were not preferred as they might cause glare issues in the interior space. Therefore, the preferred range between 300 lux and 3000 lux was sought for maximization, and the higher or/and lower values to this range were not considered in the ratio calculation. This explains the low illuminance ratios noticed in these design solutions discussed here and makes satisfying the preferred illuminance ratio challenging, and necessitating more iterations.

For Solution #34 (Figure 4-31), it is noticed that the daylight illuminance objective has been improved with 22% illuminance ratio despite that the configuration is still compact, but with higher WWRs, compared to Solution #1, and all façades contain windows. Nevertheless, this solution is worse in terms of energy use (36,748.08 kWh) as more cooling loads are needed for the increased glazing surfaces. In solution #67, the configuration becomes less compact, with more extensions in the four zones, leading to larger surfaces exposed to exterior thermal conditions. The has led to larger glazed surfaces and a higher illuminance ratio of 29%. In response, the cooling loads are higher here, with 43,967.35 kWh. Finally, Solution #100 represents, with its fan shape, an extended configuration of the 4 zones with a higher number of windows, and became one of the best solutions in illuminance ratio, with 41%. Yet, it is one of the worst solutions in regards to the cooling loads, with 44,555.67 kWh. Overall, the optimization results illustrate an expected response to satisfy the two conflicting criteria and show consistency with the predicted optimization results, which verifies that the optimization mechanism was working successfully.

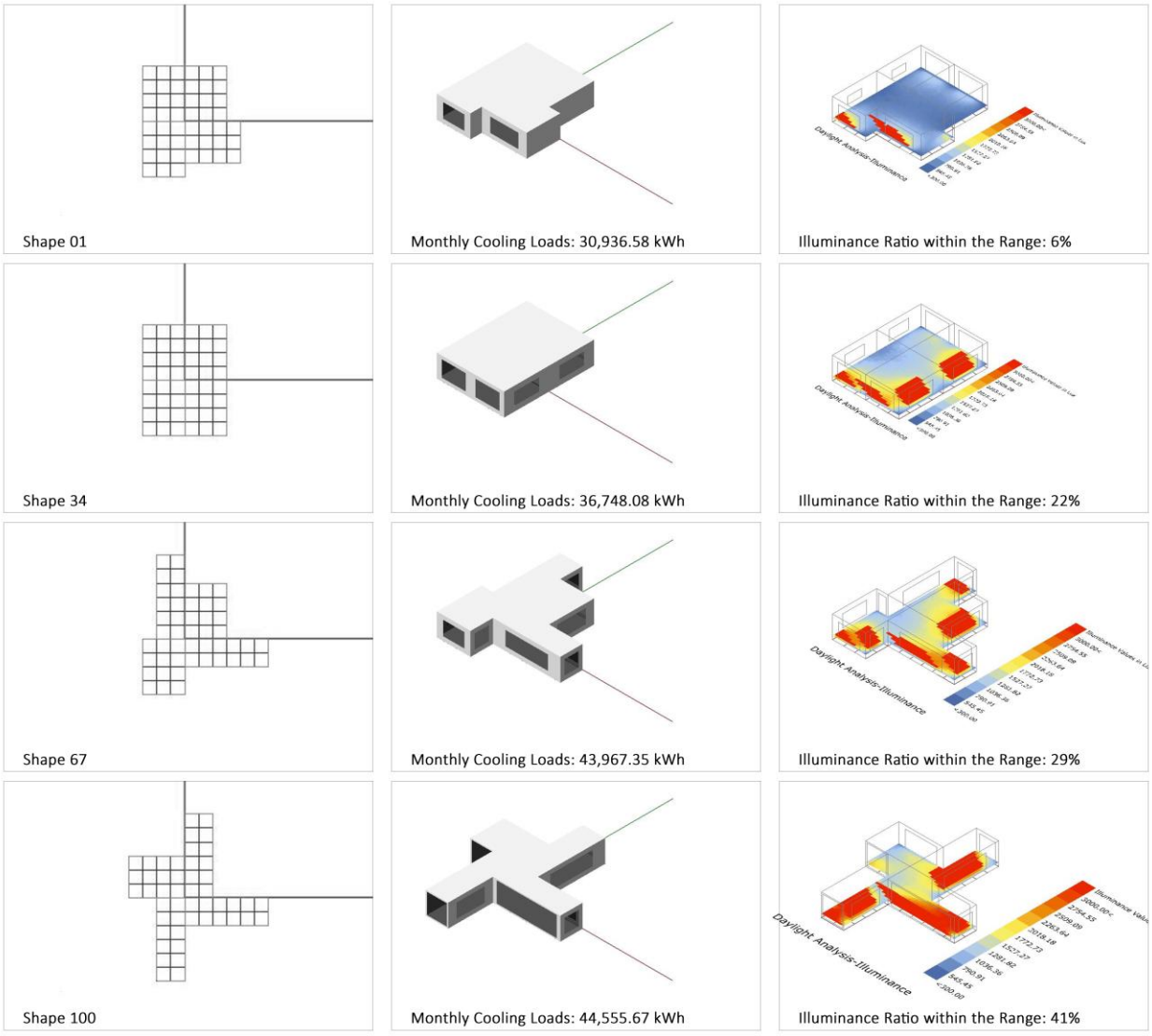

Figure 4-31. Patterns of each of the four selected solutions and its energy use in terms of monthly cooling loads, and the daylight illuminance performance (left: top view, center: 3D, right: top view of the daylight illuminance mesh).

Next, the 100 shapes, with their 48 cells each, were stored in a list, as a preparation step for applying the clustering method. (Figure 4-32) shows the top-view of the 100 shapes of the Pareto and Elite solutions.

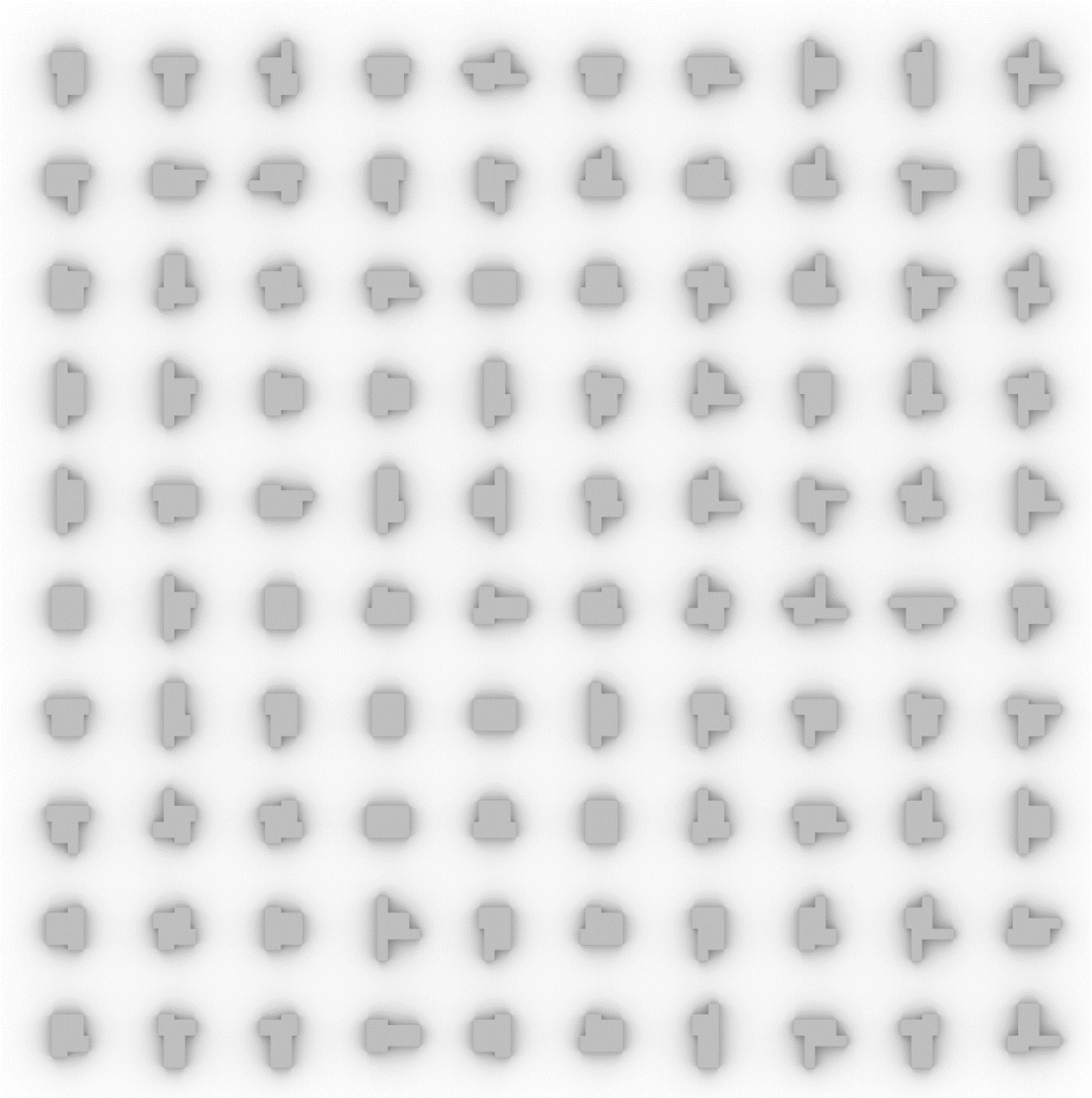

Figure 4-32. Shapes of the 100 Pareto front and Elite design solutions generated at generation 20.

4.3.4. Applying the Clustering Method

As in Test-cases 1 and 2, the 100 shapes and their cells' center-points were needed as the main input for the clustering method. At first, the "Pair-wise Shape Difference and the Hungarian Algorithm" was pursued with the list of 100 shapes to the input parameter. Similar to Test-case 2, the Batch-run component was also needed here, as extended computation time was

expected for calculating the pair-wise shape difference. Using the Batch-run algorithm, two parameters' sliders that represent the 100 shapes, each indexed (0-99), cross-reference shape difference calculation was carried out, running one pair-wise shape difference analysis at a time. This has led to (100*99/2=4950) calculation steps for the shape difference calculation. Each step took 20 seconds of computation time, totally requiring 27 hours and 30 minutes. However, the K-Medoids clustering task requires less than 10 seconds for this experiment.

4.3.5. Experimental Clustering Outcome

Before discussing the outcomes of clustering, it is significant to mention that the shapes here are generated from the Pareto optimization process, which is different from the shape samples produced in Test-case 1 that have been generated through manual changes of the parametric setup, and also different from the 72 shapes in Test-case 2 modeled after a set synthesized in another study. To test the clustering results, two scenarios were followed. In the first scenario, the input to the number of clustering was 5, generating 5 clusters as color-coded and depicted in (Figure 4-33). In the other scenario, 10 clusters were generated, leading to different clusterings as illustrated in 10 colors in (Figure 4-34). It is expected that for such a variant large set of shapes, choosing 5 clusters might not lead to preferred results, as probably a higher number like 10 can capture more coherent clusters.

In Scenario 1 of 5 clusters, apparently, perceptual coherence can be seen in a way that shapes of similar geometric features and overall proportions are clustered together. Also, the location of the extruded parts of the shapes that are most similar (top-left, top-right, bottom-right, or bottom-left) becomes important to determine the clustering. The first 4 clusters are of two rows as their member sizes are relatively large, while the fifth cluster set is of one row. Cluster 1 in red is comprised of 16 shapes with the wide T-Shape as the medoid. It can be noticed that the

shapes share similar characteristics in terms of similarity to the T-Shape or almost T-Shape in most cases, with variances. Also noticeable is that there are three identical T-Shapes clustered within this set (with different window configurations), which confirms that the clustering algorithm is functioning correctly. This membership of repeated shapes together in one cluster can be also seen in the other clusters.

In Cluster 2 in orange, the 30 shapes represent a family of larger height to width ratio in terms of proportions, compared to Cluster 1. The circled medoid clearly represents this characteristic of a vertical arrangement, almost a vertical rectangle. On the opposite, the 21 yellow shapes in Cluster 3 are of horizontal orientation. It is apparent that in this cluster, the medoid approximately represents a horizontal rectangle. Also, 2 mirrored T-Shapes exist, opposite in direction to the T-Shape medoid in Cluster 1. However, within the cluster, a left rotated T-Shape and two right T-Shapes also exist, yet with similar proportions to the horizontal rectangles that are clustered within, making it more of a horizontal rectangle cluster. In Cluster 4 in green, another typified shape, the left T-Shape is the medoid. The 20 shapes share some similar characteristics to the left T-Shape, yet with changing in projected and recessed parts. However, some outliers can be observed, when the shapes become less compact and different from the medoid. This can be explained as there are variant shapes that exceed the 5 clusters given here, and thus somehow some variant shapes of similar proportions are grouped in one cluster. Cluster 5 with its 13 members in cyan color, shows the most homogeneity and perceptual coherence, with almost all the shapes of similar verticality, represented in the dominant rotated left T-Shape, yet with other cluster members with slight to moderate variations. Overall, for the first four clusters, there is not just one dominant typified shape like in Cluster 5.

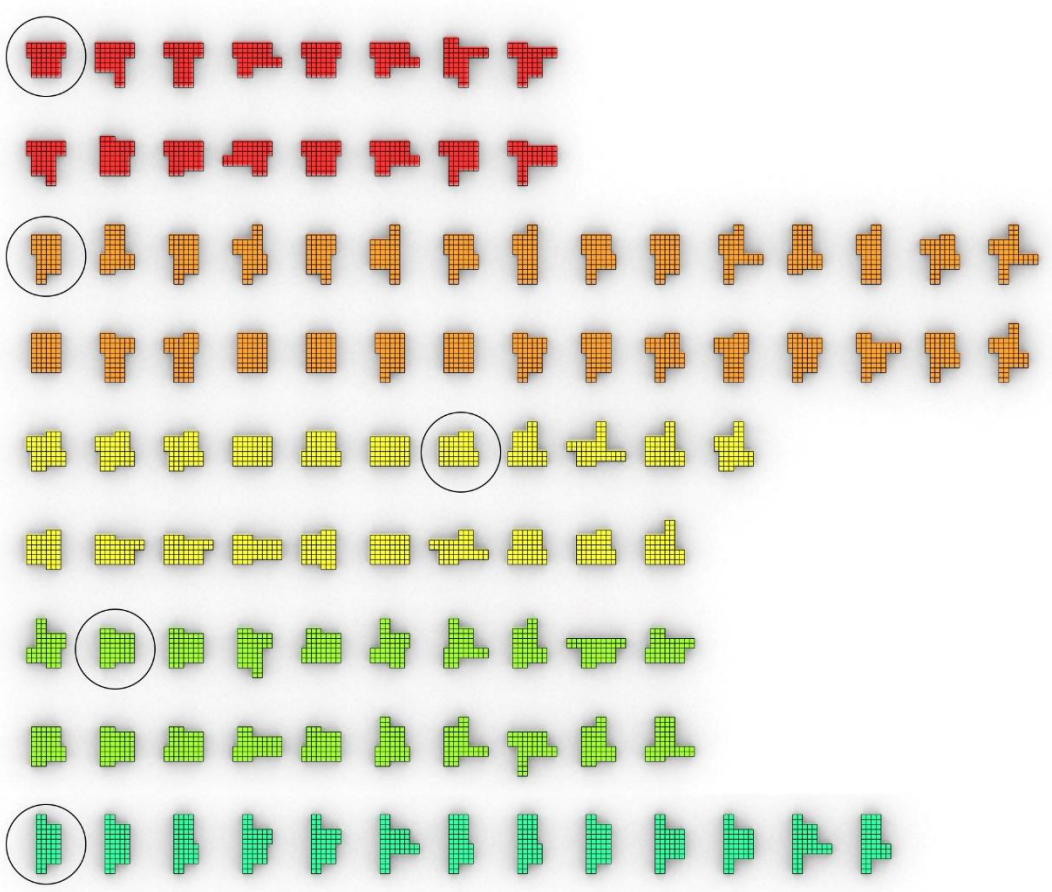

Figure 4-33. Results of clustering using 5 clusters, with each cluster represented in a different color.

In Scenario 2 (Figure 4-34), the clustering subsets show better results compared to Scenario 1, as more homogeneity and increased perceptual coherence were attained here. Cluster 1 in red, with the medoid close to a T-Shape, contains 11 shapes most of which are similar to T-Shapes, with certain similar proportions. The cluster contains 4 obvious T-Shapes, 3 of which are identical wide T-Shapes, and one T-Shape is slightly more vertical. It can be labeled as a T-Shape cluster. The shapes of Cluster 2 in orange, can be considered as a family of 18 vertical rectangles with other shapes of similar verticality and proportions, clearly of a high height/width ratio. Four redundant perfect rectangles can represent the characteristics of this subset. The yellow Cluster 3, on the other hand, can be considered as a horizontal rectangle family of shapes.

The 10 shapes share low height to width ratio, and three repeated perfect horizontal rectangles are contained within the group. Cluster 4, with its medoid as a rotated left T-Shape, contains 11 similar green shapes, close to the medoid's T-Shape and with similar proportions, yet with slight to moderate variations. In Cluster 5 in cyan, the 10 shapes share proportions of verticality and can be considered as mirrored T-Shapes. The medoid is not perfectly a mirrored T-Shape, yet the other shapes indicate approximation to this characteristic.

The 4 light blue shapes in Cluster 6 show a fragmented arrangement as a pattern, with right and bottom arms clearly projected. It can be considered a clearly homogenous and perceptually coherent group. Cluster 7 with its 4 members in dark blue shows another pattern with one upper arm projected, and almost mirrored T-Shapes with one clearly rotated right T-Shape. The purple shapes in Cluster 8 show somehow compact yet of a fan shape pattern, represented in the medoid. Two wide mirrored T-Shapes and two rotated right T-Shapes are clustered within the subset, due to the similarity in geometry. Similar to Scenario 1, there is a homogeneous grouping of rotated left T-Shapes of large height to width proportions, represented in Cluster 9 and its 13 pink members. Finally, Cluster 10 in grey shows 4 obvious fan-shape-dominance with projected arms and distributed configurations making it a coherent subset.

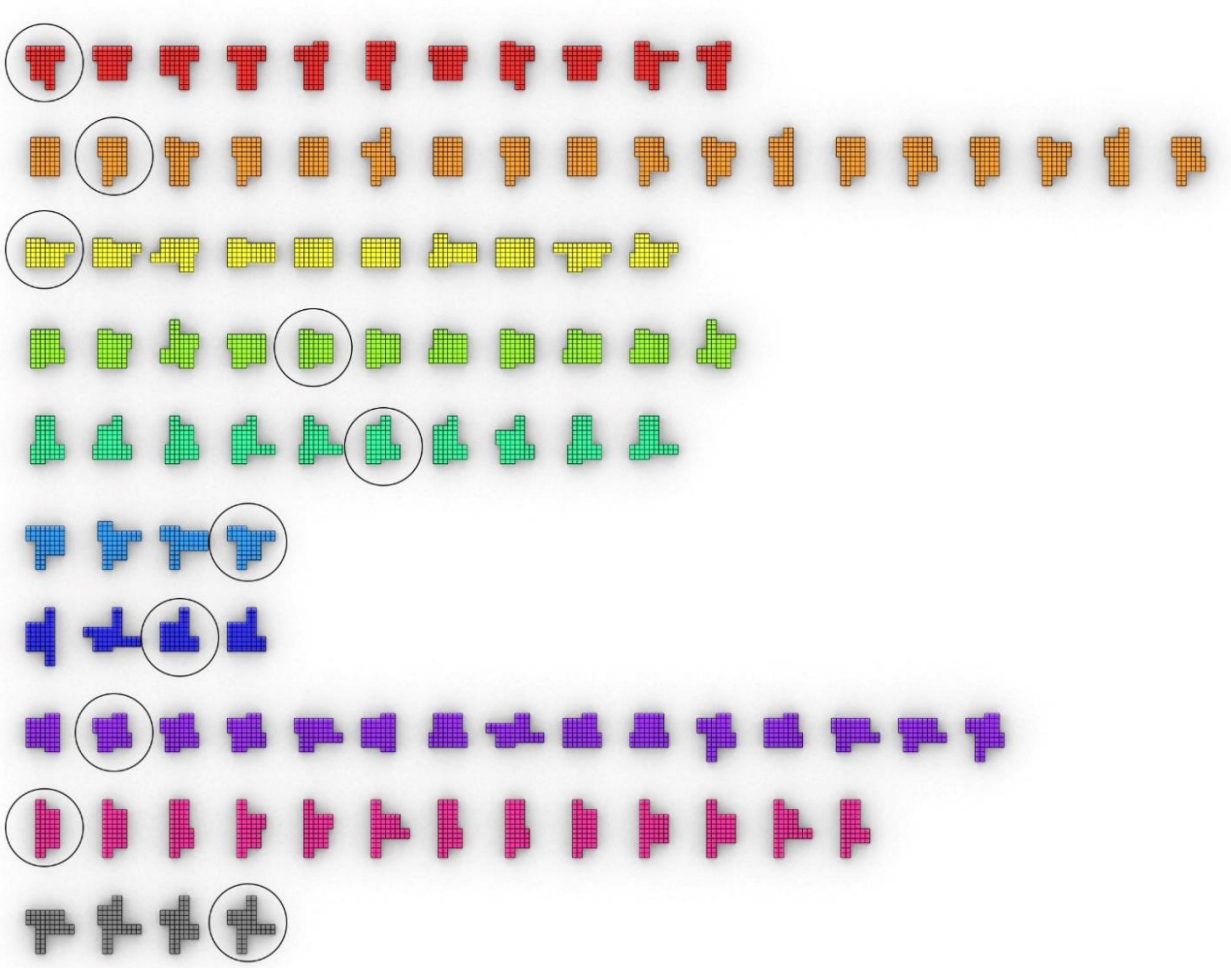

Figure 4-34. Results of clustering using 10 clusters, with each cluster represented in a different color.

Overall, the results of Scenario 2 with its 10 clusters start to define typified dominant shapes, thus show better perceptual coherence in comparison with Scenario 1. This can be due to the relatively large set, the 100 shapes, with multiple variations of shapes which required 10 as a number of clusters for the clustering algorithm to perform better. Another assertion can be drawn from the results, which is consistent with the grid-based descriptor's characteristics found in research, is that proportions are of high importance to determine shape similarity and difference. In addition, the results show that another factor, the shape characteristics in terms of

configuration or typological properties are also important. As noticed, particularly in Scenario 2, the typified shapes such as L-Shapes or fan-shapes have determined the cluster pattern.

4.3.6. Validation Study

In this experimental test-case, a demonstration of incorporating the clustering method into optimization was targeted, to validate the applicability of the system in a fully working prototype. The significance of this experiment is that based on the results, there is a valid need for the clustering method to organize the generated design solutions according to their shape similarity/difference. The shapes of the resulted design set are variant and cannot be visually articulated; application of the clustering algorithmic method has led to shape cohesion, evident through examining the clustering subsets. Therefore, this test-case represents a validation case of the successful use of the SC-KM method into the performance-driven generative design and optimization frameworks.

In terms of evaluating the clustering results, there was no preexisting reference set, thus the clustering evaluation metrics were not pursued. It was primarily perceptual coherence that was considered as a visual measure of clustering consistency. The trend of a cluster to show a dominant typified shape represents the cluster's perceptual coherence, which has been retrieved, more evidently in Scenario 2. Also, all repeated (identical) shapes were clustered together, with no exception, which proves that the clustering method worked successfully. Internal validation in Test-case 1 shows the accuracy of clustering results, and external validation of the clustering method has been already performed in Test-case 2. Therefore, in this experiment, the focus was less on clustering evaluation but more on incorporating the clustering method into optimization.

In terms of the randomness of clustering in this experiment, similar to Test-case 2, the impact of a random function in initiating the medoids was evident in the somewhat different

resulting clusters that emerge when re-running the K-Medoids algorithm. The reason for the randomness despite increasing the number of iterations to 1,000,000 in K-Medoids is that the number of combinations of selecting 10 random medoids out of 100 shapes is excessive. The number can be calculated as: (100x99x98x97x96x95x94x93x92x91, or 100! /90! = 6.2815651e+19), that is much larger than the number of iterations. Simply, there are many different possible combinations not only in the first state of assigning the medoids but also for later steps, when switching the medoids with non-medoids. The K-Medoids algorithm can only test part of the combinatory possibilities to get relatively accurate results, and the search space is a really big space for finding the best medoids and the clusters. At each search step, there is a test to one or more of the possibilities, and even within a million or billion iterations, it is difficult to get convergence of clustering results. Despite this randomness, from architectural perspective, random yet reasonable results of clustering may be actually desired in the creative design process.

4.3.7. Discussion and Conclusions

The experiment was designed to further test the SC-KM method, incorporating it to a workflow that includes building performance simulation and an MOEA run, with multiple preparation tasks and functionalities needed. To the best of the author's knowledge, this is the first time, a clustering method is integrated into an optimization process within a generative system. This means that the clustering method has been tested within the framework of optimization and proved useful as a shape articulation strategy.

The generated population of 100 Pareto front and Elite solutions at generation 20 shows an expected behavior of the design solutions to respond to the two determined fitness values and proves consistency with typical MOEA-based optimization results. Importantly, the July cooling

loads resulted were validated in a hand-worked example, utilizing a typical energy use for an office building for the same area and climate conditions in Texas. The experiment results show consistency with such examples.

The clustering results show an overall shape cohesion, with the cluster subsets attaining coherence, more clearly in the 10-clusters scenario. In terms of the impact of randomness, similar to Test-case 2, due to the random function embedded in initiating the first medoids, there are different clustering results generated in multiple runs of the K-Medoids algorithm. Yet, the results are all reasonable and consistent. Compared to Test-case 2, the clustering subsets of this experiment are perceived better, and more homogeneous. The reason can be due to the packing method used in the previous case, in which an adjustment to the number of cells to achieve the same packed cells for all shapes was needed, while here the exact number of cells (48) was made automatically, without the need of removing and adding cells.

4.4. Summary of the Section

The section is made of three parts, dedicated to demonstrating the three instances of the prototype. In one instance, the first case, the protocol consisted of parametric form generation, parametric change, and developing the clustering method. In particular, the methods and algorithms used, and the workflow of developing the SC-KM method, applied to samples of shapes, were explained. The second part offers a detailed explanation of another case in which testing the developed method was carried out for a new set of shapes, with analysis of the clustering results using multiple evaluation metrics. The experiment includes an external validation study. The last part describes a third instance of the porotype, with coupling the SC-KM method with performance evaluation and optimization methods.

5. CONCLUSIONS AND FUTURE WORK

This section provides remarks on concluding concepts on the developed Shape Clustering K-Medoids (SC-KM) method and the proposed generative system, along with their expected applications, limitations, and future work. The study produced a prototype for a new GDS that couples two routine processes of parametric form generation and design optimization, incorporating a third innovative process of shape clustering and design presentation. This new shape clustering method was developed to articulate a dataset of shapes into groups of similar ones, formulated using two main functionalities: (1) a pair-wise shape comparison using a grid-based shape representation for retrieving shape similarities/differences, and (2) implementation of the K-Medoids clustering algorithm. Demonstration and application of the prototype were carried out through three test-cases, of which each experiment was conducted for specific purposes, with modified functionalities and added components, leading to a fully working prototype in Test-case 3. Within each of the test-cases, validation was conducted, and particularly, external validation was carried out in Test-case 2.

5.1. Concluding Points on Experimental Test-cases

In carrying out the first experiment, the aim was to develop the shape clustering method and test its performance. The algorithm performed as expected, successfully articulating the sample shapes into groups of similar shapes, and the representative shapes were attained. The samples of shapes were predetermined as reference sets to evaluate the clusterings. The results of this experiment proved correct in clustering all the given samples accurately, in regards to the expected cluster-sets and provided enough assurance for the clustering method to be applied to other test-cases. Collectively, Test-case 1 has led to the successful development of the proposed

SC-KM and offered insight to limitations and potential improvements of the formulated algorithmic package.

In Test-case 2, the clustering method was applied to a new dataset of shapes for further testing. To apply the method to other shapes, one task is needed beforehand, which is to use a packing algorithm to convert boundary-based shapes into grid-based ones. A sample of 72 shapes was used, and a new algorithm for packing was formulated. In packing, two scenarios were pursued: Scenario 1 with 36 cell-packing and 2 with 64 cells. Also, the number of packed cells in each shape needed to be matched for all the shapes for the accuracy of shape comparison. Another objective was pursued as well, which was to conduct an extensive validation study in order to evaluate the clustering results. The clustering evaluation metrics resulted from the two scenarios showed improvement to the compared study. For better clustering results in our method, it is expected that the non-fixed aspect ratio method in the grid-based description is the next step. This modified grid-based description can reduce the impact of proportions as a determining factor of shape similarity/difference. In addition, in Test-case 2, the impact of randomness in initiating the medoids in K-Medoids clustering was observed, tested, and discussed.

For Test-case 3, the main purpose to carry out the experiment was to test the SC-KM inside a system with the building performance evaluation simulation and optimization process integrated. At first, parametrizing the initial model was directed to achieve more realistic building shapes (in compared to Test-case 1 with abstract shapes), defined by spatial arrangement and adjacency relations. The packed cells became less important for defining the shapes. Yet, it was obligatory to keep the number of packed cells exact for all the possible produced shapes. The MOEA tool was the search mechanism for fitter design solutions resulted

in a set of designs with certain illuminance ratios and cooling loads which proved the success of the optimization run to satisfy the two objectives. Overall, the clustering results showed consistency, when visually examined. A conclusion was drawn that the 10 cluster-scenario led to better results than the 5 cluster-scenario, for the variance of the 100 shapes. This leads to an assertion that defining the number of clusters should be considered as a variable to be changed and tested for better clustering results.

Using three experiments, the test-cases have led to satisfactory clustering results. The three test-cases represent a series of improvement procedures, testing and validation studies of the SC-KM, and the progress culminated in the complete prototype in Test-case 3.

5.2. Contributions to the Body of Knowledge

The aim of this research was to develop and explore mechanisms for making sense of the generated design set in terms of form/shape evaluation, advancing the existing generative protocols. The focus is on the idea that reviewing qualities of architectural shapes is essential for the design process and thus needs to be facilitated and integrated into the GDS frameworks. The underlying argument in this research is that architectural design is not merely focused on finding the optimal solution in terms of certain performance-related criteria. More importantly, for generative models, additional design criteria in regards to organizational mechanisms that facilitate design exploration and shape-related decision making, need to be incorporated. In addition, I argue that the designers and architects' agency should be maintained, and their design knowledge is significant to achieve successful production and exploration of design alternatives in GDSs. Additionally, human-computer interaction needs to be accommodated in such systems. This study's contributions are primarily the new comprehensive generative design system and

the innovative SC-KM method. The following sub-sections describe these primary research contributions.

5.2.1. A New Generative Design System

The study marks one of the early attempts to develop a working SC-KM method incorporated into a new and comprehensive GDS, demonstrated by a prototype that is general enough and applicable to a wide range of design problems. To the best of the author's knowledge, an extensive literature review has not revealed another existing attempt that brings together parametric modeling, environmental performance evaluation and optimization, and an automatic shape clustering method for achieving organized design presentation. Architectural design is an inevitably complex process with multiple design criteria that need to be addressed and satisfied. Aesthetics and shape features are important design aspects, and methods that support successful design presentation and review in GDSs need to be further investigated.

5.2.2. A New (SC-KM) Method

Experimenting with shape comparison analysis with the search for optimum overlap, finding a shape difference definition, and adopting a new strategy, the K-Medoids clustering that is often used for organizing data and revealing hidden patterns, were through to be innovative experimentations in this study. Developing the SC-KM involved a process of experimenting and testing to formulate the functionalities needed in terms of finding a reliable shape description method (the grid-based), a validated shape difference measure (distance-based), an algorithm that solves the assignment problem (the Hungarian algorithm), and a validated distance-based clustering method (K-Medoids). Improvements were continuously made to revise the algorithms, and guarantee the seamless connection between those tools and algorithms. More importantly, the method was applied to three tests, all led to successful clustering of the design options.

Implementing the K-Medoids algorithm has proved successful in combination with the grid-based shape description. As confirmed by the experiments in this study and other research, e.g. (Gullo et al., 2008), K-Medoids is robust and applicable to a wide range of datasets. One of the advantages of the clustering method developed in this work is that it does not require extensive computation time at each run (iteration) of the K-Medoids clustering. In fact, the "Pair-wise Shape Difference and the Hungarian Algorithm" algorithmic set is an independent GH_CPython node, separate from the K-Medoids algorithm. This means that computing the shape difference is performed only once, and inside the k-Medoids node, multiple clustering runs can be made with only seconds of computation time. This SC-KM method is applicable to a wide range of shapes. The algorithmic set of this method will be open for download as an accessible open-source package of algorithms for designers and architects in the near future.

Collectively, this method is applicable to the built and virtual environments, in the framework of architectural design, including speculation, education, and practice, as illustrated in (Figure 5-1). In this context, speculation refers to creative expression and imaginative design rather than real building design. The circle size in the figure represents the magnitude (scale) of the applicability of the SC-KM method and the new GDS at the intersection of the built and virtual environments with each of the speculation, education and practice endeavors. For instance, the application is expected to be most valuable for virtual and built environments in education. Next comes speculation, and last is practice, with less use of generative design systems is expected. However, this can change in the near future. The argument here is that we now have the capability to explore different geometry and shapes in parametric methods, so we will encounter more variable design shapes, in which the developed algorithm could be applied.

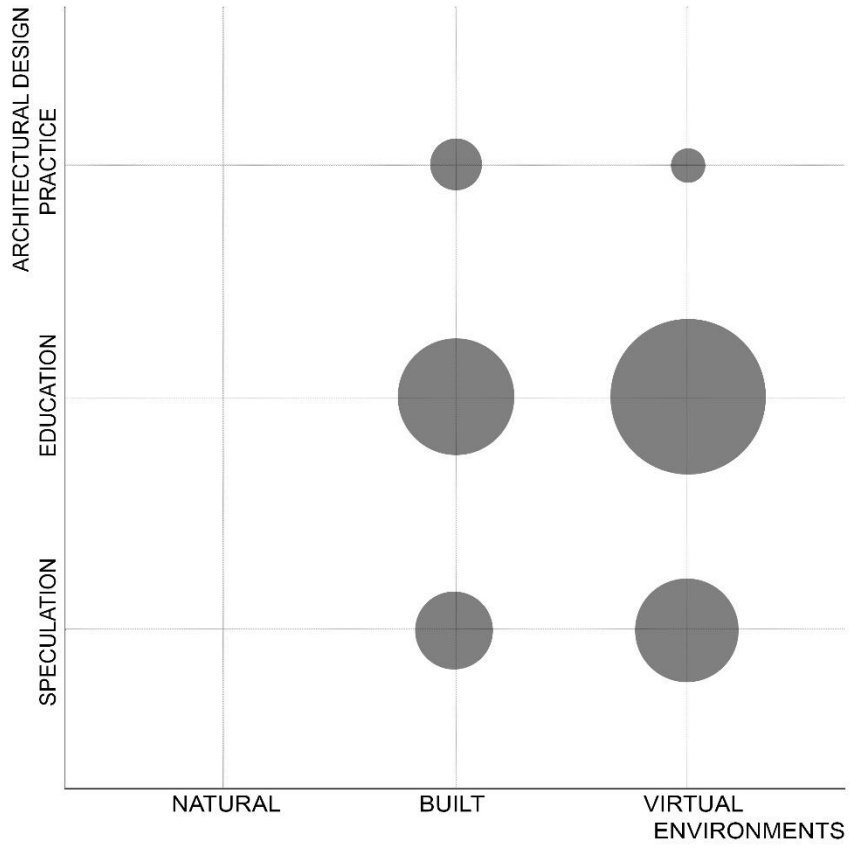

Figure 5-1. The application of the SC-KM in relation to architectural design (y-axis) and the three environments (x-axis).

5.3. The User Interface

The system's user interface described in Section 3 and applied in Section 4 is from the authors' standpoint of developing the system. The user interface is thus a complete and arranged package of plugins of algorithms with three test-case examples and variable input parameters, in addition to a visualization package to view the results. For the user, the process can be linear, which means it can be performed in a single iteration once the sample shapes are plugged in. An alternative possibility for the user is to follow alternation, moving between the design optimization, and the clustering and design presentation process. The system workflow was

designed to be performed automatically, without the need to configure or change the nodes of algorithms.

The options in which users make use of the SC-KM and its functionalities can be identified as follows:

- *Use of SC-KM as the final process*. The application of the SC-KM that has been carried out in Test-case 3 suggests the use of the method as the final phase of the GDS. In this case, the users can carry out their own parametric form generation, prepare the simulation tasks and run the optimization tool, and when terminating optimization, the emerged design set will be organized by the SC-KM. This requires first the form generation to include a packing algorithm to convert the shapes into grid-based and retrieve a nested list of each shape and its cells. Next, the pair-wise shape difference calculation would be conducted. The Batch-run node can be necessary for shapes with 36 cells and above, leading to an agglomerative calculation, and recording and storing the shape difference matrix become important. Finally, the K-Medoids clustering can be applied with a few seconds, and users can customize and change its parameters in terms of the number of clusters and iterations.
- *Use of SC-KM intermediately in the process.* The SC-KM can also be applied early or in an intermediate phase in the form finding process – this is expected but not tested in this research. The diverse clustered set can be updated to avoid certain shapes, for example, the clustering method can help remove certain shapes that are not preferred by designers. This filtering of only desired shapes will save computation time early in the process. It is not feasible for designers to foresee what forms will be generated in existing generative systems and to articulate the shapes; therefore, the clustering system can help with

articulating the preferred shapes. In the case of using an evolutionary algorithm as a generative tool, the users can pause the search and apply the SC-KM to organize and filter the representative shapes. Next, the representative set or the shapes of interest can be used for running new generative iterations focused on those representatives. In addition, in the application of clustering to optimization, alternating between optimization and examination of the optimized solutions, in a comprehensible way can be useful to eliminate unwanted shapes, and this reduces the computational burden of simulating the performances of such design candidates. A diagram of the user workflow is illustrated in (Figure 5-2).

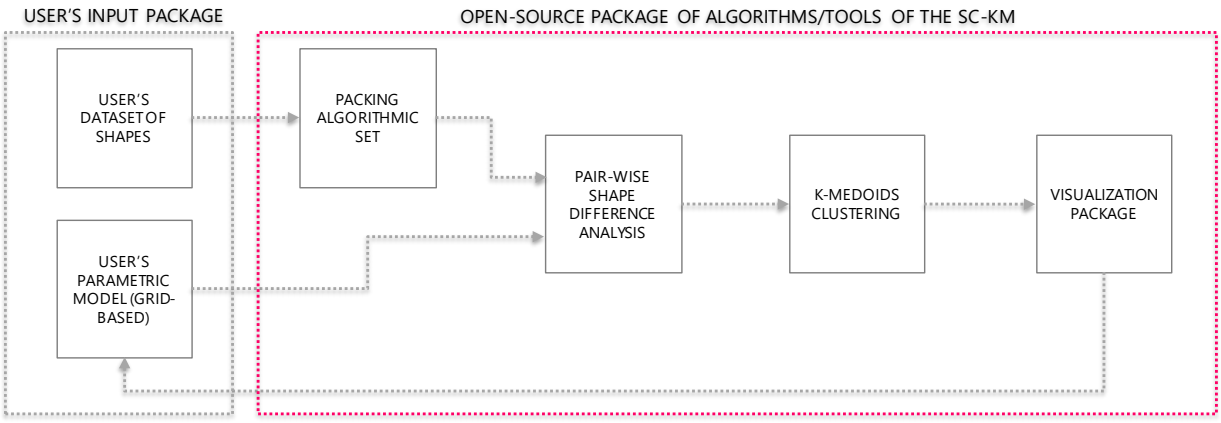

Figure 5-2. The workflow of the user interface.

5.4. Limitations in the Proposed SC-KM and the New GDS

Throughout developing and experimenting with the algorithms and tools of the SC-KM, some limitations have been identified. Since GDSs are widely used and experimented with by designers, the focus of discussing the limitations is on the SC-KM method. The primary limitations of SC-KM are described in the following sub-sections.

5.4.1. Required Computation Time

One of the limitations of the introduced clustering method is its computation time needed, particularly for running the pair-wise shape difference analysis. For low-resolution grids, like in Test-case 1, the total computation time needed for performing the clustering method was 5 minutes and 20 seconds for each run of both plugins, the shape difference calculation, and the K-Medoids clustering. In the experiment of Test-case 2, running the shape difference analysis for the 72 shapes with 36 packed cells took 7 hours and 6 minutes of computation time. In the second scenario, the pair-wise shape difference calculation took significantly longer computation time, with 56 hours and 48 minutes. For Test-case 3, the shape difference calculation took 27 hours and 30 minutes. This means that when increasing the grid resolution, the increase in computation burden was exponential. This extended computation is considered a limitation and requires improvement, which is planned in Future Work.

5.4.2. Randomness as a Limitation

The random function used in the K-Medoids algorithm that caused the multiple difference clustering results can be considered as a limitation although it still does lead to comparable and reasonable clustering. This relative accuracy is assured due to one of the K-Medoids clustering characteristics, its iterative process of switching medoids and non-medoids until the quality of the emerging clustering cannot be enhanced in any replacement. Such iteration is measured through a cost function of the average dissimilarity between the medoid and non-medoid objects (Han et al., 2011), which makes the results accurate in spite of the medoids' arbitrary initiation.

5.4.3. Orientation, Reflection, and Scaling of Clustered Shapes

Currently, the SC-KM method leads to shape clustering with invariance to scaling and translation, yet variance to rotation and reflection. In terms of human perception, it can be considered that rotated or/and reflected shapes are still the same shapes. However, human perception of shape similarity/difference is complex. Such psychological analysis of shape can be significantly affected by human experience, culture, and other factors. According to (Izard et al.), studies show that in regards to human perception of shapes, angles, and proportions of dimensions are considered defining features whereas orientations and positions are not as definitive (2011). In addition, "geometric concepts may emerge spontaneously on the basis of a universal experience with space or reflect intrinsic properties of the human mind" (Izard et al., 2011, p. 319). Human analysis of shape similarity and difference is an important area that can be researched in the future.

Overall, the method can be developed to lead to the invariance of rotation and reflection. However, the choice of considering such variance to both rotation and reflection is from an architectural standpoint; change in orientation leads to different design alternatives. This change can be also visually perceived by humans from the building environment where other buildings or environmental components exist in a context or background. In other words, from any fixed point in the building physical environment, the view of a shape (perspective) will be different from the view of the shape rotated or reflected (if not symmetric). Importantly, the functional difference between identical but rotated shapes is already considered in the objective (performance) space; rotated and reflected building shapes mean different energy consumption and different daylight metrics. In regards to the geometric difference analysis, the rotation and reflection operations can be included or excluded. When included, those procedures of rotation

and reflection can be incorporated as components into the algorithmic pair-wise shape comparison definition yet will add computational burden.

5.4.4. Inevitable Complexity of the Design Process

It is significant to note that the clustering method proposed in this study was aimed to support designers' exploration and evaluation of the architectural designs produced in generative models. It is to the author's awareness that the design process cannot be simplified to merely performance optimization, and this can be misleading and leads to ill-defined design problems. In architectural design, other related design aspects that are important were not addressed here. Simply, the complexity of the architectural design process is beyond the scope of this work; the study is limited to solve one problem of lack of geometric articulation in computational generative design and optimization systems.

5.5. Future Work

The direction of further development of the SC-KM can be threefold. The first direction is the improvement of the current algorithmic set of the SC-KM. Another aspect of further improvement involves the application of the method to three-dimensional forms. The third direction is the method's application to Machine Learning.

5.5.1. Improving the SC-KM Method

There are approaches to resolve the computing load problem in the future. In terms of shape comparison, it is expected that the elimination of the identical overlap cases will lead to saving in computation time. A different strategy, is to optimize the overlapping search, since we used an exhaustive method which was too costly, and has led to increasing the calculation time exponentially. The aim is to achieve a scalable solution with better computation time when the number of shapes or/and the number of packed cells increase. The search for overlap within the

shape comparison task may be facilitated by the use of Genetic Algorithms as a search mechanism. Another approach is to perform the shape comparison using cloud-computing.

5.5.2. Application of the Clustering Method to 3D Forms

One of the typical requirements of clustering is its capability to multiple or high dimensional data; a dataset may contain numerous attributes, and each attribute can be considered a dimension (Han et al., 2011). Most clustering methods behave well in handling two-dimensional to three-dimensional data objects; however, clustering datasets with high dimensionality is challenging (Han et al., 2011). In applying the SC-KM to 3D forms, there is a need to take into account the computation time needed, as it is expected to be exponential for adding the third dimension. More importantly, the algorithmic set needs to be updated allowing for considering the third dimension. Nevertheless, this aspect yields important improvement to the clustering method and is considered the next phase of its development.

5.5.3. Use of Machine Learning in the SC-KM

Observing the development of computational methods to support the design process, the recent advancement is around the computer's capability to be trained. As asserted by Steinfeld: "computers are being trained to see, and this new capacity matters to design" (2017, p. 592). Machine Learning (ML) is a sub-field in artificial intelligence that involves processes of knowledge discovery (Provost & Kohavi, 1998) relying on preexisting large datasets (Steinfeld, 2017). Generally defined as learning through observation, in ML, patterns are mapped in a way that allows the computer to learn through experience (Bechtel & Abrahamsen, 2002; DeLanda, 2002; Steinfeld, 2017) .

The primary aspect of Machine Learning is that it is not about programming the computer, rather, training it; the training can be for instance on classification, and the computer

is supposed to recognize images and classify them correctly using a large dataset each of which is labeled and this dataset is called “training set” (Steinfeld, 2017). The training set offers the computer the experience to learn and later to predict new dataset outside the one used for training (Steinfeld, 2017). It is expected that in applying the SC-KM into Machine Learning, the results of the current test-cases can be used as the training set and new grid-based shapes will be the input to the neural network that is trained. I predict that the neural network will cluster the new set similarly to the results of the SC-KM with a faster speed. However, there will be a significant amount of research to do for the feasibility and implementation of using Machine Learning in this field of study.

REFERENCES

Abdelrahman, M. M., & Toutou, A. M. Y. (2019). [ANT]: A Machine Learning Approach for Building Performance Simulation: Methods and Development. *The Academic Research Community Publication, 3*(1), 205-213.

Aish, R. (2003). Extensible Computational Design Tools for Exploratory Architecture. In B. Kolarevic (Ed.), *Architecture in the Digital Age: Design and Manufacturing*. New York, NY: Spon Press.

Aish, R., & Woodbury, R. (2005). Multi-level Interaction in Parametric Design. In A. Butz, B. Fisher, A. Krüger, & P. Olivier (Eds.), *Smart Graphics: 5th International Symposium, SG 2005, Frauenwörth Cloister, Germany, August 22-24, 2005. Proceedings* (pp. 151-162). Berlin, Heidelberg: Springer.

Anderberg, M. R. (1973). *Cluster analysis for applications*. New York, NY: Academic Press.

Anderson, K. (2016). *So Happy Together: Architects and Energy Modelers Informing Building Design*. Paper presented at the ASHRAE and IBPSA-USA SimBuild 2016: Building Performance Modeling Conference, Salt Lake City, Utah.

Bailey, K. (1994). Numerical Taxonomy and Cluster Analysis. In M. S. Lewis-Beck (Ed.), Typologies and Taxonomies: An Introduction to Classification Techniques. Thousand Oaks, California: SAGE Publications, Inc. Retrieved from https://methods.sagepub.com/book/typologies-and-taxonomies. doi:10.4135/9781412986397

Barnes, M. R. (1999). Form Finding and Analysis of Tension Structures by Dynamic Relaxation. *International Journal of Space Structures, 14*(2), 89-104. doi:10.1260/0266351991494722

Bauckhage, C. (2015). *NumPy / SciPy Recipes for Data Science: k-Medoids Clustering*. Retrieved from https://dx.doi.org/10.13140/2.1.4453.2009

Beaudouin-Lafon, M. (2004). *Designing Interaction, not Interfaces*. Paper presented at the Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '04), Gallipoli, Italy.

Bechtel, W., & Abrahamsen, A. A. (2002). *Connectionism and the mind : parallel processing, dynamics, and evolution in networks. 2nd ed. William Bechtel and Adele Abrahamsen*: Malden, Mass. ; Oxford, England : Blackwell, 2002. 2nd ed.

Brière.-Côté, A., Rivest, L., & Maranzana, R. (2012). Comparing 3D CAD Models: Uses, Methods, Tools and Perspectives. *Computer-Aided Design & Applications, 9*(6), 771-794. doi:10.3722/cadaps.2012.771-794

Brown, N. C., & Mueller, C. T. (2019). Quantifying Dversity in Parametric Design: A Comparison of Possible Metrics. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM, 33*(1), 40-53. doi:10.1017/S0890060418000033

Caldas, L. G. (2001). *An Evolution-Based Generative Design System: Using Adaptation to Shape Architectural Form.* (Doctoral Dissertation), Massachusetts Institute of Technology, MIT Press.

Caplan, B. (2011). Parametric Design's Greatest Value to Architecture Is to Attain Eco-sustainability. *The Architectural Review, EMAP Architecture*.

Celani, G., & Eduardo Verzola Vaz, C. (2012). CAD scripting and visual programming languages for implementing computational design concepts: A comparison from a pedagogical point of view. *International Journal of Architectural Computing, 10*(1), 121-137. doi:10.1260/1478-0771.10.1.121

Cha, M. Y., & Gero, J. S. (1998). Shape Pattern Recognition Using a Computable Pattern Representation. In J. S. Gero & F. Sudweeks (Eds.), *Artificial Intelligence in Design '98* (pp. 169-187). Dordrecht: Springer Netherlands.

Chakrabarti, K., Ortega-Binderberger, M., Porkaew, K., & Mehrotra, S. (2000). *Similar shape retrieval in MARS.* Paper presented at the 2000 IEEE International Conference on Multimedia and Expo.

Clarke, J. A. (2001). *Energy simulation in building design. (2nd ed.) J.A. Clarke*: Oxford ; Boston : Butterworth-Heinemann, 2001.

Collette, Y., & Siarry, P. (2013). *Multiobjective optimization : principles and case studies. Yann Collette, Patrick Siarry*: Berlin ; New York : Springer.

Coyne, R., Gero, J., Rosenman, M., & Radford, A. (1987). Innovation and creativity in knowledge-based CAD. In J. S. Gero (Ed.), *Expert Systems in Computer-Aided Design* (pp. 435-465). North-Holland, Amsterdam.

Cvetkovic, D., & Parmee, I. C. (2002). Preferences and their application in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, (1), 42. doi:10.1109/4235.985691

de las Heras, L.-P., Fernández, D., Fornés, A., Valveny, E., Sánchez, G., & Lladós, J. (2014, 2014//). *Runlength Histogram Image Signature for Perceptual Retrieval of Architectural Floor Plans.* Paper presented at the Graphics Recognition. Current Trends and Challenges, Berlin, Heidelberg.

DeLanda, M. (2002). *Deleuze and the Use of the Genetic Algorithm in Architecture*. Paper presented at the Architectural DesignBetween Bladerunner and Mickey Mouse: New Architecture in Los Angeles, Madrid, Spain.

Dutta, A., Lladós, J., Bunke, H., & Pal, U. (2014). *A Product Graph Based Method for Dual Subgraph Matching Applied to Symbol Spotting*, Berlin, Heidelberg.

Elghazi, Y., Wagdy, A., Mohamed, S., & Hassan, A. (2014). *Daylighting driven design: optimizing kaleidocycle facade for hot arid climate.* Paper presented at the Aachen: Fifth German-Austrian IBPSA Conference, RWTH Aachen University.

EnergyPlus. (2015). *EnergyPlus Input-Output Reference 8.6*. Retrieved from https://energyplus.net/sites/all/modules/custom/nrel_custom/pdfs/pdfs_v8.9.0/InputOutputReference.pdf

Everitt, B. (2011). *Cluster analysis. 5th ed. Brian S. Everitt ... [and others]*: Chichester, West Sussex, U.K. : Wiley, 2011.

Fonseca, C. M., & Fleming, P. J. (1993). *Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization.* Paper presented at the Icga.

Funkhouser, T., Kazhdan, M., Min, P., & Shilane, P. (2005). Shape-Based Retrieval and Analysis of 3D Models. *Communications of the ACM, 48*(6), 58-64. doi:10.1145/1064830.1064859

Gero, J. S. (1990). Design prototypes: a knowledge representation schema for design. *AI Magazine, 11*(4), 26-36.

Gullo, F., Ponti, G., & Tagarelli, A. (2008). *Clustering Uncertain Data Via K-Medoids*, Berlin, Heidelberg.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*: Elsevier.

Harding, J., & Brandt-Olsen, C. (2018). Biomorpher: Interactive evolution for parametric design. *International Journal of Architectural Computing, 16*(2), 144-163. doi:10.1177/1478077118778579

Hauglustaine, J.-M., & Azar, S. (2001). *Interactive tool aiding to optimise the building envelope during the sketch design.* Paper presented at the Proceedings of the Seventh International IBPSA Conference (BS2001), Rio de Janeiro, Brazil.

Holzer, D. (2015). BIM and Parametric Design in Academia and Practice: The Changing Context of Knowledge Acquisition and Application in the Digital Age. *International Journal of Architectural Computing, 13*(1), 65-82.

Huang, T., Mehrotra, S., & Ramchandran, K. (1997). Multimedia Analysis and Retrieval System (MARS) Project. In B. Sandore (Ed.), *Digital Image Access & Retrieval*: Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign.

Izard, V., Pica, P., Dehaene, S., Hinchey, D., & Spelke, E. (2011). Chapter 19 - Geometry as a Universal Mental Construction. In S. Dehaene & E. M. Brannon (Eds.), *Space, Time and Number in the Brain* (pp. 319-332). San Diego: Academic Press.

Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing Surveys (CSUR), 31*(3), 264-323.

Jayanti, S., Kalyanaraman, Y., & Ramani, K. (2009). Shape-based clustering for 3D CAD objects: A comparative study of effectiveness. *Computer-Aided Design, 41*(12), 999-1007.

Jin, X., & Han, J. (2016). K-Medoids Clustering. In C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning and Data Mining* (pp. 1-3). Boston, MA: Springer US.

Joshi, S., & Srivastava, A. (2003). *A geometric approach to shape clustering and learning.* Paper presented at the IEEE Workshop on Statistical Signal Processing, Piscataway, NJ, USA.

Kalvelagen, E. (2015). Visualization of large multi-criteria result sets with plot.ly. Retrieved from http://amsterdamoptimization.com/viz/plota1.html

Kaufmann, L., & Rousseeuw, P. J. (1987). *Clustering by means of medoids*. Delft: Faculty of Mathematics and Informatics.

Konak, A., Coit, D. W., & Smith, A. E. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety, 91*(9), 992-1007.

Korf, R. E. (2002). *A new algorithm for optimal bin packing.* Paper presented at the Eighteenth National Conference on Artificial intelligence, Edmonton, Alberta, Canada.

Korsah, G. A., Stentz, A., & Dias, M. B. (2007). *The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs*. Retrieved from Carnegie Mellon University, Pittsburgh, PA:

Kunz, J. (1989). Concurrent knowledge systems engineering. In: Stanford University, Center for Integrated Facility Engineering.

Lagios, K., Niemasz, J., & Reinhart, C. F. (2010). Animated building performance simulation (abps)-linking rhinoceros/grasshopper with radiance/daysim. *Proceedings of SimBuild*.

Lu, G., & Sajjanhar, A. (1999). Region-based shape representation and similarity measure suitable for content-based image retrieval. *Multimedia Systems, 7*(2), 165-174.

MacKenzie, I. S. (2012). *Human-Computer Interaction : An Empirical Research Perspective*. San Francisco, US: Elsevier Science & Technology.

Malkawi, A. M. (2005). Performance simulation: research and tools. In A. M. Branko Kolarevic (Ed.), *Performative Architecture: Beyond Instrumentality* (pp. 85-96): Spon Press, New York.

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*: Cambridge ; New York : Cambridge University Press, 2008.

Mark, E., Martens, B., & Oxman, R. (2001). *The Ideal Computer Curriculum.* Paper presented at the Architectural Information Management, 19th eCAADe Conference, Helsinki University of Technology, Helsinki, Finland.

Martello, S., & Toth, P. (1990a). Bin-packing problem. In *Knapsack problems: Algorithms and computer implementations* (pp. 221-245): Chichester ; New York : J. Wiley & Sons.

Martello, S., & Toth, P. (1990b). Lower bounds and reduction procedures for the bin packing problem. *Discrete Applied Mathematics, 28*(1), 59-70.

Mitchell, W. J. (1975). The theoretical foundation of computer-aided architectural design. *Environment and Planning B: Urban Analytics and City Science, 2*(2), 127-150. doi:https://doi.org/10.1068/b020127

Musnjak, M., & Golub, M. (2004). *Using a set of elite individuals in a genetic algorithm.* Paper presented at the 26th International Conference on Information Technology Interfaces - ITI 2004, Cavtat, Croatia.

Nejur, A., & Steinfeld, K. (2016). Ivy: Bringing a Weighted-Mesh Representation to Bear on Generative Architectural Design Applications. *ACADIA 2016: POSTHUMAN FRONTIERS: Data, Designers, and Cognitive Machines, Proceedings of the 36th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, 140-151.

Nguyen, A.-T., Reiter, S., & Rigo, P. (2014). A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy, 113*, 1043-1058.

Norouzi, M., Fleet, D. J., & Salakhutdinov, R. R. (2012). *Hamming Distance Metric Learning.* Paper presented at the NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, Nevada.

NREL. (2011). *U.S. Department of Energy Commercial Reference Building Models of the National Building Stock* Retrieved from https://www.nrel.gov/docs/fy11osti/46861.pdf

Phillips, J., Livingston, G., & Buchanan, B. (2002). Toward a computational model of hypothesis formation and model building in science. In *Model-Based Reasoning* (pp. 209-225): Springer.

Piker, D. (2013). Kangaroo: form finding with computational physics. *Architectural Design, 83*(2), 136-137.

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies, 2*(1), 37-63.

Provost, F., & Kohavi, R. (1998). Glossary of terms. *Journal of Machine Learning, 30*(2-3), 271-274.

Radford, A. D., & Gero, J. S. (1987). *Design by optimization in architecture, building, and construction*: John Wiley & Sons, Inc.

Radiance. (1997). *The RADIANCE 5.1 Synthetic Imaging System*. Retrieved from http://radsite.lbl.gov/radiance/refer/ray.html#Materials.

Ramsden, J. (2015). Batchrun Component for Grasshopper. Retrieved from http://james-ramsden.com/download/batchrun-component-for-grasshopper/

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association, 66*(336), 846-850.

Robinson, D. J. (2008). *An introduction to abstract algebra*: New York : Walter de Gruyter.

Rodrigues, E., Gaspar, A. R., & Gomes, Á. (2013). An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology. *Computer-Aided Design, 45*(5), 887-897.

Rodrigues, E., Sousa-Rodrigues, D., de Sampayo, M. T., Gaspar, A. R., Gomes, Á., & Antunes, C. H. (2017). Clustering of Architectural Floor Plans: A Comparison of Shape Representations. *Automation in Construction, 80*, 48-65. doi:https://doi.org/10.1016/j.autcon.2017.03.017

Roudsari, M. S. (2018). Ladybug 0.0.67 and Honeybee 0.0.64 [Legacy Plugins]. Retrieved from https://www.food4rhino.com/app/ladybug-tools

Roudsari, M. S., Pak, M., & Smith, A. (2013). *Ladybug: a parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design.* Paper presented at the Proceedings of the 13th International IBPSA Conference Lyon, France.

Safar, M., Shahabi, C., & Sun, X. (2000). *Image retrieval by shape: a comparative study.* Paper presented at the Proceedings of International Conference on Multimedia and Expo. ICME 2000, New York, NY, USA.

Sajjanhar, A., & Lu, G. (1997). A grid-based shape indexing and retrieval method. *Australian Computer Journal, 29*(4), 131-140.

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys, 16*(2), 187-260. doi:10.1145/356924.356930

Samuelson, H., Claussnitzer, S., Goyal, A., Chen, Y., & Romo-Castillo, A. (2016). Parametric energy simulation in early design: High-rise residential buildings in urban contexts. *Building and Environment, 101*, 19-31.

Sargent, R. G., & Balci, O. (2017, 3-6 Dec. 2017). *History of verification and validation of simulation models.* Paper presented at the 2017 Winter Simulation Conference (WSC).

Sasaki, Y. (2007). The truth of the F-measure. In *Teaching, Tutorial Materials* (Version: 26th, pp. 1-5).

Shea, K., Aish, R., & Gourtovaia, M. (2005). Towards integrated performance-driven generative design tools. *Automation in Construction, 14*(2), 253-264.

Shen, W., Wang, Y., Bai, X., Wang, H., & Latecki, L. J. (2013). Shape clustering: Common structure discovery. *Pattern Recognition, 46*(2), 539-550.

Simergy. (2013). Ideal Loads, Stencil: Zone HVAC Stencil: Ideal Loads System. Retrieved from https://d-alchemy.com/html/helpdocs/Simergy/Content/HVAC_Systems/ZHG_Components/Ideal_Loads.htm

Sousa-Rodrigues, D., de Sampayo, M. T., Rodrigues, E., Gaspar, A. R., Gomes, Á., & Antunes, C. H. (2015). *Online survey for collective clustering of computer generated architectural floor plans*. Retrieved from http://arxiv.org.srv-proxy1.library.tamu.edu/abs/1504.08145

Steinfeld, K. (2017). *Dreams May Come.* Paper presented at the ACADIA 2017: DISCIPLINES & DISRUPTION. Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), MIT, Cambridge, MA.

Stocking, A. W. (2009). Generative Design Is Changing the Face of Architecture. Retrieved from https://www.cadalyst.com/cad/building-design/generative-design-is-changing-face-architecture-12948

Story, M., & Congalton, R. G. (1986). Accuracy assessment: a user's perspective. *Photogrammetric Engineering and Remote Sensing, 52*(3), 397-399.

Toffolo, A., & Benini, E. (2003). Genetic diversity as an objective in multi-objective evolutionary algorithms. *Evolutionary Computation, 11*(2), 151-167. doi:https://doi-org.srv-proxy1.library.tamu.edu/10.1162/106365603766646816

Tom. (2014). Rand index calculation. https://stats.stackexchange.com/q/110712. Retrieved from (https://stats.stackexchange.com/users/23823/tom)

Torres, S., & Sakamoto, Y. (2007). *Facade design optimization for daylight with a simple genetic algorithm.* Paper presented at the Proceedings of Building Simulation.

Velmurugan, T., & Santhanam, T. (2010). Computational complexity between K-means and K-medoids clustering algorithms for normal and uniform distributions of data points. *Journal of Computer Science, 6*(3), 363-368.

Vierlinger, R., & Bollinger, K. (2014). *Acommodating Change in Parametric Design.* Paper presented at the ACADIA 14: Design Agency. Proceedings of the 34th Annual

Conference of the Association for Computer Aided Design in Architecture (ACADIA), Los Angeles.

Wagstaff, K., Cardie, C., Rogers, S., & Schrödl, S. (2001). *Constrained k-means clustering with background knowledge.* Paper presented at the ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning.

Wang, W., Zmeureanu, R., & Rivard, H. (2005). Applying multi-objective genetic algorithms in green building design optimization. *Building and Environment, 40*(11), 1512-1525.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association, 58*(301), 236-244.

Wiegand, T., Sullivan, G. J., Bjontegaard, G., & Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(7), 560-576.

Wilks, D. S. (2011). Cluster analysis. In *International Geophysics* (Vol. 100, pp. 603-616): Elsevier.

Woodbury, R. (2010). *Elements of parametric design*: Routledge.

Wortmann, T., & Nannicini, G. (2017). Introduction to Architectural Design Optimization. In *City Networks* (pp. 259-278): Springer.

Yan, W., Asl, M. R., Su, Z., & Altabtabai, J. (2015). Towards Multi-Objective Optimization for Sustainable Buildings with Both Quantifiable and Non-Quantifiable Design Objectives. In *Sustainable Human–Building Ecosystems* (pp. 223-230).

Yi, Y. K., Jung, B. R., & Sullivan, J. (2012). *Agent-based Geometry Control Optimized by Genetic Algorithm for Daylighting.* Paper presented at the ASim 2012 Proceedings, Shanghai, China.

Yousif, S., Clayton, M., & Yan, W. (2018). *Towards Integrating Aesthetic Variables in Architectural Design Optimization*. Paper presented at the The 106th ACSA Annual Meeting, The Ethical Imperative, The Association of Collegiate Schools of Architecture (ACSA), Denver, Colorado.

Yousif, S., & Yan, W. (2018). *Clustering Forms for Enhancing Architectural Design Optimization*. Paper presented at the Learning, Adapting and Prototyping, The 23rd Conference of the Association for Computer-Aided Architectural Design Research in

Asia (CAADRIA), Beijing, China. http://papers.cumincad.org/cgi-bin/works/paper/caadria2018_257

Yousif, S., & Yan, W. (2019). *Shape Clustering Using K-Medoids in Architectural Form Finding.* Paper presented at the Computer-Aided Architectural Design Futures (CAADFutures) 2007: Proceedings of the 18th International CAAD Futures Conference, Daejeon, Korea.

Yousif, S., Yan, W., & Culp, C. (2017). *Incorporating Form Diversity into Architectural Design Optimization*. Paper presented at the ACADIA 2017: DISCIPLINES & DISRUPTION Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), MIT, Cambridge, MA. http://papers.cumincad.org/cgi-bin/works/paper/acadia17_640.

Zhang, D., & Lu, G. (2004). Review of shape representation and description techniques. *Pattern Recognition, 37*(1), 1-19.

Zhang, Y., Lin, K., Zhang, Q., & Di, H. (2006). Ideal thermophysical properties for free-cooling (or heating) buildings with constant thermal physical property material. *Energy and Buildings, 38*(10), 1164-1170.

Zwierzycki, M., Nicholas, P., & Ramsgaard Thomsen, M. (2018). Localised and Learnt Applications of Machine Learning for Robotic Incremental Sheet Forming. In K. De Rycke, C. Gengnagel, O. Baverel, J. Burry, C. Mueller, M. M. Nguyen, P. Rahm, & M. R. Thomsen (Eds.), *Humanizing Digital Reality: Design Modelling Symposium Paris 2017* (pp. 373-382). Singapore: Springer.