IMPROVING DESIGN OPTIMIZATION AND OPTIMIZATION - BASED DESIGN KNOWLEDGE DISCOVERY

A Dissertation

by

ZHOUZHOU SU

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

August 2015

Major Subject: Architecture

ABSTRACT

The use of design optimization in the early stages of architectural design process has attracted a high volume of research in recent years. However, traditional design optimization requires a significant amount of computing time, especially when there are multiple design objectives to achieve. What’s more, there is a lack of studies in the current research on automatic generation of architectural design knowledge from optimization results. This paper presents computational methods for creating and improving a closed loop of design optimization and knowledge discovery in architecture. It first introduces a design knowledge-assisted optimization improvement method with the techniques - offline simulation and Divide & Conquer (D&C) - to reduce the computing time and improve the efficiency of the design optimization process utilizing architectural domain knowledge. It then describes a new design knowledge discovery system where design knowledge can be discovered from optimization through an automatic data mining approach. The discovered knowledge has the potential to further help improve the efficiency of the optimization method, thus forming a closed loop of improving optimization and knowledge discovery. The validations of both methods are presented in the context of a case study with parametric form-finding for a nursing unit design with two design objectives: minimizing the nurses’ travel distance and maximizing daylighting performance in patient rooms.

DEDICATION

To my beloved husband, Yin Jiang, and our son, Dylan.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Wei Yan for his unconditional support. My gratitude also goes to Dr. Mardelle Shepley, Dr. Xuemei Zhu, and Dr. Zhulian Kang, for their insights and help throughout the years. In addition, a thank you to Mohammad Rahmani Asl for inspiration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

This chapter addresses two existing problems associated with the present process of architectural design. Problem 1: traditional design optimization and building performance simulation are very time consuming and; Problem 2: there is a lack of study about generating design knowledge through optimization automatically. It then proposes a research framework to solve these problems. It later states the research objectives, research questions and the significance of this study. The outline of this study is presented lastly.

1.1 Problem Statement

This section discusses the existing problems in the present architectural design process. These problems include: the complexity in architectural design, limitations in traditional design process, the time consuming issue in building performance simulation and optimization, and the lack of study to discover knowledge through optimization.

1.1.1 Design Complexity

Architectural design is a complex decision-making and goal-oriented activity. It can be seen as a Multi-Objective Optimization process aimed at finding optimal solutions for multiple objectives (Radford & Gero, 1987). Architects make design decisions about the practical functions and physical forms of buildings to meet design objectives. Among all types of architectural design, healthcare facilities are one of the most complex building types. Therefore, in this dissertation, healthcare design is used as an example for architectural design. The reasons are threefold and listed below.

First, there are a large number of possible solutions for a given design problem in healthcare architectural design. In the traditional design process, architects and planners create, develop, and modify a potential design based on assumptions and previous experiences. Due to tight project timelines, the number of design options an architect can create and evaluate is often limited. It is difficult to achieve an optimal design solution through the limited number of options that can usually be created during a traditional design process. Thus, architects might find it difficult to convince clients that their proposed solution is ideal; instead, it might just be one of many acceptable solutions (Kim & Shepley, 2007). Kim and Shepley pointed out that one of the reasons architects' credibility can be perceived as low is their lack of confidence that their proposed design solutions are the absolute best.

Second, healthcare facilities involve a significant number of design objectives. Evidence-Based Design (EBD) is a popular design process wherein decisions about the physical space are based on research outcomes. Learning from EBD, we know that the design of healthcare facilities is governed by the needs and goals of the physical space, such as geometric typology, related functions, travel distance, access to daylight, energy consumption, etc. These design objectives often conflict with one another (Radford & Gero, 1987). For example, increasing access to views of nature from patient rooms might also increase overall energy consumption. It is very difficult to consider and balance all of the design objectives through the traditional design process.

Third, healthcare facilities not only impact the wellbeing of patients (Dettenkofer et al., 2004; Ulrich, 1984), but also affect physicians, staff members, caregivers (Harris et al., 2006), and patients' family members (Conner & Nelson, 1999).

The complexity of healthcare design can be reflected by the data about healthcare design firms' size and their years in operation (Kim & Shepley, 2007; 2011). Healthcare firms generally are older than other types of design firms. About 55% of the healthcare firms currently operating were founded before 1970, while 48% of all design firms now in operation were established after 1985 (Kim & Shepley, 2007; 2011). In most cases, only large firms have the ability to design healthcare projects. About 71% of healthcare design firms have more than 20 staff members; only 10% of all architectural firms are of approximately that same size (Kim & Shepley, 2007; 2011). These data show that healthcare design firms require more specialized knowledge, more experience, and more employees.

Given the complexity of the healthcare industry, there is a need for better methods to assist architects with the decision making process, help them create and evaluate design options, and optimize design solutions within a reasonable amount of time.

1.1.2 Traditional Design Process and Design Optimization

At present, architectural design activities usually proceed in the traditional fashion. Mainly, architects use assumptions and previous experiences to create and modify potential design solutions in order to arrive at a design that is acceptable in both form and function. The number of solutions an architect can create and evaluate within a reasonable timeframe is limited. Thus, there is no guarantee that the final design product will be the optimal design solution, or even close to it. Furthermore, due to various factors including a large number of building parameters, a variety of design needs which often conflict with one another, and an enormous number of possible design solutions, the traditional design process often leads to final design outcomes that are far from optimal.

Computer-Aided Architectural Design (CAAD) has reformed architectural design through its ability to support the creation and analysis of a design. One of the most powerful techniques of computer-aided design is design optimization. Design optimization is a design method that searches for the optimum solution – the design that best meets all specified requirements, for example, achieving the lowest energy consumption, or in a nursing unit design allowing the least amount of walking distance for nurses (Ansys, 2007). Compared to the traditional design process, design optimization can search for and evaluate a large number of design solutions and pinpoint the best-fitting ones, according to the specified design objectives.

1.1.3 Building Performance Simulations and Design Optimization: Time Consuming Process

Building performance simulations are now regularly being used by building professionals to test design alternatives before the construction phase of the project is initiated (Hong et al., 2000), although the number of alternatives is limited as discussed above. These simulations are commonly used together with design optimization tools to find the best performing design alternatives. Both building performance simulations and design optimization are powerful techniques for helping architects make better design decisions. However, including building performance simulations in the design optimization process has also made the practice of architectural design much more complex and time consuming. As a result, the integration of both techniques is often unrealistic in real-world design practice; most projects simply have too tight a project timeline to apply both.

The first part of this study focuses on reducing the computing time needed for design optimization, utilizing design knowledge, when building simulation techniques are involved. The methods used in this research not only facilitate faster optimization, but more importantly they enable a much larger search space within the same amount of time, which offers a better chance of finding the optimal design.

1.1.4 Knowledge Discovery through Optimization

Knowledge discovery is the extraction process of nontrivial information that is unclear, previously unknown, and has potentially useful knowledge obtainable from data (Piateski & Frawley, 1991). Useful knowledge can be discovered from design optimization results. Previous work in the area of knowledge discovery through optimization has mostly been on prototypes created for demonstration purposes, and the knowledge (design rules) yielded by these simple prototypes have not offered new information to designers (e.g., one should use pre-stressed concrete for minimum slab thickness, etc.) (Mackenzie & Gero, 1987). Moreover, existing knowledge discovery methods are based on manual and visual analyses of the optimization results. For example, researchers point out that knowledge about design and performance relationships can be obtained through a manual analysis of the results of Pareto optimization (Radford & Gero, 1987). As the use of design optimization grows, however, there is an increasing need to generate useful knowledge from the results of optimization quickly through an automatic process. In the second part of this study, I use an automatic data mining approach to generate design knowledge - rules and the relationships among design variables and design outcomes - based on optimization results; the learned knowledge is the relationship between the objectives (e.g., building performance) and the decision variables (e.g., building layouts).

1.2 Proposed Research

This study proposes a closed loop of design optimization improvement and knowledge discovery in architecture. This new framework consists of two studies: (1) a design knowledge-assisted optimization improvement method that uses the techniques of offline simulation and Divide and Conquer to reduce the computing time and improve the efficiency of a design optimization process utilizing architectural domain knowledge; and (2) a new design knowledge discovery system where design knowledge can be generated from optimization through an automatic data mining approach. This new knowledge has the potential to further help improve the efficiency of the optimization method, thus forming a closed loop of optimization improvement and knowledge discovery. The validations of both methods are presented in the context of a case study with parametric form-finding for a nursing unit design with two design objectives: minimizing the nurses' travel distance and maximizing daylighting performance in patient rooms. Each methodology is explained in detail below.

1.2.1 Improving Design Optimization Using Architectural Domain Knowledge

Integrated with parametric modeling and an improved optimization process, the first part of this study examines methods for design optimization improvement using architectural domain knowledge.

Computational simulation is considered as one of the most powerful analytic tools to study things (Hensen & Lamberts, 2012). Building performance simulation is one kind of computational simulation. It seeks to predict the performance such as energy performance of a building in the real world (Hensen & Lamberts, 2012). It is broadly used by building professionals to test design alternatives before the construction phase of the project (Hong et al., 2000). When design objectives include energy and daylight performances in design optimization, building performance simulations (such as energy simulations) are needed in the optimization process. Both building performance simulation and design optimization are very time consuming and require a significant amount of computing power. More time is needed when both techniques are required to work together. Thus, the first part of this research focuses on reducing the computing time in design optimization when building simulation techniques are involved. This reduction in computing time is enabled by the utilization of architectural domain knowledge to alleviate the need for expensive simulations.

This study facilitates a process of simulation and optimization for specific architectural design objectives in a healthcare design problem. It focuses on the early stages of the design decision-making process and is composed of the following computational methods: parametric modeling, performance simulation, and multi-objective design optimization.

1.2.2 Discovering Design Knowledge from Optimization

The second part of this research study explores data mining techniques in design optimization to automatically generate design rules and knowledge using learned correlations between optimal performances and design variables. The information generated by one optimization problem can be applied to other, similar problems (Radford & Gero, 1987). Design knowledge - rules and the relationships among design variables and design outcomes - can be learned from the results of multi-objective optimization. This knowledge has the potential to be utilized as guidelines for future designs that reduce the need for simulation and optimization during the design process.

1.3 Research Objectives

The research objectives of this study include providing methods and prototypical tools to: (1) improve the efficiency of optimization by reducing simulation needs when building simulations are incorporated into design optimization; (2) study how design knowledge can be used to speed up the optimization process by reducing the computational complexity of the design problem; (3) find a design knowledge discovery method towards automatically generating useful correlations and causal relationships among decision variables, among design solutions, and between decision variables and design solutions for a given optimization problem; the knowledge can form design guidelines for specific design problems with similar design variables, constraints, and design objectives and can form general design guidelines in the future to reduce the need for costly optimization computations.

1.4 Research Questions

The main research question in this study is:

How can we create a knowledge-based design optimization and optimization-based knowledge discovery framework for architectural design?

The answer to this question has the potential to improve the applications of optimization in future architectural design practice.

The main research question can be subdivided into several sub-questions:

1. *How can we reduce the number of simulation runs when building simulations are coupled with design optimization?*

Reducing the number of simulation runs will significantly improve the efficiency of optimization.

2. *How can we improve computational optimization by reducing the complexity of the design problem in optimization?*

Optimization problems are generally very complex, and thus require a long computational time. Reducing the complexity of a problem can greatly benefit optimization in terms of the amount of time spent.

3. *How can we automatically generate valuable design knowledge from optimization?*

Much useful design knowledge can be gleaned from the results of an optimization. Design knowledge generated from previous optimizations has the potential to improve the efficiency of the overall optimization method, thus forming a closed loop of optimization improvement and knowledge discovery.

1.5 Significance

This research provides a knowledge-based optimization and knowledge discovery framework for architectural design practice. It enables an efficient optimization method that can improve the productivity of current design optimization, as well as a proposed design knowledge discovery system. In this design knowledge discovery system, new design knowledge can be discovered through design optimization and, in turn, provide feedback for future design optimization, thus improving the overall efficiency of the optimization method. Figure 1.1 shows the relationship between the two interrelated components of this study: (1) Design Knowledge Assisted Optimization Improvement and (2) Design Knowledge Discovery through Optimization.

In the first phase of this study, the Design Knowledge Assisted Optimization Improvement method facilitates a fast and extensive creation and evaluation of design alternatives through a process of searches, simulations, and optimization in order to meet specified architectural design objectives. Compared to the traditional optimization method, it provides a rapid search process by reducing simulation runs and computational complexities for an optimal design using architectural domain design knowledge. The improved method used in this research not only provides a faster system for optimization, but more importantly it enables a larger search space within the same amount of time, which offers a better chance of finding the optimal design. This rapid design optimization method will make optimization more useful for architects in their practice.

In the second phase of this study, useful knowledge about the correlations among decision variables and optimal design solutions can be obtained from design optimization results. The new knowledge can, in turn, feed back into the optimization process to guide and help with future design optimizations.

This study focuses on the early stages of the design decision-making process. While improving today's practice is the ultimate goal of the project, the major contribution of this project is the future potential demonstrated by the present advancement of the optimization method.

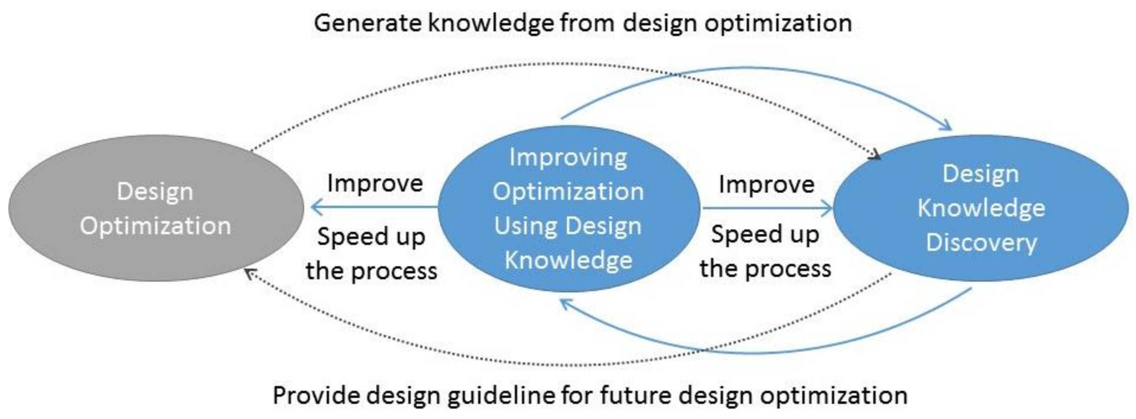

Generate knowledge from design optimization
Design Optimization
Improve
Speed up the process
Improving Optimization Using Design Knowledge
Improve
Speed up the process
Design Knowledge Discovery
Provide design guideline for future design optimization

Figure 1.1 Conceptual Model of the Research.

1.6 Outline of Dissertation

This dissertation is structured as follows:

Chapter 1 introduces existing problems associated with the present process of architectural design and proposes a research framework to solve these problems.

Chapter 2 presents a literature review on various topics closely related to this study.

Chapter 3 describes the research methodology used for this study.

Chapter 4 suggests two methods of improving the Genetic Algorithm process of design optimization. It continues with a case study to validate these methods.

Chapter 5 proposes a knowledge discovery system to help future design.

Chapter 6 discusses the reliability and validity of the methods of this study, and provides a summary of the study, its findings, and future work.

CHAPTER II

LITERATURE REVIEW*

This chapter presents a literature review on the following topics that are closely related to this study: optimal design in architecture, design complexity in healthcare architecture, parametric modeling, optimization methods and Genetic Algorithm, the time complexity of building simulation, the time complexity of GA, methods for improvement of optimization, data mining, and machine learning.

2.1 Optimal Design in Architecture

The role of the architect is to design and create buildings that best satisfy users' needs. Three main elements are involved in the design process: requirements, creation, and alternatives (Papalambros & Wilde, 2000). Papalambros and Wilde (2000, pp. 11-12) define design optimization as:

"*1. The selection of a set of variables to define the design alternatives.*

2. The selection of an objective (criterion), expressed in terms of the design variables, which we seek to minimize or maximize.

3. The determination of a set of constraints, expressed in terms of the design variables, which must be satisfied by any acceptable design.

4. The determination of a set of values for the design variables, which minimize (or maximize) the objective, while satisfying all the constraints."

---

* Part of this chapter is reprinted with permission from "Improving Genetic Algorithm for Design Optimization Using Architectural Domain Knowledge" by Su, Z., Yan, W., 2014, *Proceedings of the Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), Copyright 2014 by ACADIA.*

Generally, architectural design problems are complex and often involve multiple design objectives. Besides energy efficiency, other design objectives architects might consider throughout the comprehensive architectural design process include: aesthetic, structural, functional, social, historical, behavioral, etc. (Lobos & Donath, 2010). Thus, architectural design can be seen as a multi-objective optimization process aimed at finding optimal solutions for multiple, often conflicting, objectives (Radford & Gero, 1987). As stated above, these design objectives may at times be in conflict with one another. For example, increasing access to views of nature from patient rooms might, at the same time, increase overall energy consumption; reducing nurses' travel distances might also decrease the overall daylight performance in the nursing unit. Some design objectives are quantifiable and can be expressed by numerical values; they can be maximized or minimized. Other design objectives are unquantifiable and difficult to measure. As an example, according to Lobos and Donath (2010), in the realm of healthcare architectural design, the design objectives involved during the decision-making process may include:

1. Aesthetics: about the physical form of the building. Objective: to provide aesthetically pleasing architecture designs; unquantifiable.

2. Geometric Typology: possible configurations of space distributions such as L-shaped, U-shaped, Linear, Rectangular, etc. Objective: to address program needs and other considerations such as building site restrictions; unquantifiable.

3. Functional Relationships: some functions of a space program are more closely related to one another, and therefore should be adjacent. Objective: to establish a hierarchy (such as highly, fairly, or scarcely related) in the relationship between functions and rooms; quantifiable.

4. Travel Distance: the travel distance between certain rooms is an important consideration in healthcare design, such as in nursing unit design where travel distance is a major burden for nurses (Hendrich et al., 2008). Objective: to provide the minimum travel distance among rooms; quantifiable.

5. Functional Efficiency: the net to gross area ratio. Objective: to keep most of the area for healthcare functions and less for circulation; quantifiable.

6. Nature Views: research studies have shown a positive relationship between access to views of nature and improvement in patient outcomes such as reductions in stress, pain, and length of stay (Ulrich, 1984). Objective: to locate spaces (such as patient rooms) and windows in appropriate places and orientations in relation to views; unquantifiable.

7. Daylighting: it has been found that access to daylight contributes to higher satisfaction for nurses and can reduce the pain and the incidence of depression for patients (Zimring et al., 2008). Objective: to provide sufficient daylight to every room in need; quantifiable.

8. Energy Consumption: sustainable, efficient energy consumption and minimal environmental impact have become major objectives in building design (NSF, 2009). Objective: to minimize building energy consumption; quantifiable.

9. Visual Communication: Nurses can better supervise patients when they have better visual communication from nursing stations. Objective: to maximize visual communication in certain areas of the building; quantifiable.

In this study, travel distance and daylighting have been selected as the sample quantifiable design objectives in a nursing unit design for the case studies.

2.2 Design Complexity in Healthcare Architecture

Among the various types of architectural design, healthcare facilities are one of the most complex because they require a wide range of specialized knowledge. Kim and Shepley (2011) point out that such specialized knowledge and skills can increase healthcare architects' autonomy. This specialized knowledge includes functional complexity, technological complexity, research complexity, aesthetic complexity, and interest group complexity (Kim & Shepley, 2011).

1. Functional complexity. Compared to other building types, healthcare facilities have complex functional and circulation systems. There are three main functional zones in a contemporary hospital: (1) medical services such as medical units, Intensive Care Units, emergency departments, and imaging departments; (2) medical supports such as central sterile supply; and (3) general support services such as linen and food supply. Healthcare architects need to have specialized knowledge and abundant past experience to design any of the above-mentioned functions. Perhaps that is the reason that healthcare firms generally are older than other types of design firms. About 55% of the currently existing healthcare firms were founded before 1970, while 48% of all design firms were established after 1985 (Kim & Shepley, 2008).

2. Technological complexity. The field of medical technology has gained spectacular improvement in the past 50 years (Farncombe & Iniewski, 2013). Hospitals around the world are now constantly in need of renovation and expansion in an effort to incorporate the latest medical equipment (TMHC Staff, 2014). Healthcare architects should have knowledge of the newest medical equipment in order to provide sufficient space in their healthcare projects.

3. Research (scientific knowledge) complexity. Evidence-based design has dramatically changed healthcare-focused architectural design. It shows that healthcare facilities not only impact the wellbeing of patients (Dettenkofer et al., 2004; Ulrich, 1984), but also affect physicians, staff members, caregivers (Harris et al., 2006), and patients' family members (Conner & Nelson, 1999). Kim and Shepley (2011) suggest that the research can be categorized into two knowledge domains: medical knowledge that focuses on the wellbeing of patients, and environmental psychology that focuses on providing a healing and welcoming clinical environment for patients and their families. According to Kim and Shepley's research (2011), there are two main purposes for architectural research: program development and design decision making.

4. **Aesthetic complexity**. Architects' professional autonomy with regards to the aesthetic components of their designs is decreasing due to external constraints such as client requirements and budget (Ferris, 1996). Different people may have different aesthetic preferences. These differences may lead to disagreements between architects and clients about aesthetics-based design decisions (Kim & Shepley, 2011).

5. **Interest group complexity**. Many interest groups (funders, doctors, nurses, managers, etc.) can be involved during the decision-making process for a healthcare facility's design. Multiple voices among various interest groups often result in conflicting ideas (Kim & Shepley, 2011). Healthcare design complexity can result in task uncertainty. Specialized knowledge, especially research knowledge, should be brought in to overcome this uncertainty. The inclusion of specialized knowledge may also increase architects' autonomy (Kim & Shepley, 2011).

6. **Professional autonomy**. Autonomy is described as an individual's level of self-governance (DeVinne, 1987). Architects have a high level of autonomy when they have the freedom to make design decisions without the limitations of external controls and constraints. Healthcare design firms usually lack autonomy with regards to decision making. Based on Kim and Shepley's research (2011), 55% of the architectural firms surveyed claimed that they experienced low levels of autonomy. However, firms reporting high levels of autonomy (18%) tended to be large firms and more focused on healthcare projects (healthcare work made up 65% of all their projects).

In conclusion, Kim and Shepley (2008, 2011) suggest that specialized domains of knowledge in design such as functional efficiency, building technology, medical technology, and research knowledge play important roles in healthcare design complexity, and healthcare design complexity results in task uncertainty. Specialized knowledge, especially research knowledge, should be brought in to overcome this uncertainty. Specialized knowledge also increases architects' autonomy. Specifically, incorporating architectural research into design decisions may raise the level of professional autonomy.

Due to the design complexity of healthcare architecture, the complexity of computational optimization methods, and the limitation of available computing power, a significantly simplified healthcare architecture case study are selected for this research.

2.3 Methods for Design Optimization

Currently, two emerging computer modeling technologies can significantly benefit design optimization in architectural design. These two new technologies are parametric modeling and Generic Algorithm (GA). Each technology is further explained below.

2.3.1 Parametric Modeling

Digital modeling has greatly influenced the field of architectural design and construction. As new digital tools (such as parametric modeling) emerge, architects are able to explore new approaches to conceptual design (Schnabel, 2007; Stavric & Marina, 2011). Generally speaking, parametric modeling enables architects to use parameters and relationships (e.g., by using equations) to describe a complex building form; these descriptions can be updated automatically when the parameters change. Existing parametric geometry modeling tools used in architectural design include: SolidWorks®, Rhino® / Grasshopper®, and GenerativeComponents®, just to name a few. Most of these types of tools employ change propagation modeling methods, and some employ visual programming methods like Grasshopper® (Eastman et al., 2011; Woodbury, 2010).

2.3.2 Optimization Methods and GA

2.3.2.1 Optimization Methods

Design optimization is the process of using optimization methods to find the best design solutions that satisfy the project's requirements. Numerous methods exist for conducting such searches and solving such design problems. Besides GA, other optimization methods include: differential calculus, linear programming, and dynamic programming. Each method has its own advantages and disadvantages, and architects should be careful to select the right method based on the types of problems faced and the information at hand. The details of each method are discussed below.

(1) Differential Calculus. Differential calculus can provide quick, analytical solutions to design problems that can be expressed algebraically (Radford & Gero, 1987). In other words, the relationship between the design variables and design objectives is expressed as a series of continuous and differentiable equations. For example, take a public housing development design for which the design objective is to maximize the net benefit (Radford & Gero, 1987, pp. 39). A real estate consultant provides the relationship between the net benefit N and the target area of the housing to be developed as X: $N=100 + 100X – 40X^2$. The problem now is to find the maximum of N, when $N=100 + 100X – 40X^2$. By finding the derivative of N, we know that when the housing area is X=1.25, the net benefit is the maximum, which is 162.5. It is important to remember that differential calculus can only work with a single relationship between variables. When multiple relationships between variables occur, designers should assign the additional relationships as constraints (Radford & Gero, 1987). Differential calculus is useful when a design optimization problem can be formulated in a simple, continuous differential equation. The advantage of using differential calculus is that the method is simple, straightforward, and involves very little calculation. With a basic knowledge of differential calculus, one can quickly solve a problem without the use of a computer. Unfortunately, real world architectural design rarely sees design problems as simple as the example shown above. In most cases, an architectural design problem involves multiple design variables; also, there is usually more than one design objective that the designer may want to optimize simultaneously. What's more, the relationships among the design variables and design objectives cannot easily be formulated by equations. As a result, although differential calculus is a simple and elegant analytic method for solving optimization problems, rarely can we use it to solve real world architectural design optimization problems.

(2) Linear Programming. Linear programming is a subset of mathematical programming (Radford & Gero, 1987). In an optimization problem, when the objective can be expressed by a linear function with certain constraints, the problem can be solved mathematically by using linear programming. Due to its simplicity and flexibility, linear programming methodologies have been widely used in many areas (such as the physical and social sciences) since the development of the theory in 1948 (Spivey, 1962). The method is well developed and "*it guarantees to find the optimum solution in a fixed number of steps*" (Radford & Gero, 1987, pp. 50). Linear programming cannot solve all of the problems in a linear relationship except by satisfying the following three conditions. First, all design variables must be continuous, and at the same time greater than or equal to zero. Second, the relationships among the variables and objective functions must be expressed linearly. Third, the relationships among the variables and constraints must also be expressed linearly (Radford & Gero, 1987). Although linear programing is an important and useful method for solving optimization problems, generally speaking, a large percentage of design optimization problems do not satisfy the above-mentioned requirements.

(3) Dynamic Programming. Most architectural design problems have discrete, nonlinear, and stochastic decision variables; optimization methods such as classical calculus and linear programming are not suitable in these situations. Dynamic programming can solve design problems with these features by breaking the original problem down into a series of sub-problems that can be solved sequentially. Assembling the optimal solutions for these sub-problems yields an optimal solution to the original problem (Cooper & Cooper, 1981). Dynamic programming is applicable to a problem if it satisfies two requirements. First, the original objective function must be separable into a series of smaller problems. Second, the original problem must be able to be organized in a way that "*later decisions do not invalidate earlier ones*" (Radford & Gero, 1987, p. 110). For example, you want to travel from point A to point K as quickly as possible. There are many intersections between points A and K (see Figure 2.1). A heuristic designer might solve this problem by choosing the road that looks the shortest at every intersection. However, this will not guarantee the optimal solution because the designer might fail to see the entire picture. We can tackle this problem using an exhaustive search, which means finding all of the feasible routes and choosing the shortest. This would guarantee the finding of the optimal solution, but the process would be very slow. A dynamic programming algorithm looks at finding the shortest paths from points H, I, and J to point K, and uses those solutions to find the shortest paths from points E, F, and G to point K, and eventually the shortest path from point A to point K (Radford & Gero, 1987).

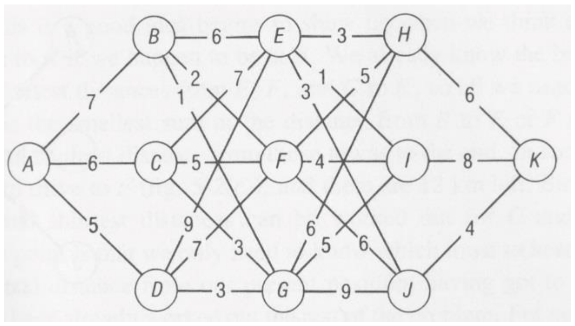


Figure 2.1. Finding the Shortest Route From A to K (Image Source: Radford & Gero, 1987)**.**

The computational power required for dynamic programming is much smaller than for an exhaustive search. However, as the number of design variables in each sub-problem increases, the computational needs increase exponentially. Conversely, when the number of sub-problems increases, the computational needs only increase linearly. Therefore, dynamic programming works better for problems that can be separated into more sub-problems than those with more design variables in each sub-problem. Additionally, there are other limitations to dynamic programming. Sometimes it is impossible to solve even the smallest problem. In other cases, there are too many sub-problems to solve (Radford & Gero, 1987). There are no clear and general solutions for dynamic programming, as compared to other optimization methods (Radford & Gero, 1987). Because dynamic programming can solve problems with discrete, nonlinear, and stochastic decision variables, it is suitable for many of the design problems faced in architectural design. However, designers need to have a basic knowledge of mathematics and should be able to find the logic in each design problem. This may be another limitation since not all architects have sufficient training in math.

2.3.2.2 Genetic Algorithm (GA)

Since architectural design problems often have discrete, nonlinear, and stochastic decision variables with multiple objectives, optimization methods such as classical calculus, linear programming, and dynamic programming are not applicable to these optimization problems (Radford & Gero, 1987). Generally speaking, GA is more suitable for solving many architectural design problems than the other above-mentioned optimization methods, for the following three reasons: (1) Most architectural problems are complex and involve more than one design objective. Compared to differential calculus, linear programming, and dynamic programming, GA not only works well with discrete, nonlinear, and stochastic decision variables, but it also can handle problems with multiple design objectives. GA can either convert multi-objective optimization into single objective optimization by using a weighted sum of the objective functions, or identify a group of equally ranked solutions as the Pareto optimal set (Mackenzie & Gero, 1987). (2) A design problem must satisfy certain conditions in order to be solved by differential calculus, linear programming, or dynamic programming, which decreases the applicability of these methods to architectural design. However, GA has no such requirements. It can cover many design problems seen in architecture. (3) Modeling tools such as Rhino have been used widely in architectural design firms. "*Popular among students and professionals, McNeel Associates' Rhino modelling tool is endemic in the architectural design world. The new Grasshopper environment provides an intuitive way to explore designs without having to learn to script*" (Day, 2009). The integrated Rhino/Grasshopper program provides ready-to-use GA plugins – Galapagos and Octopus – for optimization. Galapagos is a single objective GA optimization tool, and Octopus is another GA tool for multi-objective optimization. Thus, this study adopts GA for design optimization.

GA mimics natural selection and the process of evolution (Holland, 1975). Holland's research (Holland, 1975) includes two parts. First, he described and explained the adaptive process of natural selection and evolution; second, he designed a software program to mimic the most important mechanism in the natural selection system using techniques such as inheritance, selection, crossover, and mutation.

GA is helpful in solving design optimization problems when there is a need to search through a large number of possibilities for solutions (Mitchell, 1998). Using mechanisms similar to those of natural selection in evolving individuals who adapt to an environment over time, GAs provide a robust search process; they have been used in optimizing complex and poorly-understood scientific and engineering problems (Gero & Louis, 1995) such as automotive design (Mahmoodabadi et al., 2013; Vidal et al., 2012), engineering design (Deb, 2012; Gen & Cheng, 2000), medicine (Chen & Chen, 2011), etc. In architecture, the applications for GA include structural design (Kociecki & Adeli, 2013), green building design (Attia et al., 2012; Wang et al., 2005), and space planning

(Jo & Gero, 1998; Rawat et al., 2012). Tools (Gerber et al., 2012) are being developed that employ GA in the generation of design alternatives.

The workflow of GA is described as follows. (1) In the beginning, GA randomly creates a large population (candidates for design options). (2) GA tests each candidate to see how good it is at solving the problem and assigns a fitness score. The fitness function is determined by the designer. For example, if the design objective is to minimize nurses' travel distance, the designer must define the fitness function, which in this case is the calculation of the nurses' travel distance in given hospital buildings. (3) GA selects a group of candidates with high fitness scores from the current population to act as parents. The chance of being selected is proportional to the fitness of the candidates. (4) GA crosses over the high-fitness parent solutions to generate child solutions. Since the parent solutions have high fitness scores, their child solutions are expected generally to have even higher fitness scores. (5) GA deletes the candidates with low fitness scores while maintaining the high score candidates; it also mutates a certain percentage of the candidates. These form a new generation. (6) GA repeats steps two through five until an optimal or at least a satisfactory candidate is found. Figure 2.2 shows the GA process.

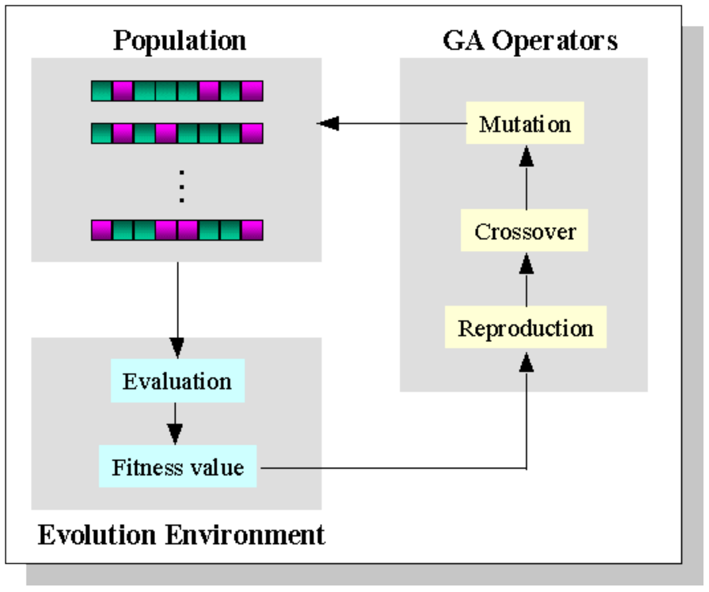


Figure 2.2. Evolution Flow of a Genetic Algorithm (Image Source: Liao & Sun, 2011).

2.3.2.3 Pareto Optimization

In general, there are two ways to solve multi-objective optimization problems: 1) convert the multi-objective optimization into a single objective optimization by using a weighted sum of the objective functions, often with arbitrary weights; and 2) search for the Pareto optimal set, which is more widely accepted for practical, multi-objective optimization because normally there is a set of optimal solutions (design options) rather than a single solution for multi-objective optimization (Mackenzie & Gero, 1987). In multi-objective optimization, Pareto frontier or Pareto optimal set or Pareto optimal front refer to a set of solutions in which none of the objective functions can be improved without making at least one other objective value worse off (Hochman & Rodgers, 1969). There can be multiple design solutions, but each must meet the following condition: any improvement of one objective (e.g., decreasing the nurses' travel distance in a unit) will degrade at least one other objective (e.g., decreasing the overall daylighting performance). These design solutions make up the Pareto optimal set.

Existing computational algorithms such as NSGA-II (Deb, 2002) and Strength Pareto EA (SPEA) (Zitzler & Thiele, 1999) have been devised generally to deal with and/or improve multi-objective optimization. Design tools such as Octopus (Vierlinger, 2014) utilize one such algorithm (SPEA-2) and integrate it with geometric modeling tools such as Rhino/Grasshopper.

2.4 An Improved GA for Design Optimization

Building performance simulation and genetic algorithm (GA) are powerful techniques for helping designers make better design decisions in architectural design optimization. However, they are very time consuming and require a significant amount of computing power. More time is needed when two techniques work together. This has become the primary impediment in applying design optimization to real-world projects.

2.4.1 The Time Complexity of Building Performance Simulation

With the growing demand for energy-efficient buildings, numerous building energy simulation tools have been developed. Green building standards such as LEED (USGBC, 2009) have been issued and implemented in many countries. A more sustainable and energy-saving type of design has been advocated by designers, engineers, and developers. The applications for building simulations include: building heating and cooling load calculation, daylighting calculation and reflective roof analysis, building energy management and control system design, building regulations, code checking, cost analysis, etc. (Hong et al., 2000). Although many simulation tools exist today, the optimization process, which involves iterative simulations, is inefficient and time consuming.

In addition to the improvement of multi-objective genetic algorithms from the general computing point of view, there is a need to research for a more efficient optimization method to make the use of simulations (e.g. energy simulations) more efficient during the optimization process.

Time and computational complexity analyses are used to evaluate a system's use of computing and time resources. Previous studies have shown that building simulation is a very time-consuming and labor-intensive process. For instance, in a high rise office building energy simulation study, the computer will need more than 30 hours to perform a one year hourly simulation, and the time required for monthly simulations will vary from 36 minutes to around 6 hours, using four desktop computers running at the same time (Ahn et al., 2013). Researchers point out that the time and resources needed for simulation are one of the reasons that simulation tools have not been fully embraced by architects (Shi, 2011). Architectural design firms have been using genetic algorithms as a way to search for the optimal design options to assist with the design decision making process (Besserud, Skidmore & Merrill, 2008; Claussnitzer et al. 2014). However, due to the tight project timeline, architects do not have the time and resources. Most optimization projects have a small population. Therefore, the optimal solutions are not guaranteed to find.

2.4.2 The Time Complexity in GA

A well-known limitation of GA is that it often takes a tremendous amount of search and evaluation iterations before reaching optimal solutions, especially when the task involves multiple objectives. A GA optimization process could take days to complete using today's conventional computers. In an optimization study of insulation usage and space conditioning load, it took GA 50 minutes to run 25 different designs and 2 hours and 18 minutes to run 100 different designs on a standard laptop (1.7-GHz CPU, 512-MB RAM) (Shi, 2011). In a study of green building design optimization using multi-objective GA, a computer with Windows XP (3.06GHZ Pentium-IV processor, 512 RAM) took 30 hours to run 29 solutions (Wang et al., 2005). This time-consuming aspect of GA could be a serious impediment to design automation, effectively discouraging designers from conducting more optimization studies. Both the number of generations and solutions in the above GA examples are too few to be of any practical use. On the one hand, when GA's execution time can be saved by using a smaller search space, global optimality is unlikely to be achieved. On the other hand, increasing GA search space to allow for a greater number of optimal solutions will result in a longer execution time, which in turn will prevent GA from being useful in the design process. Thus, reducing the run time of GA is essential to its use in design optimization.

2.4.3 Methods for Improvement

When a standard GA is used with simulation for optimization, a significant computer effort is expected (Renner & Ekárt, 2003). As a result, various methods have been researched for increasing the speed of GA. The literature on this topic can be divided into four groups. One involves improving the infrastructure, running GA on a faster machine, or using multiple machines working together. The Parallel Genetic Algorithm uses a group of cooperating computers to solve complex problems in less time (Muhlenbein, 1991). Cloud computing (Armbrust et al., 2010) has also been introduced as a way to reduce the computing time of optimization problems. Cloud services are now being used in many design firms to speed up the optimization process (Claussnitzer et al., 2014).

The second group of research in the literature investigates the construction of approximate simulation models, surrogate models that mimic the actual simulation model but use less computing power (Forrester et al., 2006). The computational cost of GA can be reduced by replacing computationally expensive fitness evaluations with cheap approximation models such as surrogate models (Ong et al., 2003). A review paper (Jin, 2005) points out that approximation models are beneficial when (1) the computation of the fitness function evaluation in GA is very time consuming; (2) no analytical fitness function model exists, the fitness function needed to be evaluate by human, for example, in music composition and art design; (3) the results of GA is not always consistent; and (4) the fitness landscape has multiple local optimal. Surrogate models are used to replace the accurate and time-commuting fitness evaluation. It is recommended to use surrogate model with the real/original fitness function to avoid having a false optimum in the surrogate model (Jin et al., 2000). The trade-off between the fidelity (approximation accuracy) and computational cost is shown in Figure 2.3 (Jin, 2011). Generally speaking, fitness evaluations with high fidelity are more time-consuming, and fitness evaluations with low fidelity are usually less-consuming. The user should balance between the needs for accuracy and available computational cost. Surrogate models are applicable to many operators in GA, for example, population generation, crossover (Anderson & Hsu, 1999), mutation (Abboud & Schoenauer, 2002), local search (Ong et al., 2006), global search (Simpson et al., 2001), and fitness evaluations (Buche et al., 2005; Jin, 2011). Multiple surrogate models can be applied together to a GA problem. For example, researchers (Zhou et al., 2007) combine global and local surrogate models to improve the speed of an evolutionary optimization. However, surrogate models require simulation knowledge, i.e. how to construct approximate simulation models, which architects usually don't possess. This inspired the present project in the reuse of offline or pre-simulation that architects already know how to conduct in the optimization process. The proposed methods are expected to be simpler for implementation than surrogate models because the original simulation models can be used in the optimization process.

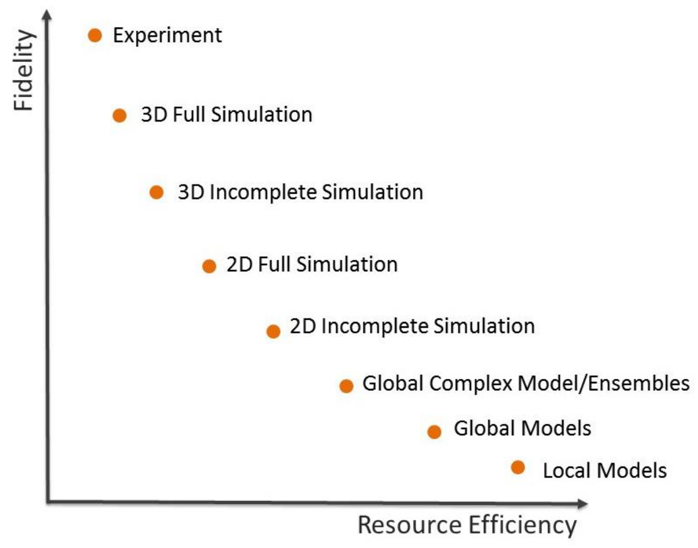


Figure 2.3. The Tradeoff Between Fidelity and Computational Cost (Adapted From Jin, 2011).

The third group of research suggests using simplified building simulation tools or reducing the number of simulation needs. Sentient building simulation system construct a result database of a portion of the search space and interpolate the results based on designer's need (Negendahl et al., 2014).

The fourth group in the literature focus on making the GA works more efficiently. Associative parametric models can be used to describe a large and complex geometric system with fewer variables, which would make the optimization process faster (von Buelow et al., 2010). The Evolutionary Divide & Conquer algorithm reduces the complexity of the problem by dividing a large problem into simple sub-problems (Valenzuela & Jones, 1993).

2.5 Design Knowledge Discovery through Optimization

Designers are not only interested in the optimum design solution to a specific problem, but are also care about the general rules or design knowledge that can be applied in any other citations (Radford & Gero, 1987). "The information generated by optimization often is generally applicable" (Radford & Gero, 1987, pp. 302). An approach to a previous optimization problem can be used to guide future designs, by designers or by computer expert systems (Radford & Gero, 1987). For example, we can obtain useful information by using design optimization to investigate the relationship between window size and daylighting performance. As the use of design optimization increases, there is a need to generate useful knowledge from the results of such optimization. Michie (1990) predicted that the next popular research area would be the use of machine learning tools for knowledge discovery in large data sets. Radford & Gero (1987) also pointed out that the future of computer-aided architectural design lay in advanced design systems that could generate and incorporate design knowledge by the systems themselves instead of depending on human judgements.

Knowledge discovery is the extraction of nontrivial information that is unclear, previously unknown, and has potentially useful information obtainable from data (Piateski & Frawley, 1991). Figure 2.4 depicts the components of a knowledge discovery system. The discovery methods used for searching and evaluating in a database make up the core of this system. The input includes: data from a database, domain knowledge, a data dictionary, and user-defined biases. The output is discovered knowledge.

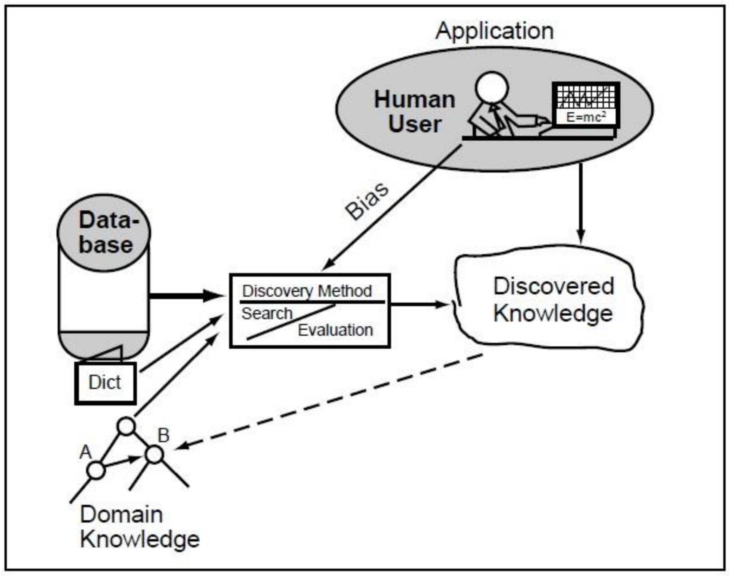


Figure 2.4 The Framework for Knowledge Discovery in Databases (Image Source: Frawley et al., 1992).

Architectural research has already addressed the use of optimization to discover rules or principles and generally apply them (Mackenzie & Gero, 1987). One kind of knowledge that can be discovered is the types of relationships that exist among design objectives (e.g., no conflict between two design objectives, linear relationships, etc., in Radford & Gero, 1987). Other kinds of knowledge include the relationships among decision variables, and among the objectives and decision variables. For example, the information generated can include the relationship between window size and the type of external environment, or between building performances and building forms; learning these relationships might otherwise require years of practical experience.

Figure 2.5 shows five different Pareto optimal sets for two design objectives. The relationships between the two objectives can be analyzed using optimization results.

(a) **No conflict exists between two design objectives**. The optimized performance for one design objective also tends to be the optimized performance for the other design objective. The short span of the Pareto set tells us that all performances are similar.

(b) **A heavily convex Pareto front**. When a heavily convex shape appears in the Pareto set, a balance exists between the two design objectives. The optimized performance in both design objectives is located at a point. However, improving performance in either direction will decrease performance in the other direction.

(c) **Linear relationship**. When the Pareto optimal is in a straight line or hyperbolic curve, there is a simple mathematical function between both objectives. We can use this rule to predict the Pareto performance without using optimization tools.

(d) **A heavily concave Pareto front**. When a heavily concave shape happens, there is a conflict between the two objectives. Trying to compromise both will result in bad performances in both objectives.

(e) **Partly convex and partly concave**. The relationship between the two objectives changes and there is no simple rule to explain the relationship.

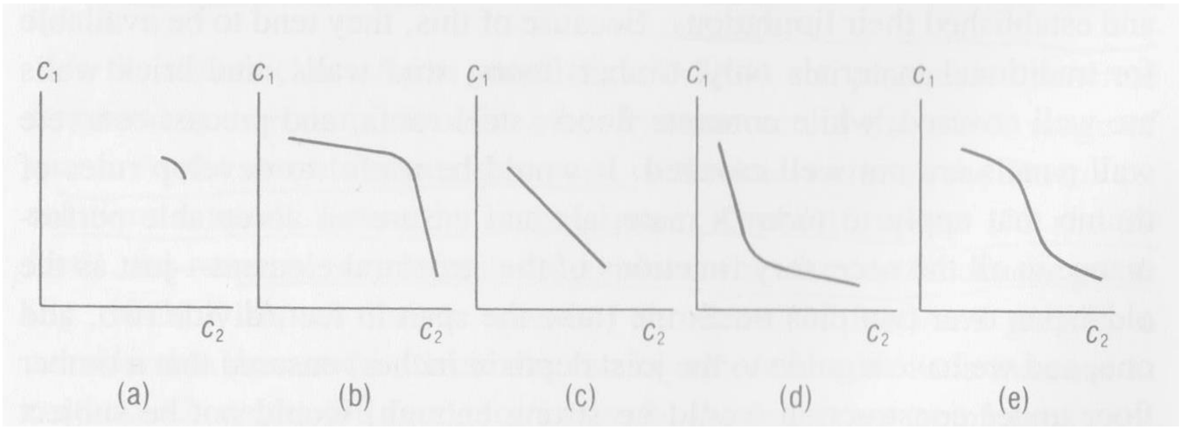


Figure 2.5. Patterns of Pareto Optimal Sets for Two Criteria (Image Source: Radford & Gero, 1987).

However, previous work in knowledge discovery through optimization for architectural design is mostly for demonstration purposes and the rules found from the simple prototypes are not new to designers, e.g. one should use pre-stressed concrete for minimum slab thickness (Mackenzie & Gero, 1987). What is more, existing knowledge discovery methods are based on manual and visual analysis of the results. Knowledge about the design and performance relationships can be learned through Pareto optimization and a manual analysis of the results (Radford & Gero, 1987). As the use of design optimization is rapidly growing, there is a need to generate useful knowledge from the results of optimization quickly and even automatically. In this study, a data mining, automatic approach is used to generate design knowledge based on optimization results, the learned knowledge is the relationship between the objectives (e.g. building performance) and the decision variables (e.g. building layouts).

2.5.1 Data Mining

2.5.1.1 What is data mining?

Data mining is the analytical step in the process known as Knowledge Discovery in Databases, or KDD (Fayyad et al., 1996). It involves many interdisciplinary fields of research, such as artificial intelligence, machine learning, statistics, and visualization (Chakrabarti et al., 2006).

The main purpose of data mining is to automatically extract useful information from large data sets and convert it into a format that is easy to understand. In large data sets, many patterns are likely to be uninteresting. However, when strong patterns appear, they can provide useful information about the present use and, more importantly, precise predictions for future use (i.e., what is happening now, and what will happen in the future when similar data sets are present) (Witten and Frank, 2005).

2.5.1.2 What are the benefits of data mining?

A large amount of data can be generated from design optimization, especially for multi-objective optimization, which may have many Pareto optimal set. It is difficult and time consuming to perform knowledge discovery and extraction using the traditional methods of data analysis, which largely depend upon manual examination and interpretation. What is more, manual inspection is highly subjective and may even be impossible in cases of very large data sets (Fayyad et al., 1996).

One of the advantages of data mining is information sharing, including sharing with other companies. This kind of information sharing may or may not be public (Clifton & Marks, 1996).

Data mining can provide a friendly interface between users, data, and the product (Fayyad et al., 1996) (in this study, this interface would be design optimization).

2.5.1.3 What are the disadvantages of data mining?

Because data mining can analyze data automatically and is used for information sharing, it may cause security issues, privacy issues, and misuses of information/inaccurate information (Clifton & Marks, 1996). One solution is to restrict access to data or control access to data (Clifton & Marks, 1996).

2.5.1.4 The use of data mining in the real world

The data mining for knowledge discovery approach has been utilized in many areas, such as:

- **Marketing and business**: Data mining is used in marketing and business to categorize customers and predict their behaviors (Berry & Linoff, 2004). Many companies use data mining and genetic algorithms for investment; however, most firms will not reveal their systems (Hall, Mani, & Barr, 1996).
- **Science and engineering**: Data mining has widely been used in many science and engineering fields such as electrical power engineering (McGrail et al., 2002), bioinformatics (Frank et al., 2004), and biomedicine (Zhu, 2007).
- **Medical**: Data mining is now used in electronic patient records for medical data analysis (Cios and Moore, 2002).

- **Spatial studies**: The purpose of data mining in spatial studies is to find useful information in large amounts of geospatial data (Miller and Han, 2009).
- **Visual Data Mining**: Compared to traditional data mining, visual data mining is quicker and more intuitive. It allows users to visualize the data mining process (Keim, 2002).

Despite the popular use of data mining techniques in the real world, there is a lack of study in the application of data mining to architectural design. The existing literature includes research on using data mining to discover patterns that can help improving building design in energy efficiency (Kim et al., 2011); and predict building performance in building simulation exercises (Morbitzer, Strachan, & Simpson, 2003). More studies are needed, especially in the application of data mining to architectural design.

2.5.2 Machine Learning

Machine learning serves as a technical support for data mining (Witten & Frank, 2005). It is a subset of artificial intelligence, which is the study and creation of intelligence (Poole & Goebel, 1998). It is used to understand, explain, and predict data sets (Witten & Frank, 2005). Witten and Frank define learning as follows:

> *"Things learn when they change their behavior in a way that makes them perform better in the future"* (Witten and Frank , 2005, pp. xxiii).

Intelligent mechanisms can be constructed by: (1) interviewing an expert or experts in the relevant field; or (2) using particular case studies to discover and generalize knowledge (Quinlan, 2014). Many knowledge-based mechanisms have been constructed using these methods (Michie, 1987, 1989; Quinlan, 2014).

In this study, data mining supported by machine learning is used to assist in the process of knowledge discovery, based on design optimization.

CHAPTER III

METHODOLOGY OVERVIEW

This chapter presents the methodology used in this study. It begins with a basic definition of research, and follows with the principles and guidelines of this project's specific research methodology: design science research methodology (DSRM).

3.1 Introduction

The definition of research, as described by Leedy and Ormrod (2005), is:

> *"… a systematic process of collecting, analyzing, and interpreting information (data) in order to increase our understanding of a phenomenon about which we are interested or concerned."* (pp.2)

Quality research should originate with a question, hypothesis, or problem, proceed with an extensive literature review on related research, continue thereupon with a clear articulation of the goal of the research, and finally follow a specific research design plan or procedure that will assure the work's logical consistency, implementability, and plausibility (Haber, 2010). The main research question is often divided into sub-problems to be solved separately (Leedy & Ormrod, 2005). Proposed methods or solutions should be validated by case studies or mathematical proofs, and the design process should be clearly described and explained so that other researchers can reproduce and verify the process (Hevner et al., 2004). Finally, research requires the collection and interpretation of data, a definition of the project's limitations, and an assessment of the research results (Leedy & Ormrod, 2005).

3.2 Design Science Research Methodology (DSRM)

Design Science Research Methodology attempts to improve the functional performance of the designed artifact. It is widely used in many disciplines such as information systems, computer science, engineering, etc. (Vaishnavi & Kuechler, 2004). It seeks to provide innovative ideas, solutions, and products for problem solving through analysis, design, evaluation, and implementation. Hevner et al. (2004) list seven guidelines and Peffers et al. (2007) provide six steps for DSRM. The seven guidelines for DSRM (Hevner et al., 2004) are as follows:

Guideline 1: The end product of design science research should be an innovative and purposeful artifact such as a method, model, construct, or instantiation.

The end product of this study is computer program prototypical tools and methods created using computer modeling, optimization, and building simulation methods for improving design optimization and optimization-based design knowledge discovery.

Guideline 2: The main purpose of design science research is to develop and implement technically-based solutions to solve important and specific problems.

The main research objective of this study is to provide methods to improve the time complexity issue in building simulation and optimization, and to fill up the gap in research on design knowledge discovery through optimization.

Guideline 3: Evaluation is an important part of this research process. An artifact should be evaluated rigorously to demonstrate its value, efficiency, and use.

In this study, the proposed methods were validated by mathematical proofs when needed, and the process was demonstrated by a simplified case study.

Guideline 4: Design science research should provide new and interesting discoveries in one or more of these areas: design artifacts, design knowledge, and/or design methodologies.

This study provides new discoveries in the area of design methodologies.

Guideline 5: The development and assessment of a designed product should be conducted rigorously.

Rigorous methods have been applied in both the construction and evaluation of the methods and prototypes in this study.

Guideline 6: Design should be seen as a problem solving process searching for a desired solution.

This study begins with identifying the problems in the present architectural design process, and then trying to solve the problems with effective methods.

Guideline 7: The design outcomes should be presented to both technical and management personnel.

Several parts of this study have been presented to technical audiences. More details about the publications of this study can be seen in Section 3.7 in this chapter. In order to promote the proposed prototypical tools to industry organizations, presenting this study to management – oriented audience will be future work.

Peffers et al. (2007) define the six steps in design science research as follows:

> *"...problem identification and motivation, definition of the objectives for a solution, design and development, demonstration, evaluation, and communication."* (pp. 46)

This study was carried out by following the above-mentioned seven guidelines and six steps below. The implementation of the six steps is discussed below.

3.3 Step 1 - Problem Identification and Motivation

This study began by identifying an existing problem, which later was confirmed by conducting a thorough literature review of related research.

3.3.1 Problem Identification

A problem should be addressed before one begins conducting research. Research problems usually come from personal experience, something the researcher is interested in and of which they have knowledge (Tuckman & Harper, 2012). The research problems addressed by this study - the complexity of architectural design, limitations of the traditional design process, the time consuming nature of building performance simulation and optimization, and the lack of research studies focused on discovering knowledge through optimization – were obtained from this researcher's personal experience and experiments. Please refer to Chapter 1, Section 1.1 for a more detailed discussion of the research problem addressed in this study.

3.3.2 Literature Review

An extensive literature review on related research is presented in this study; it covers the following main topics: design optimization methods, Genetic Algorithm (GA), building simulations, time spent on GA, time spent on building simulations, methods for time savings on the use of GA, time savings in building simulations, surrogate models, offline simulation, Divide & Conquer techniques, machine learning, and knowledge discovery through optimization.

The use of computational design optimization methods in building design has been discussed since the 1980s. A large and extensive body of work has been presented by John Gero (Gero & Louis, 1995; Jo & Gero, 1998; Mackenzie & Gero, 1987; Radford & Gero, 1987). However, in the area of building design, most of the work relating to simulation-based optimization is in the area of building system design (HVAC) and structural design. The literature review indicates that more research should be focused on simulation-based optimization in architectural design.

3.4 Step 2 - Objective of the Solution

The term “research objective” describes what a study hopes to accomplish. One or more research questions should be developed after the researcher has narrowed down the research problem (Tuckman & Harper, 2012). Good research questions are specific, clear, refer directly to the research problem, reflect improvement, and address the focus participants (Tuckman & Harper, 2012).

The research objectives of this study include providing methods and prototypical tools to: (1) improve the efficiency of optimization by reducing simulation needs when building simulations are incorporated into design optimization; (2) study how design knowledge can be used to speed up the optimization process by reducing the computational complexity of the design problem; (3) find a design knowledge discovery method towards automatically generating useful correlations and causal relationships through optimization.

More discussion about the research objectives and research questions for this study are in Chapter 1, Sections 1.3 and 1.4.

3.5 Steps 3 & 4 – Design, Development, and Demonstration

Design, development, and demonstration are combined through prototyping in this study, and presented in Chapters 4 and 5.

3.5.1 Prototyping

This study provides a prototyped proof of concept by using computer modeling and building simulation techniques. Prototyping produces an early version of a solution in order to test a new hypothesis or design (Budde et al., 2011). Experiments with early working versions can provide valuable information for use in future applications. The prototyping process provides a channel of communication between users and designers (Budde et al., 2011).

A simplified nursing unit layout design is used as a case study in this study in order to verify the proposed methods. More complicated and realistic projects can be experimented in the future to improve the methods.

> "*Design-science research often simplifies a problem…Such simplifications …may not be realistic enough to have a significant impact on practice but may represent a starting point… As means, ends, and laws are refined and made more realistic, the design artifact becomes more relevant and valuable.*" (Hevner et al., 2004, pp. 88-89)

3.6 Step 5 - Evaluation

Evaluation in DSRM involves the assessment of design methods and outcomes (Pries-Heje, Baskerville & Venable, 2008). Hevner et al. (2004) emphasize that researchers should rigorously evaluate the value, efficiency, and property of their research. March and Smith (1995) points out that evaluation is one of the two most important activities in DSRM, after the building of the artifact. Pries-Heje et al. (2008) provide a strategic, two-dimensional evaluation framework. One dimension, ex ante (prior) versus ex post (after), concerns the time a researcher takes to evaluate, and offers the possibility of evaluating either prior to or after the artifact has been built. The other dimension, naturalistic versus artificial, provides the opportunity to evaluate a real artifact according to its use by real users, or proving/disproving a hypotheses or artifact solely in theory.

This study evaluates the proposed methods through artificial evaluation techniques (mathematical proofs) before undertaking the task, and then evaluates the methods a second time through the naturalistic method (a case study).

The evaluation methods are presented and discussed in Chapters 4 and 5. Further discussion regarding the reliability and validity of this study can be found in Chapter 6.

3.7 Step 6 - Communication

Thus far, one manuscript about using the proposed offline simulation method to improve genetic algorithm (GA) has been published and presented at the 2014 Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA) (Su & Yan, 2014). Another manuscript about the combined use of offline simulation and

Divide & Conquer (D&C) to effectively improve architectural design optimization has been accepted by the journal Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AI EDAM) (Su & Yan, 2015). A third manuscript about the proposed computational methods for creating and improving a closed loop of design optimization and knowledge discovery in architecture is under review by a journal.

CHAPTER IV

DESIGN KNOWLEDGE ASSISTED OPTIMIZATION IMPROVEMENT*

4.1 Introduction

The first part of this study introduces a fast and extensive creation and evaluation of design alternatives through a process of searching, simulating, and optimizing, in order to meet specified architectural design objectives. Compared to the existing optimization workflow, it aims to provide a quick search by reducing simulation runs and computational complexities for optimal design using architectural domain knowledge. The improved method used in this research not only provides a faster optimization, but more importantly, it enables a much larger search space within the same amount of time that offers a better chance of finding the optimal design. The fast design optimization method will make it more applicable for architects.

Two techniques, namely offline simulation (Su & Yan, 2014) and Divide & Conquer are proposed to reduce the computer run time in optimization based on architectural domain knowledge.

4.2 Offline Simulation

In existing optimization processes, if building simulation programs are involved, simulations must be performed every time when building parameters (chromosomes in GA) change to form a new GA solution. The result of each simulation is used in the GA fitness function to produce another generation of solutions. This process continues until

* Part of this chapter is reprinted with permission from "Improving Genetic Algorithm for Design Optimization Using Architectural Domain Knowledge" by Su, Z., Yan, W., 2014, *Proceedings of the Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA), Copyright 2014 by ACADIA.*

GA finds the optimal solution or reaches a pre-defined calculation time limit. As a result, the simulation program must run until GA finds the optimal solution. The number of simulations that need to be conducted is equal or proportional to GA's population size.

In this study, the idea of offline simulation is introduced to reduce the number of simulations in the early stage of architectural design. Terms such as "offline", "online" or "real-time" have been widely used in engineering and scientific areas for physically-based animation, physical system modeling, and simulation. The terms "online" or "real-time" simulation refer to a computer model simulation tool that works at the same rate as the actual physical system in real time. In other words, it will take the same amount of time in the real world for calculation in a real-time simulation. Offline simulation tools usually work within a fixed time (Gole, 2000).

In the domain of building design, "offline" and "online" concepts have been used in building system control design (Yu & Dexter, 2009; Coffey, 2012, 2013; Corbin et al., 2013; Hu & Karava, 2014). In the above literature, "offline" means that the optimizer is not connected with the server that controls the physical system (e.g. the physical building is not controlled by the optimizer). The term "online" or "real-time" refers to the optimization process that connects the optimizer with a server-client framework controlling the building HVAC system (Corbin et al., 2013). Hu and Karava (2014) simulated a mixed-zone building using an offline Model Predictive Control (MPC) framework with Particle Swarm Optimization (PSO) as an optimizer. Coffey (2012, 2013) used offline optimization with building simulation tools to approximate MPC with lookup tables for optimal control setups. The results from offline optimization - the lookup tables – can be used in real-time building control simulations so that the online optimization does not need to run at each time step. With offline optimization, online problems are easier to tackle and simulations can be faster (Coffey, 2012, 2013). Similarity and difference lie between the above mentioned “offline” methods and the way “offline” was used in this study. The similarity is that the results from “offline” simulation can provide rules or knowledge for the “online” system later. The difference is that in the work mentioned above, the offline optimization is separated from the real-time control of HVAC systems; in this study, the offline simulation is separated from the optimization process and the simulation results can be used and re-used in the optimization process later in the early stage of architectural design.

In conclusion, when the design problem is decomposable into sub-problems, where some sub-problems can be solved offline, offline simulation/optimization methods can be used to tackle part of the problem by making the original optimization problem easier to solve, reducing the computational time, and increasing the search space within the same amount of time.

Compared to real-time simulation, offline simulation in this study refers to a computer simulation model that can execute at a time different from that of the general GA optimization process. In order to save time, building simulation is separated from the GA optimization process, all required simulations are conducted in advance, and simulation results are reused whenever appropriate in the GA’s fitness evaluation process. In other words, the correlations between building performance and decision variables can be obtained from offline simulations. These correlations can then be used and reused in the GA fitness function. This way, simulations do not need to be repeated in GA, and, as a result, a significant amount of time can be saved.

4.3 Divide & Conquer (D&C)

The idea of the Divide & Conquer (D&C) technique is to divide a large problem into manageable sub-problems. The solution to the original problem can be obtained by combining the solutions of the sub-problems. This technique has the potential to be used in any complex optimization situation (Valenzuela & Jones, 1993). There are a number of different D&C methods already being used in the areas of mathematics and computing. For example, dynamic programming solves sub-problems and combines the knowledge gained through the process to reach the final solution; this process can be conducted without knowing how to decompose the original problem (Bellman, 1956). In the area of GA, Potter and De Jong (1994, 2000) presented an evolutionary D&C technique called Cooperative Coevolution. In Cooperative Coevolution, the subcomponents are described as a collection of cooperating species. The individual species are coevolved and solved independently in order to ultimately solve a complex problem.

Prior work on engineering design has used multi-level hierarchical design optimization frameworks to solve large and complex design problems (Papalambros, 2002) and a problem decomposition method called Analytic Target Cascading (ATC) (Kim, 2001; Kim et al., 2001). The benefits of the approach include efficiency, robustness, and organization.

In building design, ATC has been extended to thermal and HAVC design (Choudhary et al., 2003; Choudhary, 2004; Choudary & Michalek, 2005). Different performance analysis goals may be involved in a complex design problem that requires unique and separate analysis tools in order to successfully achieve the results (Choudhary et. al., 2003). The decomposition permits sub-problems to be constructed separately, allows appropriate simulation and optimizer tools to be chosen from the tool repository, and supports optimization based on the individual performance objectives of the sub-problems (Choudhary, 2004). In terms of computational expense, simulation tools take the majority run-time in this hierarchical framework, and therefore cheap simulation models are recommended (Choudhary, 2004). Expensive models can be substituted with computationally cheap surrogate models (Papalambros, 2002).

In Choudhary's study (2004), the problems are decomposed into sub-problems using Object Decomposition (Wagner, 1993) and Aspect Decomposition. Object Decomposition divides the problems by physical components such as zones and parts. Aspect Decomposition separates the problems by disciplines. While the idea of decomposing a problem into sub-problems is similar to the Divide & Conquer method, this study demonstrates a sequential object decomposition approach (decomposing the design problem into the layout optimization for patient rooms in sequence) that is applied to the process of Generic Algorithm in the design of the spatial layout.

The idea of using the decomposition method to reduce the simulation time can also be seen in the work by Welle et al. (2012). The authors developed an automated method to decompose and recompose a building model for climate-based daylighting simulation, with great simulation time saving and simulation results that are acceptable. This method decomposes a building into many spaces. Each space is then evaluated for distinct features and whether it should be simulated. A cloud computing platform is then used for the simulations, and the results can be weighted and used for other spaces with similar characteristics. This decomposition method is used for daylighting simulation but not for optimization purpose as in the present study.

The specific D&C method used in this paper, described by Watson (2002), is based on a decomposable and separable design problem. It is straightforward, easy to understand, and easy to conduct from the perspective of an architect. If the variables are independent from each other, e.g. in simulations that usually use simplified models of the real-world design problems, design problem with a large number of populations can be broken into several sub-problems, each with a smaller population, and solve them separately with less time.

Suppose there are N variables in a complex optimization function F, and each variable has K possible values. If these variables are independent of each other (which means the value of one variable is not affected by the values of others), this problem can be broken down into N separable sub-problems, each with only one variable. The function F can be expressed as the sum of N individual functions: $F(V_1,V_2,\ldots,V_N)= F_1(V_1)+ F_2(V_2) +\ldots+ F_N(V_N)$. To maximize/minimize the function F, the functions F_1 to F_N can be maximize/minimized separately and with less computing time as described below.

In GA, the computing time needed to solve the problem increases as the number of populations of candidate solutions increases. The D&C method decomposes a GA problem with a large population into several sub-GA problems with fewer populations. For a standard GA, the search space S standard is equal to the population size, which is:

$S_{standard} = K^N$. For a GA using D&C, the search space is: $S_{D\&C} = N\cdot K$. $S_{D\&C} = N\cdot K << S_{standard} = K^N$, especially when K and N are large.

When the D&C method is applicable, the search space is much smaller. The algorithm can be finished with less computing time or the search space within the same timeframe can be enlarged. One requirement for the applicability of D&C is that the decision variables are independent of one another. In certain architectural design optimization problems, this requirement can be satisfied.

4.4 Case Study

4.4.1 A Simplified Case Study

Validations of the above mentioned methods are presented in the context of a simplified case study of parametric form-finding for a children’s unit design with two design objectives: minimizing the nurses’ travel distance and maximizing the daylighting performance in patient rooms.

The reason for selecting a simplified case study is that it is useful to validate the methods and workflow. For a particular optimization problem how does one verify that the results are the real optimized solutions, especially in healthcare design, when design problems are usually very complex and involve multiple design objectives? For these kinds of complex problems, the optimized solutions cannot be foreseen in order to verify the results. However, a complex problem can be converted into a simplified problem to validate the optimization method and workflow. The optimization method for the simplified problem can then be verified by converting the problem into an analytically solvable problem or by using exhaustive search. In contrast, it is generally impossible to use exhaustive search for a complex problem because the search space is too big and it may take a long time for computers to calculate.

4.4.2 Design Objectives

Validations of the method are presented within the context of a case study for parametric form-finding in a nursing unit design with two design objectives: (1) to minimize nurses' travel distance from the nurses' station to each patient room; and (2) to maximize the daylight illuminance in all patient rooms using LEED standards (healthcare supplement; USGBC, 2009) as a reference. Both design objectives are used as sample EBD principles in this case study.

In any study related to the behavior and working efficiency of the nursing staff, one of the most important variables is the distance that a nurse is obligated to walk in a hospital. Walking has been identified as a major time-consuming activity for nurses, and evidence from previous studies suggest that the time saved by walking can be turned into more time spent on patient care activities (Zimring et al., 2004). Individual nurses across all study units travel between 1 and 5 miles per 10-hour daytime shift. Average travel distance ranges between 2.4 and 3.4 miles with a median of 3.0 miles per 10 hours (Hendrich et al., 2008). Unnecessary walking may lead to time waste and add to fatigue and stress. In the case study, the objective is to minimize the total nurses’ walking distance.

Daylight is an essential element for patient wellbeing. Research shows that patients in sunny rooms feel less pain and stress, and take less medication as compared to patients in rooms with less sunlight (Walch et al., 2005). This case study uses the LEED standard as a reference to construct the fitness function for the daylight illuminance level calculation. The objective is to achieve maximum daylighting under LEED requirement. LEED requires 75% or more of the perimeter area to achieve a daylight illuminance level between 110 lux and 5,400 lux. Therefore, in this case study the daylight illuminance in every inpatient unit layout design is expected to meet this requirement. The daylighting performance is evaluated by a building simulation tool. The use of LEED daylighting requirement as a reference to construct the daylighting performance fitness function can be found in (Rahmani Asl, etc. 2013).

It's worth noting that the travel distance and daylighting performance were selected as the sample design objectives in this study among all quantifiable design objectives involved in nursing unit design, which include construction cost, energy consumption, equipment placement, and others. These two objectives were chosen because, 1) Travel distance and daylighting are two of the most important concerns in nursing unit design; and 2) Travel distance and daylighting may have conflicting attributes because daylighting performance may decrease when travel distance is reduced (depending on the floor plan design), which needs to be confirmed by case studies. Both objectives can be substituted with other objectives since the focus of this study is developing and testing the new optimization methods rather than studying the particular design objectives.

The current tools and techniques used are introduced in Section 4.4.3, and the measurable baseline model is discussed in Section 4.4.4.

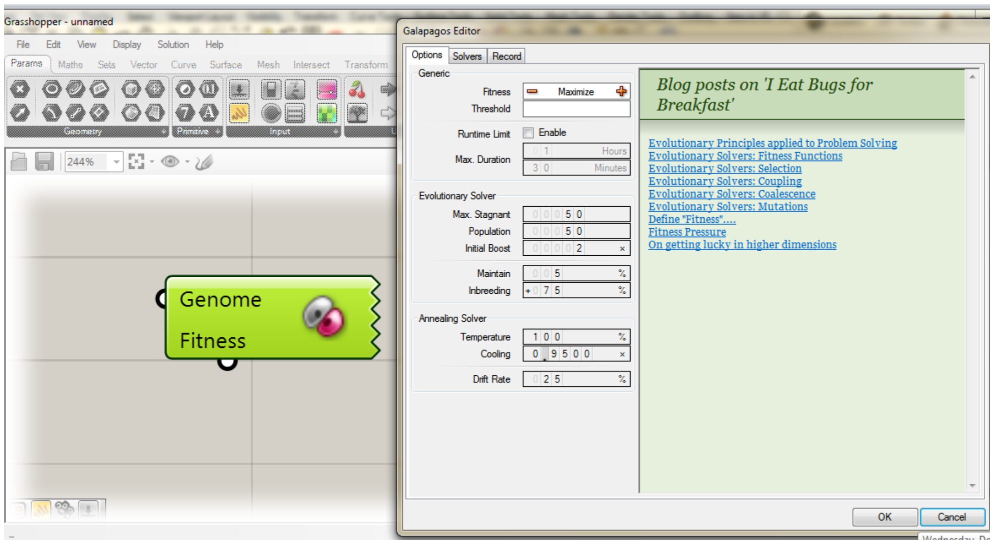
4.4.3 Design Platforms and Tools

Parametric models can generate a very complex building geometry with a number of variables, rules, and constraints that are defined by the designers. Several design software tools offer parametric modeling features. Of these software options, the integrated Rhino/Grasshopper program has widely been used because of its powerful modeling capability, intuitive interface, and abundance of plug-ins that greatly expand its functionality. It also provides a ready-to-use GA plugin – Galapagos –which can be used for optimization. Hence, this case study uses Rhino/Grasshopper as the design platform. The following is a complete list of current tools and techniques used in this case study:

(1) **Rhinoceros** (Rhino), a NURBS (a type of curves) based 3D modeling program.

(2) **Grasshopper**, a visual programming plug-in for parametrically editing models in Rhino. A user doesn’t need to have programming or scripting knowledge to use Grasshopper.

(3) **Galapagos**, a single objective optimization GA tool in Grasshopper. A user only needs to define the genome and fitness (Figure 4.1 left), and modify the settings such as population size and mutation rate in Galapagos (Figure 4.1 right).



Generic

Fitness	Maximize
Threshold	
Runtime Limit	☐ Enable
Max. Duration	01 Hours
	30 Minutes

Evolutionary Solver

Max. Stagnant	00050
Population	00050
Initial Boost	00002 ×
Maintain	005 %
Inbreeding	+075 %

Annealing Solver

Temperature	100 %
Cooling	0.9500 ×
Drift Rate	025 %

Blog posts on 'I Eat Bugs for Breakfast'

Evolutionary Principles applied to Problem Solving
Evolutionary Solvers: Fitness Functions
Evolutionary Solvers: Selection
Evolutionary Solvers: Coupling
Evolutionary Solvers: Coalescence
Evolutionary Solvers: Mutations
Define "Fitness"....
Fitness Pressure
On getting lucky in higher dimensions

Figure 4.1. The Screenshot of Galapagos in Grasshopper.

(4) **Octopus**, a multi-objective optimization GA tool in Grasshopper.

(5) **DIVA**, a thermal and daylight simulation plugin for Grasshopper. DIVA conducts daylight analyses for Rhino models through Radiance and DAYSIM daylight simulation engines. Together with Galapagos, they have been used in previous work for daylight analysis and optimization in the early architectural design process (e.g. Gallas & Halin, 2013; Portugal & Guedes, 2012).

It should be noted that the methodology developed in this study is not limited by the tools of choice. The concepts and principles can be generally applied across different platforms.

4.4.4 Baseline Model

This section presents a simplified yet representative case study used to validate the two methods proposed: offline simulation and D&C. The design problem is to find the optimal nursing unit layout that would allow for the least travel distance for nurses and a daylight illuminance that meets the LEED standard. The original problem was simplified as follows: on a 15 by 15 grid with a total number of 225 cells (discretized spatial units), a central nurses' station with a size of 29 ft by 29 ft is located in the center, and there is an 8 ft corridor outside of the central nurses' station. Twelve patient rooms need to be placed on the rest of the grid so that the final optimal layout will have the minimal travel distance from the nurses' station to the center of each patient room, and optimal daylight performance in all patient rooms based on the LEED standard (healthcare supplement; USGBC, 2009). Figure 4.2 shows the constant parameters in a possible layout solution. The number 12 is selected as the number of patient rooms for two reasons: 1) numbers such as 12, 18, and 24 are commonly used as the number of patient rooms in in-patient unit design because these numbers can be divided by 2, 3, and 4. It is easier and fair for nurse-patient assignments because all nurses (2, 3, or 4 nurses) can have an equal number of patients to take care of. 2) The number 12 is simple yet representative. A larger number will increase the computer calculation time, and a smaller number may not be sufficient. The study here mainly focuses on developing and testing the methods thus the selected parameters and their values are only for experiments. Therefore, the layout of the nursing unit is defined by a set of parameters, restrictions and objectives. The parameters include constants and variables. The constants include: (1) a 15 by 15 grid with a total number of 225 cells, each measuring 15 ft by 15 ft, as possible room spaces; (2) a central nurses' station represented by the blue square, sized 29 ft by 29 ft; (3) an 8 ft corridor outside of the central nurses' station; (4) the city of Boston as the location of the building in DIVA; and (5) a window size of 6 ft by 6 ft in every patient room. Figure 4.2 shows the constant parameters in a possible layout solution; the small red (dark) squares represent the patient rooms.

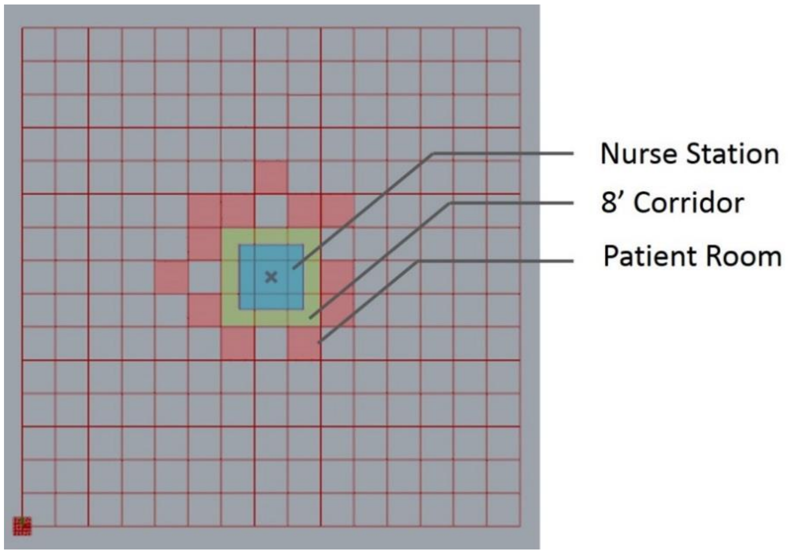


Figure 4.2 The Constant Parameters in a Possible Layout Solution.

The variables are the locations of the 12 patient rooms. A patient room can be located in any cell, with the following restrictions: (1) a patient room cannot overlap with any other patient room, nurses' station, or the corridor; and (2) in order to introduce natural light into each room, a patient room cannot be surrounded by other rooms in all four directions. In other words, at least one of the four cells surrounding each room should be vacant for a window opening.

In this case study, two objectives are converted into a single objective by using a weighted sum of the objective functions with pre-defined (architects' subjective) weights. For both travel distance and daylight illuminance, 100 points are given as the highest fitness score. The total nurse travel distance is calculated as the sum of the distances from the center of the central nurses' station to the centers of all the patient rooms. The calculation of the nurses' station to patient room distance and the total distance are simplified in the case study. In a practical situation, nurses' walking distances are usually affected by the distance from the nurses' station to the patient bed, following each nurse's travel path and affected by the order of that nurse's activities. Neither daylight nor travel distance sub-fitness functions should be linear in the overall fitness function. For daylight illuminance, losing the same amount of daylight affects more a darker room than a brighter room (Rutten, 2011) (see Figure 4.3). The same additional increment of travel distance makes a nurse feel much more fatigue, if that nurse has traveled a longer distance than someone who has just started a shift (see Figure 4.4).

A fitness score of 100 is given to the nursing unit layout with the shortest total travel distance possible, 388 feet (see Figure 4.5, left), and 0 is given to the layout with the furthest total travel distance possible, 1,708 feet (see Figure 4.5, right). For any nursing unit layout solution, the fitness score for the travel distance is defined as follows. (The power of 2 was chosen in the case study for simplicity of calculation.)

$$Fitness_{\,travel\ distance} = 100\left(1-\left(\frac{Distance-388}{1708-388}\right)^2\right),\quad 388 \leq Distance \leq 1708$$

Daylight illuminance is evaluated by the percentage of sensor grid points in a room achieving a daylight illuminance level between 110 lux and 5400 lux. In this case study, the above percentage is 89% if all rooms face south and 70% if all rooms face north (and in between for east and west). Fitness values of 100 and 0 are assigned to rooms with south-facing and north-facing windows, respectively. For any nursing unit layout solution, the fitness score for daylight illuminance is defined as follows:

$$Fitness_{daylighting} = 100(Sqrt\left(1 - \frac{89\% - DaylightPercentage}{89\% - 70\%}\right))$$

$$70\% \leq DaylightPercentage \leq 89\%$$

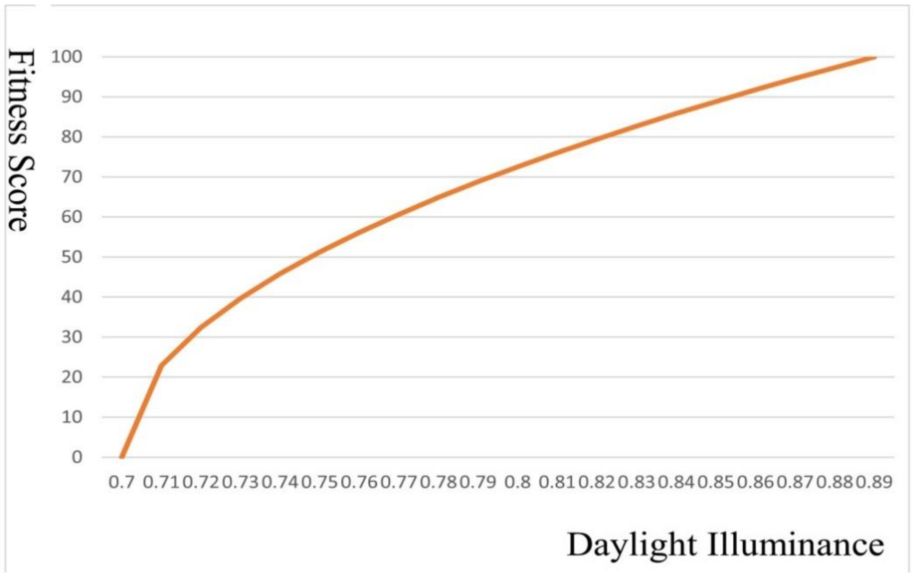


Figure 4.3 The Daylight Illuminance Sub-Fitness-Function is Nonlinear.

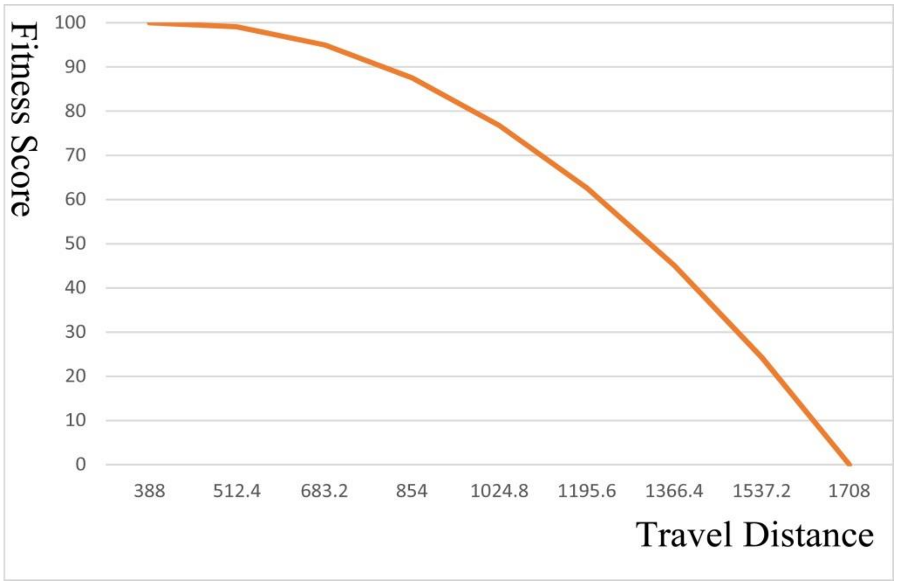


Figure 4.4 The Travel Distance Sub-Fitness Function is Nonlinear.

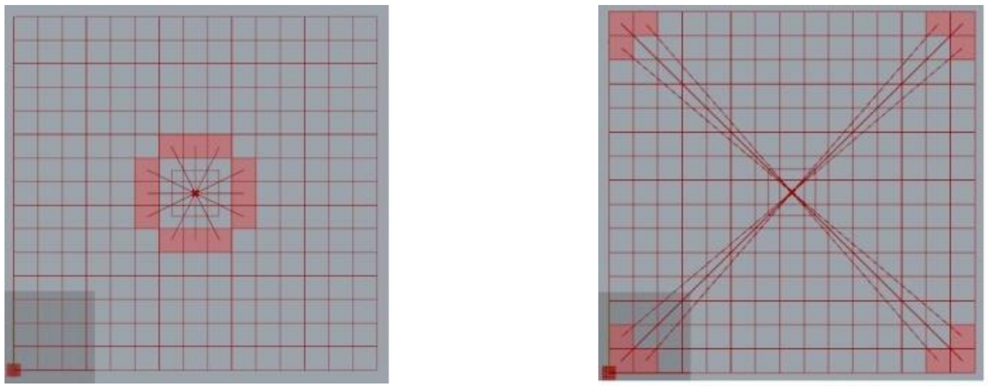


Figure 4.5 Unit Layouts with the Shortest Travel Distance (Left) and Furthest Travel Distance (Right).

Chromosomes. Coding for the problem is needed before proceeding with GA. The design problem may be represented as a series of parameters (genes). These genes connect together to become a string of values (chromosome). In my simplified study, the genes are the individual location of 12 rooms, and the chromosomes are the layout of the 12 rooms combined. Each chromosome can be represented by a bit string – an array of data structure containing bits. GA will code each design option, which is a layout of the 12 rooms in this way.

Crossover. Using the mechanisms of crossover, offspring are reproduced by selecting two parents with high fitness scores from the last generation, and recombining their genes. Since their parents have high fitness scores, it is expected that the offspring will have high fitness scores too. The offspring will keep parts of one individual's (parent) chromosome, and take the remaining parts of the chromosome from the other individual (parent) to form a new full-length chromosome (child; Beasley & Martin, 1993; Figure 4.6). When crossover happens in this simplified study, GA will select two design options with high fitness scores and swap their chromosomes, which are the layouts of the 12 rooms between the two design options, and form new design options as offspring. For example, the locations of the first 6 rooms in design option A will be switched with the locations of the corresponding rooms in design option B to produce offspring.

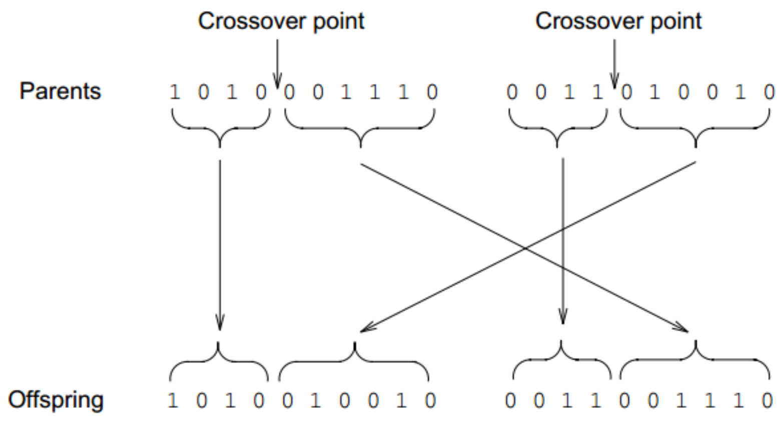


Figure 4.6 Single Point Crossover. Image Source: Beasley & Martin, 1993.

Mutation. The mechanism of crossover is designed to improve the solutions in the next generation. However, it will tend to decrease the bio-diversity. Mutation is brought in to increase the diversity in a population. It will give each gene a small probability to alter the gene. The user of GA can define the rate of the probability. Figure 4.7 shows the fifth gene being mutated in a chromosome. In my study, a single point mutation to a chromosome (one possible layout of the nursing unit) means to change one gene (e.g. the location of one room, depending on the actual implementation of the Galapagos GA software) in the chromosome.

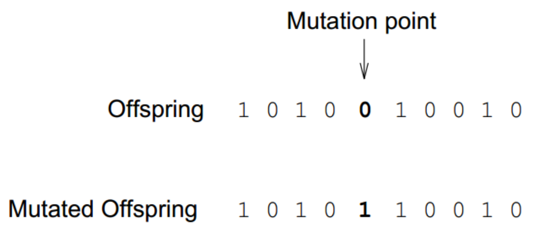

Mutation point
Offspring 1 0 1 0 0 1 0 0 1 0
Mutated Offspring 1 0 1 0 1 1 0 0 1 0

Figure 4.7 Single Point Mutation. Image Source: Beasley & Martin, 1993.

According to LEED IEQ Credit 8.1 “Daylight and Views-Daylight,” daylight illuminance simulations should be conducted under clear sky conditions at two different times. In this research, those times were 9 a.m. and 3 p.m. on September 21. Thus, in DIVA, "clear sky with sun" and "illuminance" were selected for the sky conditions. However, DIVA can only conduct simulations at a specified time point in each run. In order to satisfy the LEED requirement, the optimization process must run two separate times, one on September 21 at 9 a.m., and the other at 3 p.m. Although the setups of both optimization processes have the same objectives and parameters in DIVA (except the solar time), the results may be different because daylighting conditions differ. Both a 9 a.m. and 3 p.m. calculation were performed in this case study.

The DIVA daylight simulation accuracy increases if more sensors are used in the patient rooms. However, the more sensors it uses, the more computing time it needs. Consequently, designers should balance the need for accuracy with the associated computing time, based on the project’s requirements. In the baseline model used in this research, one hundred sensors were evenly distributed in each room (1,200 sensors in 12 rooms total) at the level of desk height. All simulations were run on a standard laptop

(ThinkPad series, Windows 7 64-bit, Intel(R) Core(TM) i5-2520M CPU @ 2.50 GHz, 8GB memory). In Galapagos, the population size per generation is 50, initial boost is 2, the inbreeding factor is 75%, and the maintaining factor is 5%. For this particular study, a single run of the daylight simulation in DIVA required approximately one to two minutes to complete. However, when DIVA is associated with Galapagos in the optimization process, it can be very time consuming; each solution requires a simulation. In this experiment, 78 hours of DIVA run time + Galapagos could only calculate 13 generations of GA, and the results were not nearly optimal upon examination. The complete optimization could take days, which is not practical even for this simplified case study.

The problems encountered during design optimization when using GA and energy simulation as the platform are as follows: (1) due to the limitations of the software (Grasshopper + Galapagos + DIVA), only one specific time of a day (e.g., 9 a.m.) can be calculated per simulation. If design optimization involves a building simulation at a different time (e.g., 3 p.m.), the entire process must be re-performed. Thus, the information from both building simulations cannot be combined into one design optimization to find the optimal solution. (2) Design optimization is very time consuming and most architectural design projects have tight schedules. Therefore, it is unrealistic for architects to spend days on one optimization problem.

4.4.5 Offline Simulation

Because the current design optimization process has the above-mentioned problems, offline simulation was introduced to (1) integrate multiple building simulations at different times into one optimization problem (e.g., 9 a.m. and 3 p.m.); and (2) reduce computing time when a building simulation is coupled with design optimization by separating the simulation from the optimization process. The building simulation was separated from the GA optimization process and all required simulations were conducted in advance. This way, the simulation results could be reused in similar situations (e.g., all the rooms with windows facing the same direction will have the same daylight illuminance results in any GA generation and across the generations), and time consuming simulations would not need to be run for each solution.

Because the location of patient rooms is the variable, one question is how to define the window opening directions. In terms of daylight, a very large window facing north and a relatively small window facing south might both satisfy the LEED daylight requirement. However, window opening directions not only affect the interior daylight level, but also have a significant impact on building energy consumption. In LEED, a project can earn up to 24 points in the "energy performance" category. This suggests that a building can earn points depending upon the percentage of improvement in building energy consumption as compared to its baseline performance. This is calculated using a computer simulation model for the entire project based on Appendix G of the ANSI/ASHRAE/IESNA Standard 90.1-2007 (ASHRAE, 2007). The baseline model performance is defined as an average of the results of four simulations from four orientations: the original orientation of the building, and the orientation rotated by 90 °, 180 °, and 270 °. The results of the thermal energy simulation using DIVA show that the building consumes the least amount of heating and cooling energy when the window is facing south. The second best orientation is east, the third is west, and the worst north. Based on this analysis, the window opening priority order should be south - east - west - north. Using this pre-defined priority list saves time spent computing the GA; if the priority list is not used, all possible directions must be included in the search space and evaluated the by simulation and fitness functions. The floor plan was divided into four sections (see Figure 4.8). A VB Script node was written in Grasshopper to implement the following rules, the goal of which was to determine automatically each window's opening direction for the patient rooms during the GA process: (1) if the center of a room is in Section 1, the priority of window opening direction is south - east - west - north. VB will check the availability of adjacent rooms in the above sequence; (2) If the center of the room is in Section 2, the priority is west - south - east – north; (3) If the center of the room is in Section 3, the priority is north - south - east – west; (4) If the center of the room is in Section 4, the priority is east - south - west – north. The center of a room is possible to be on a line that separates two sections. The priority rules for window opening directions still hold in this situation: (1) if the center of the room is on the line that separates Sections 1 and 2, or 1 and 4, the room has a south window; (2) if the center is on the line that separates Sections 3 and 4, the room has an east window; and (3) if the center is on the line that separates Sections 2 and 3, the room has a west window. Any of the four directions of every room is possible for window opening, following the priority rules.

In more realistic simulations than this present, simplified study, further design modifications are required to make the design more practical in real projects: (1) if the patient rooms are away from the corridor (not connected), additional corridors need to be added into the design layout; (2) if the door of a room is blocked by other patient rooms, further modification to the design layout is needed. For example, in Figure 5, one room in the upper left corner needs access to the corridor.

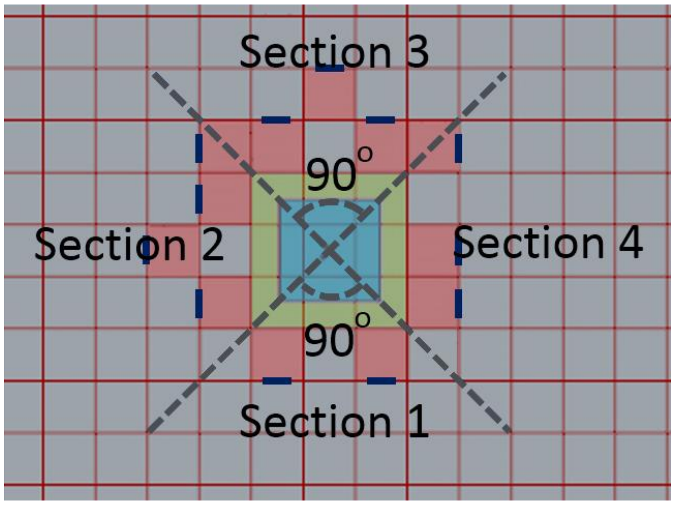


Figure 4.8 The Floor Plan is Divided into Four Sections in Order to Determine the Priority of Window Opening Directions. The Windows' Locations are Indicated in the Figure as Dark Line Segments on the Edges of the Rooms.

One of the research questions can be stated as follows: since the parameters are consistently changing during the optimization process, how to categorize the results of each simulation so that each category of simulation only needs a single simulation prior to the optimization process? For daylighting simulations, all patient rooms in this study share the same parameters (room dimensions, window size and location, building materials, number and location of sensors in DIVA), except for room locations and window opening directions. However, as long as the patient rooms have the same window directions, the daylighting values of the rooms can be regarded as also being the same (in a simplified experiment, when shading is ignored). Based on this, the daylight simulations of all patient rooms in this project can be simplified as simulating four patient rooms with windows facing south, east, west and north, respectively. Therefore, before the start of the GA optimization, a DIVA daylight simulation was conducted to calculate the daylight level in every patient room. The location of the rooms does not matter for daylight evaluation, only for walking distance calculation. The offline, pre-simulated daylight illuminance levels of all directions (south, east, west, and north) are used in the GA process. During the optimization process, although the room location and the window opening direction may change in any design solution, daylight fitness score of the entire design solution can be obtained by counting the total percentage of sensors that meet the LEED illuminance value.

Comparing the workflows of existing design optimizations (see Figure 4.9) and offline simulation-based optimizations (see Figure 4.10), the difference is that in offline simulation-based optimizations, building simulations are performed prior to the optimization process. This change can reduce a significant amount of computing time. In this simplified case study, it took the computer approximately 40 minutes to calculate 13 generations of GA when using the offline simulation method (plus about 8 minutes for the simulations of the four rooms with windows facing the four different directions), while the existing method uses 13 hours for 13 generations of GA. However, due to the large population size, although the obtained best nursing unit layout is close to the (perceived) optimal solution (an elliptical layout of patient rooms surrounding the nurses' station), the improvement in solutions got very slow after the 13th generation. Again, improvements to the GA process still need to be made.

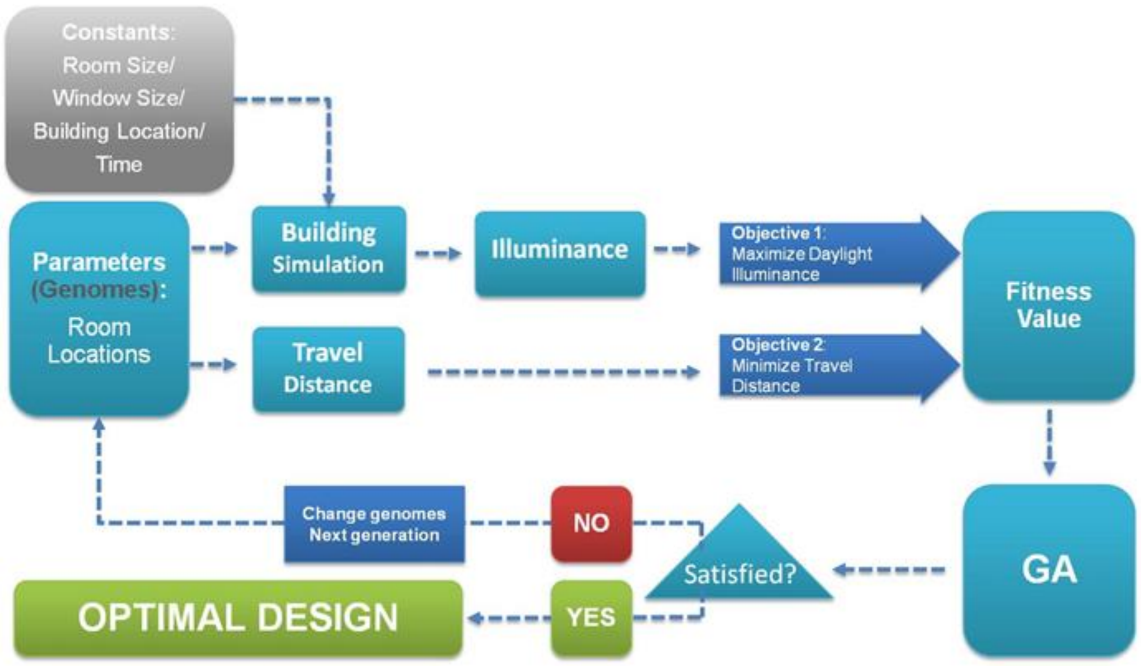


Figure 4.9 The Traditional GA Optimization Workflow.

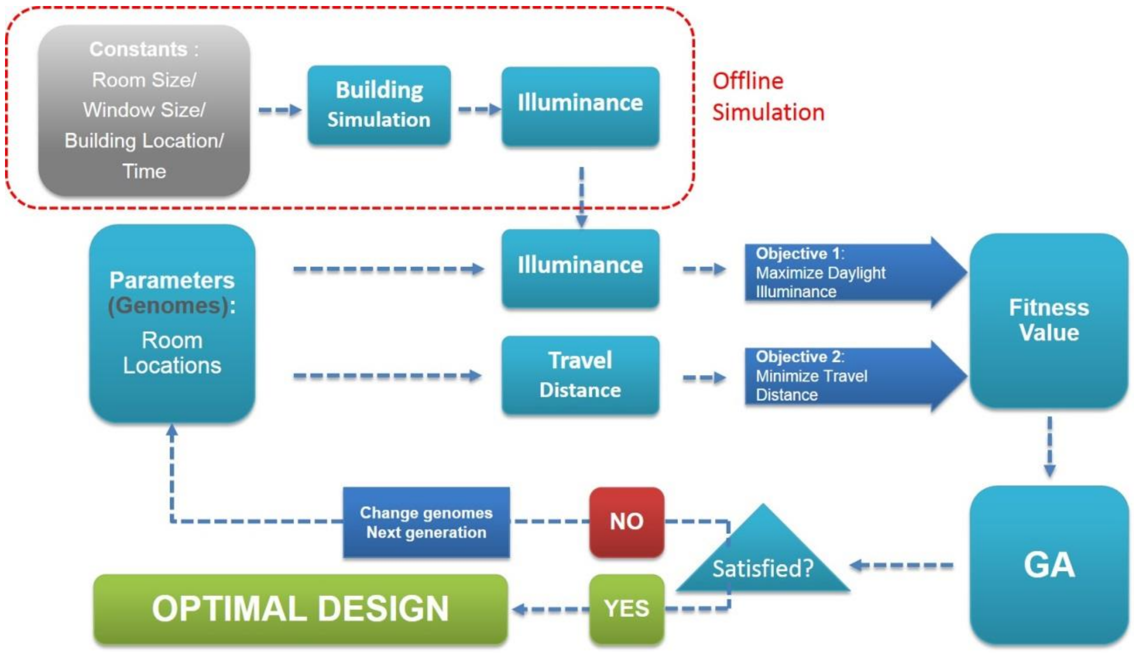


Figure 4.10 The Improved GA Optimization Workflow with Offline Simulation.

4.4.6 Divide & Conquer (D&C)

The method of Divide & Conquer is evaluated here in an effort to further improve GA efficiency and save computing time. This method suggests the use of GA to search in sub-problems with fewer populations instead of the entire population. As mentioned in Section 3.2, one requirement for the applicability of D&C is that the decision variables are independent of one another. This requirement can be satisfied in the present case study. In this study, when all 12 genomes (room locations) are used in GA, the total population size is: 225×224×…×214 = 1.249×1028, where 225 is the number of possible locations for the first genome, 224 is the number of possible locations for the second genome after the first genome has selected a spot, and so on. However, if only one genome is used in one GA run, the population size can be reduced to 225 for the first room, 224 for the second room, and so on. The idea here is that instead of directly solving an optimization problem with 12 genomes, another equivalent problem is solved: optimizing one genome at a time for 12 times. Figure 4.11 shows the working process of the nursing unit layout optimization when applying the D&C method. Each time one room is added to the previous result. This way, the nursing unit layout optimization is completed in 20 minutes.

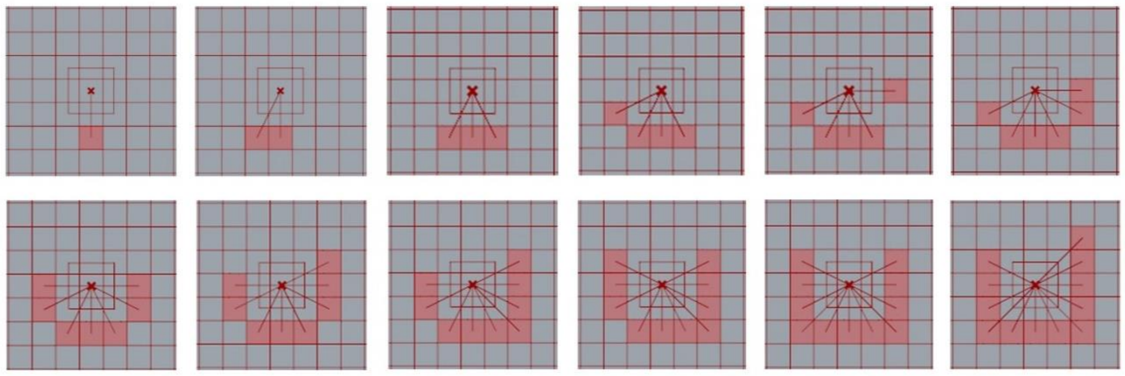


Figure 4.11 The Working Process of the Nursing Unit Layout Optimization with the D&C Method.

When the D&C method is applicable, the efficiency of GA can be further improved, and coding can be simplified by linking only one genome (the decision variable) to the fitness function. In this study, for GA with the D&C method, the fitness function for the travel distance is as follows:

$$Fitness_{\ travel\ distance} = 100(1 - (\frac{Distance - 30}{143 - 30})^2)$$

$$30\ ft \leq Distance \leq 143\ ft$$

The values 30 ft and 143 ft are the shortest and longest travel distances respectively for a single room. The fitness function for daylighting remains the same in this study. The two screenshots in Figure 4.12 show the difference in complexity of the Grasshopper files. The figures are drawn to scale.

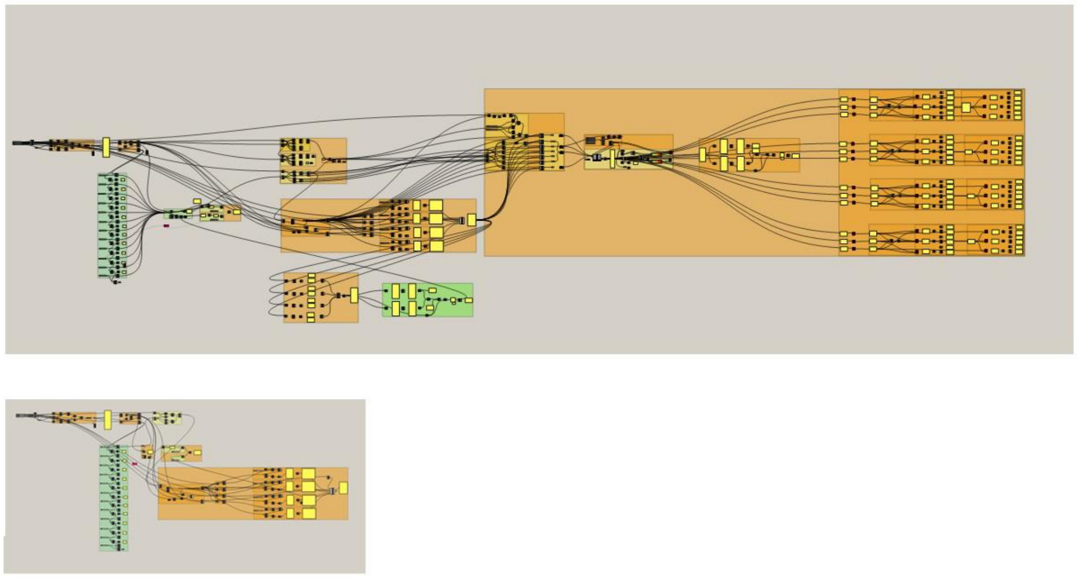


Figure 4.12 The Screenshots of Grasshopper Files of the Baseline Model Using Standard GA (Top) and the Improved Model Using GA with Offline Simulation and D&C (Bottom) in the Same Scale.

4.4.7 Results

In this study, first, a multi-objective optimization is converted into a single objective optimization by using a weighted sum of the fitness functions with arbitrary weights. Figure 4.13 shows two optimization results with different weights in their fitness functions. The image on the left is the result when using equal weights for travel distance and daylighting. The image on the right is the result when the weights between travel distance and daylighting are 1:2. While the optimal solutions are not surprising – they are consistent with our intuitive expectations regarding the results – the important point is the significant time savings when finding solutions with our new methods. Table 4.1 shows a comparison of the standard GA, GA with the offline simulation method, and GA with both the offline simulation and the D&C methods in terms of total time used, time used per generation, and whether the optimal solution is found. For this nursing unit optimization case study, using the existing method of GA optimization, the computer needed 78 hours to finish 13 generations of GA calculation (6 hours per generation), and the final result was not near optimal. By using offline simulation, the computing time was significantly reduced to 40 minutes to finish 13 generations of GA calculation (3 minutes per generation). However, 13 generations are not enough for finding the optimal layout, although the result is close to optimal. To further reduce computing time, experimentation was completed on the D&C method. This method provides a better chance of finding the peak of the fitness landscape by breaking a complicated problem down into sub-problems. When offline simulation and D&C were combined in this project, no more than 24 generations in 20 minutes (50 seconds per generation) were needed to find the optimal layout (each sub-problem's GA is stopped manually when no improvement is observed and therefore more generations are calculated than are necessary). In addition, the improved methods also help reduce the size and complexity of the model definition in the visual programming environment, Grasshopper.

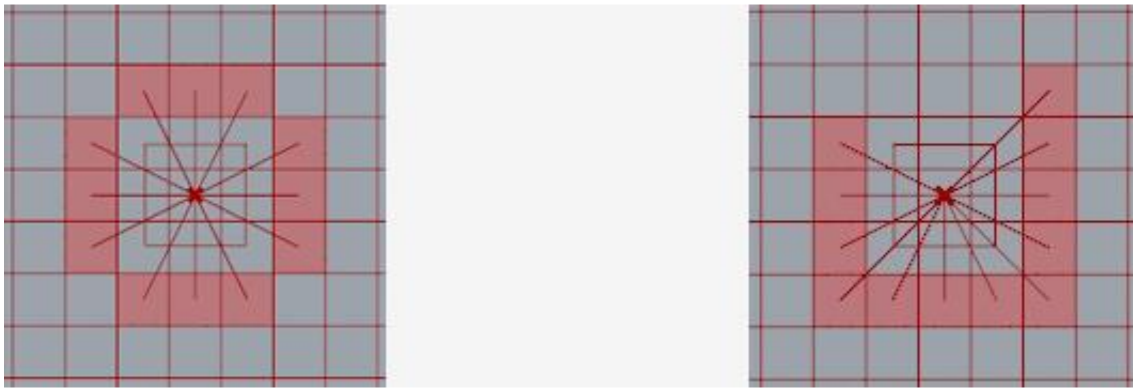


Figure 4.13 The Final Results of the Nursing Unit Layout Optimization with Different Weights in the Fitness Functions: (1) Fitness $_{\text{Travel Distance}}$: Fitness $_{\text{Daylighting}}$ = 1:1 (Left) (2) Fitness $_{\text{Travel Distance}}$: Fitness $_{\text{Daylighting}}$ = 1:2 (Right).

Table 4.1. Comparison Between Standard GA and Improved GAs.

Methods	Time	Time/Generation	Optimization Result
Standard GA	78 hours	6 hours	Not found
GA With Offline Simulation	40 minutes	3 minutes	Close
GA with Offline Simulation and D&C	20 minutes	50 seconds	Found!

Both techniques can also contribute great time saving in multi-objective optimization. However, current method does not support viewing many design options when the D&C method is used.

4.5 Conclusions and Discussions

In the previous case study, a multi-objective optimization is converted into a single objective optimization by using a weighted sum of the fitness functions with arbitrary weights. Both offline simulation and D&C methods are expanded to Pareto Pareto optimization for multiple design objectives. Octopus, a multi-objective optimization tool in Grasshopper is used as the optimization engine.

Offline simulation is first tested. It works well with the standard multi-objective optimization. Figure 4.13 shows the results in Octopus after around 9 hours of calculation. The optimization results are close to optimum. The benefits of using multi-objective optimization instead of single-objective optimization is that the designer can view multiple Pareto optimal solutions and compare the tradeoff between them. Figure 4.14 shows two Pareto optimal solutions in the Octopus results. The one on the left prioritizes more on the walking distance compared to the one on the right. However, Rhino crashed in the process when searching for the final Pareto optimal solutions when Octopus is in used the offline simulation. Further study is needed to investigate the problem.

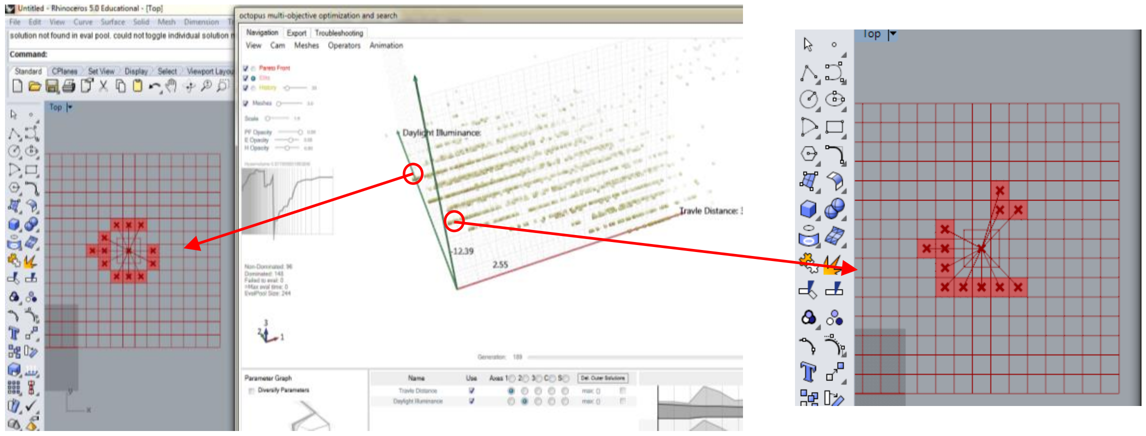


Figure 4.14 Results of Multi-Objective Optimization with Offline Simulation in Rhino/Grasshopper (Left and Right) and Octopus (Middle).

D&C method is then tested together with offline simulation in multi-objective optimization. It also works well here and Octopus finds the optimal design in about 20 minutes, about the same amount of time cost in single-objective optimization using Galapagos. However, D&C method only consider one genome in each optimization, so the Pareto optimal solutions showed in the Octopus (right image in Figure 4.15) are the ones when one genome is taken into account in each sub-problem. As a result, it will be more complex to visualize the Pareto optimal solutions for the entire optimization problem, thus losing the benefit of multi-objective optimization. Future study will further examine and resolve the problem.

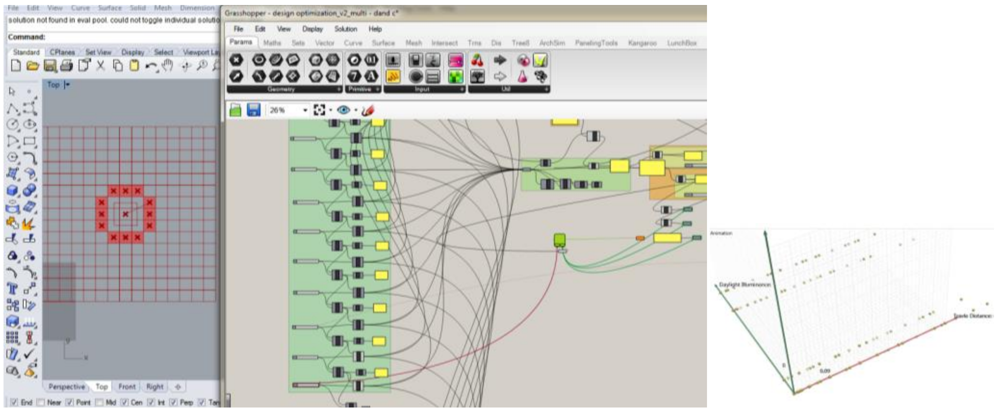


Figure 4.15 Results of Multi-Objective Optimization with Offline Simulation and D&C in Rhino/Grasshopper (Left) and Octopus (Right).

In this paper, two techniques were presented—offline simulation and Divide & Conquer—to achieve a more efficient GA optimization. The use of the two techniques (1) demonstrate significant time savings in the case study; and (2) provide a larger GA search space in the same amount of time, which offers a better chance of finding the optimal design. The use of GA in architectural design has become a trend in design optimization. Currently, however, only the general method of GA has been applied to architectural problems. A new type of study that utilizes architectural domain knowledge to customize GA techniques has been presented, and as a result the design optimization time has been significantly improved.

The tools used in this study, Rhino, Grasshopper, Galapagos, and DIVA are widely available existing tools that facilitate the development of the present methods. The methods presented in this paper, however, are not tied to these tools. Instead, the methods can be applied to similar tools, and even new implementations of such tools in more efficient programming languages.

The D&C method can be used to reduce computing time in optimization problems when the original problem can be broken down into several manageable sub-problems. In the case study, the D&C method required manual work to separately start the GA for each sub-problem in sequence, due to limitations in the software. It is expected that future GA software products will be capable of automatically starting the GA for each sub-problem in sequence. The offline simulation technique is beneficial when building simulations can be separated from the optimization process and conducted in advance. The simulation results can be reused in order to save computing time.

While the fitness function for daylighting performance is simplified, offline or pre-simulation is used with full daylighting simulation instead of surrogate models or approximate simulation models. The results of offline simulation are reused in the fitness functions of the optimization process. The simplified fitness functions can be substituted with more sophisticated fitness functions for different design objectives, but the method of reusing offline simulation in the process will remain the same for different design problems.

There are limitations in the use of the offline simulation and D&C methods. In offline simulation, the simulation result of each genome is pre-computed, so there is no mutual feedback among genomes. For example, self-shading (e.g. a room may cast shadow to other rooms) is ignored in our simplified case study, thus the effect of shading was not included in the daylight illumination result. If obstruction is considered in more complex spatial layouts, offline simulation may not be appropriate. One solution to this problem is to find and pre-compute all possible shading situations in advance. This may increase manual labor and computing time for offline simulation, but it can be assisted by scripting to automate the process. In a complex problem, designers may need to optimize the total GA computing and offline simulation time. The D&C method has a similar limitation. In D&C, a genome that is newly added into GA has no impact on the genomes that have already been placed, but in actual problems the genomes may be dependent (again, e.g. a room may cast shadow to other rooms affecting their illumination). When genomes are not independent of one another, designers may group the sub-problems into sets of sub-problems to enable feedback among genomes within each set of sub-problems. Future study is needed to further examine and resolve the limitations of both offline simulation and D&C.

To discover these improvement techniques, architectural domain knowledge was needed. For example, if two identical rooms (same shape, same windows, and same shading) in a building are facing the same direction, they have the same illuminance at any given time because the sun is far enough away that the difference in light angles between the two rooms is negligible. Another example is that in many cases in the northern hemisphere, the best window direction for optimal thermal performance is south, followed by east, west and north. This knowledge was confirmed by our thermal simulation and used in the offline simulation. To sum up, designers can play an important role in improving optimization efficiency. An architect’s design knowledge should be utilized to customize the optimization process; a process that would significantly save in computing time and eventually make optimization practical for architectural design.

In future work, the application of the methods to more complex case studies with more detailed building models and more sophisticated design objectives should be investigated. These techniques will be expanded to Pareto Optimization for multiple design objectives and investigate the utilization of more specific design knowledge, e.g. evidence-based design knowledge found in research of healthcare facility design, and correlations or causal relationships among decision variables, among design solutions, and between decision variables and design solutions that can be acquired from building science, Post-Occupancy Evaluation, etc. When more specific or complex design knowledge is embedded into GA and its improvement methods with more case studies, more interesting findings about the advantages and limitations of techniques such as offline simulation and D&C could be made. Guidelines can be developed about when the techniques can be utilized and what kind of design optimization problems can be applied.

CHAPTER V

DESIGN KNOWLEDGE DISCOVERY SYSTEM

5.1 Introduction

The second part of this study applies existing data mining techniques to automatically generate design rules and knowledge using learned correlations between the decision variables and the optimal design performances. This valuable knowledge can be obtained from the design optimization results. It can, in turn, improve the efficiency of optimization and even serve as a guideline for future designs, reducing the need for simulation and optimization during the design process. The research objectives of this part of the study are to provide a method and prototypical tools to: (1) discover useful correlations and causal relationships among the decision variables, among the design solutions, and between the decision variables and design solutions in an optimization problem; (2) discover knowledge that can form design guidelines for specific design problems with similar design variables, constraints, and design objectives; and (3) discover knowledge that can form general design guidelines in the future and reduce the need for costly computing for optimization.

5.2 Problems with Previous Knowledge Discovery

Previous work in this area has mostly been for demonstration purposes; the rules derived from the simple prototypes produced have not been new to designers (e.g., one should use pre-stressed concrete for minimum slab thickness, etc.) (Mackenzie & Gero, 1987). Moreover, existing knowledge discovery methods are based on manual and visual analyses of the results. Knowledge about the design and performance relationships can be obtained from Pareto optimization and a manual analysis of the results (Radford & Gero, 1987). As the use of design optimization grows, so does the corresponding need to generate useful knowledge from the results of optimization quickly, and even automatically.

5.3 Knowledge Discovery System

In this study, the term "knowledge" refers to the useful information in optimization results that is of interest to designers and that can be used in future design practice and design optimization. This learned knowledge is intertwined with the relationships among a project's various objectives and decision variables. For example, one piece of knowledge could be: "walking distance and daylighting in patient rooms are two conflicting attributes in patient unit design; increasing the performance of one attribute will decrease the performance of the other." Another piece of knowledge might be: "if walking distance is the main objective in designing a patient unit with less than 24 patient rooms, the best design layout is a circle." The knowledge obtained is likely to be much more complex if more design conditions and payoffs are involved. The main objective of this study is to develop a process to help designers discover useful and practical knowledge for use in enhancing their design.

The knowledge discovery process in this study is divided into three main steps. The first step is identifying project requirements and key design objectives. Every design project is unique, with unique site restrictions, client requirements/demands, design objectives, and design parameters. Site restrictions can be set as constraints later on in the design optimization process. The design leader should discuss the design requirements/demands with his or her clients and identify key design objectives early in the design process. Once the major design objectives are settled, the design leader should identify any design parameters that might have an impact on these objectives.

The second step is conducting building simulations and design optimizations; two improvement methods were discussed in Chapter 4. Both building simulation and design optimization are very time consuming. Two methods - offline simulation, and D&C - were introduced and demonstrated. The goal was to reduce simulation runs and expedite the optimization process. By using the improved methods, more optimization, data, and knowledge can be obtained in the same amount of time. This step generates a large amount of data, especially when multiple design objectives (with possibly conflicting attributes) are involved in the optimization.

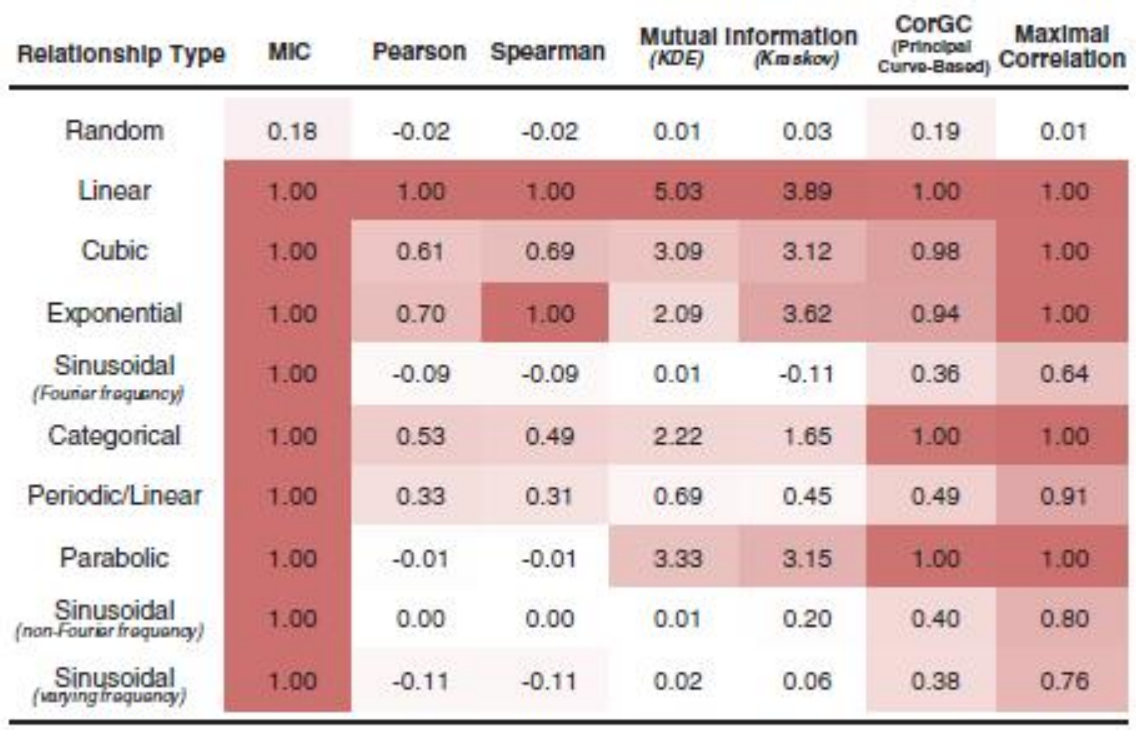
The third step is using data mining and machine learning techniques to analyze the results. An automatic system was developed to extract knowledge from the optimization results; the goal was to help with improving efficiency and providing guidelines for future design decision making and planning.

5.3.1 Maximal Information Coefficient (MIC) and Validations

The second part of this dissertation study presents an optimization-based method for knowledge discovery. The parametric form finding nursing unit case study used in the first part of this study was used again here to test the method for identifying the shape of the final optimal design.

A data mining tool called Maximal Information Coefficient (MIC) (Reshef et al., 2011) is used in this new optimization-enabled knowledge discovery system. With this tool, the computer is able to identify the simple shapes in the final design form (such as a circle, square, line, etc.) as the layouts of rooms, and correlate these layouts with the corresponding optimal performances. This knowledge can be used as a guideline for future design projects with similar design variables and objectives. Compared to the existing knowledge discovery methods that are based on manual and visual inspection of the results, the proposed knowledge discovery system has the potential to be automated. Therefore, more knowledge could be discovered in the same amount of time.

MIC was developed by Reshef and associates (2001) to measure and identify novel relationships and the strength of the correlations among the variables in large and complex datasets with thousands of variable pairs. Figure 5.1 shows the comparison between MIC and other statistic techniques such as the Pearson correlation coefficient, Spearman, mutual information, maximal correlation, and the principal curve–based dependent measure (Reshef et al., 2011). Pearson correlation coefficient measures the linear dependence between two variables (Pearson, 1895). Spearman rank correlation is a nonparametric statistical measure of monotonic relationship between two variables (Spearman, 1905). Mutual information investigates the dependence between two variables in experimental time series (Moon, Rajagopalan & Lall, 1995). CorGC is the principal curve-based measure of dependence (Reshef et al., 2011). Maximal correlation is another method to measure the dependence of two variables (Sarmanov, 1962). The highlighted areas in Figure 5.1 show the scores given to different noiseless relationships by the above mentioned statistic techniques. Higher value numbers (shown in dark red) indicate stronger relationships can be detected by the corresponding statistic methods. A low value number (in light red or white) means this method does not work well to identify a certain relationship. Different methods have different upper score limits. For example, the highest number in MIC is 1. This figure shows that MIC can identify and provide better results to various noise or noiseless functional relationships than other statistic techniques (Reshef et al., 2011). MIC has been verified for its generality (MIC will capture relationships with sufficiently sized bodies of data) and equitability (different types of data with the same noisy relationships should receive similar results in MIC) (Reshef et al., 2011). MIC belongs to a larger family - the maximal information-based nonparametric exploration statistics - which can find valuable connections in data. The MINE software program is an implementation of MIC and it is developed by David Reshef and Yakir Reshef. (http://www.exploredata.net/Downloads).

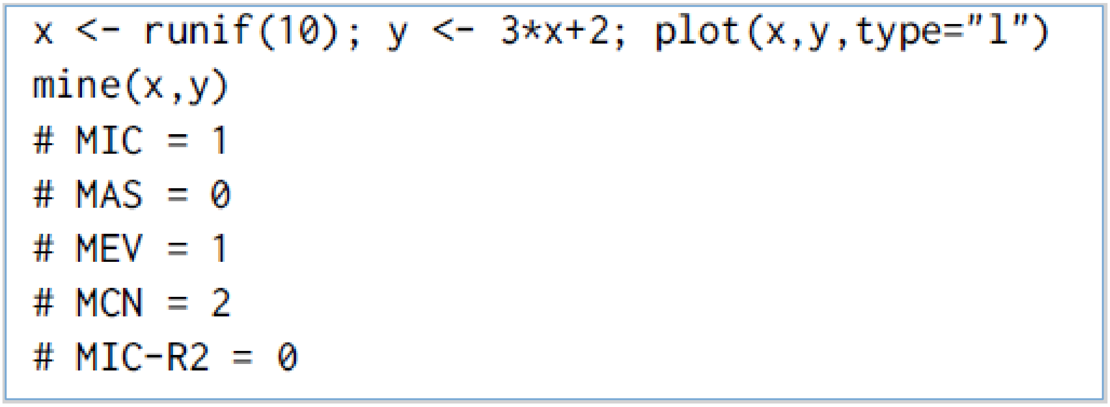


Relationship Type	MIC	Pearson	Spearman	Mutual Information (KDE)	Mutual Information (Kraskov)	CorGC (Principal Curve-Based)	Maximal Correlation
Random	0.18	-0.02	-0.02	0.01	0.03	0.19	0.01
Linear	1.00	1.00	1.00	5.03	3.89	1.00	1.00
Cubic	1.00	0.61	0.69	3.09	3.12	0.98	1.00
Exponential	1.00	0.70	1.00	2.09	3.62	0.94	1.00
Sinusoidal *(Fourier frequency)*	1.00	-0.09	-0.09	0.01	-0.11	0.36	0.64
Categorical	1.00	0.53	0.49	2.22	1.65	1.00	1.00
Periodic/Linear	1.00	0.33	0.31	0.69	0.45	0.49	0.91
Parabolic	1.00	-0.01	-0.01	3.33	3.15	1.00	1.00
Sinusoidal *(non-Fourier frequency)*	1.00	0.00	0.00	0.01	0.20	0.40	0.80
Sinusoidal *(varying frequency)*	1.00	-0.11	-0.11	0.02	0.06	0.38	0.76

Figure 5.1 Comparison Between MIC and Other Methods (Image Source: Reshef et al., 2011).

The validation of the design knowledge discovery process using the MIC technique is presented in a series of tests with simple forms or relationships, such as a linear relationship, parabolic relationship, circle, square, and U-shape. The purpose of conducting these tests on different shapes is to verify whether MIC can yield distinct feature values for different forms or relationships. The test results of the various forms are used in our case study later in the chapter.

There are five metrics for the MIC results: MIC, MAS, MEV, MCN, and MIC-R2 (see Figure 5.2).



```
x <- runif(10); y <- 3*x+2; plot(x,y,type="l")
mine(x,y)
# MIC = 1
# MAS = 0
# MEV = 1
# MCN = 2
# MIC-R2 = 0
```

Figure 5.2 The MIC Results of a Linear Relationship.

(1) **MIC** (Maximal Information Coefficient) (a measure of strength): MIC measures the linear correlation between two variables. MIC assigns 0 to uncorrelated variables and 1 to correlated noiseless variables. To accomplish this measurement, MIC overlays a grid on the data sets of two variables and increases the resolution of the grid until it reaches the maximum resolution in order to effectively detect the correlation among the data (Reshef et al., 2011). With a large sample size:

> *"(i) MIC assigns a perfect score of 1 to all never-constant noiseless functional relationships, (ii) MIC assigns scores that tend to 1 for a larger class of noiseless relationships, and (iii) MIC assigns a score of 0 to statistically independent variables."* (Reshef et al., 2011, pp. 1520).

(2) **MAS** (Maximum Asymmetry Score) (a measure of non-monotonicity): MIC has three key indices that can be used to discover non-linear relationship in the data: MAS, MEV, and MCN (Caban et al., 2012). MAS can detect deviations from monotonicity (Reshef et al., 2011).

"*MAS is useful, for example, for detecting periodic relationships with unknown frequencies that vary over time, a common occurrence in real data.*" (Reshef et al., 2011, pp. 1522).

(3) **MEV** (Maximum Edge Value) (a measure of functionality). This value measures the closeness to being a function and calculates the degree to which the variables are from a continuous function. MEV ranges from 0 to 1. A high MEV value indicates well-behaved functions (Caban et al., 2012).

(4) **MCN** (Minimum Cell Number) (a measure of complexity). MCN counts the number of cells needed to get a MIC value. A well-behaved and monotone function requires a small number of cells. A poorly defined, non-monotone function requires a large number of cells to reach MIC (Caban et al., 2012).

(5) **MIC − ρ2** (nonlinearity, represented by MIC-R2 in MINE - a software implementation of MIC): The statistic MIC − ρ2 measures linear dependence. It assigns values near 0 for variables with linear relationships and high values for variables with non-linear relationships, together with high values of MIC. It is a useful index for discovering novel, non-linear relationships (Reshef et al., 2011).

Below are some of the relationships and shapes tested in this study. The tools used include: Rhino/Grasshopper (as the drawing platform), the R programming language, and the MINE package hosted in R (http://www.exploredata.net/Downloads).

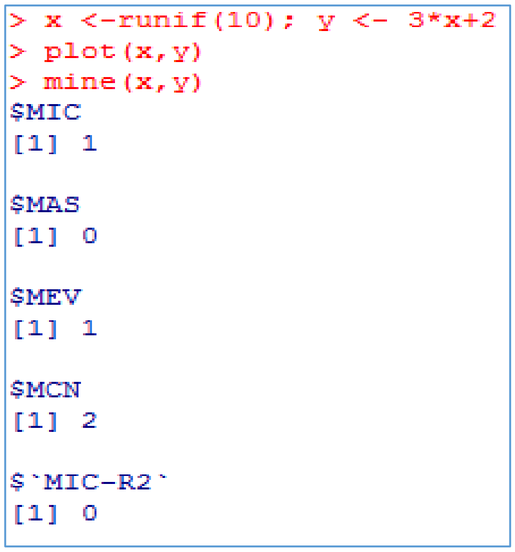
(1) **Linear relationship**

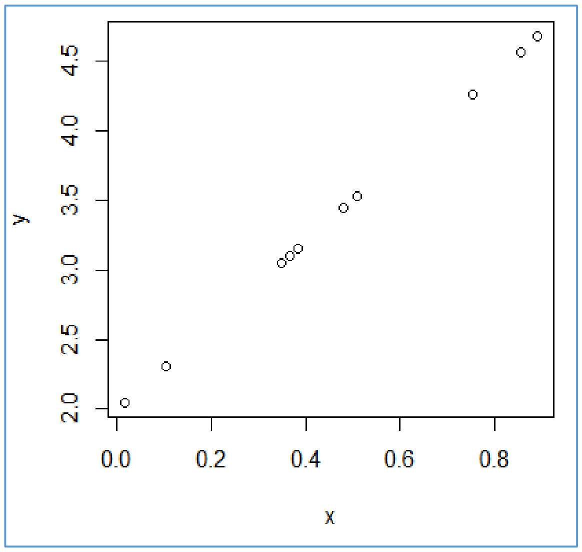
To validate this method, I used the MIC technique with the MIME program twice for each test. First, I drew the shape as the ground truth directly in R and used MINE to calculate the metrics. Second, I drew the shape in Rhino/Grasshopper, extracted the points on the shape, and imported them into MINE. MINE calculated the metrics values and detected the shape of the data. For example, in this linear relationship test (see Figure 5.3, bottom left), I drew 10 points in Rhino/Grasshopper. The X values of these 10 points were random points from 0 to 1 (the randomness was defined by Grasshopper). The Y value equals to 3x+2. The definitions of the points are as follows:

X = 10 random numbers from 0 to 1

Y = 3x+2

The top and the bottom left images in Figure 5.3 show the results and linear relationship when the shape (known as the ground truth) was drawn directly in MINE using linear equations above. The image on the bottom right shows the results in MINE when the shape was drawn in Rhino/Grasshopper. The results of both tests are identical.


```
> x <-runif(10); y <- 3*x+2
> plot(x,y)
> mine(x,y)
$MIC
[1] 1

$MAS
[1] 0

$MEV
[1] 1

$MCN
[1] 2

$`MIC-R2`
[1] 0
```



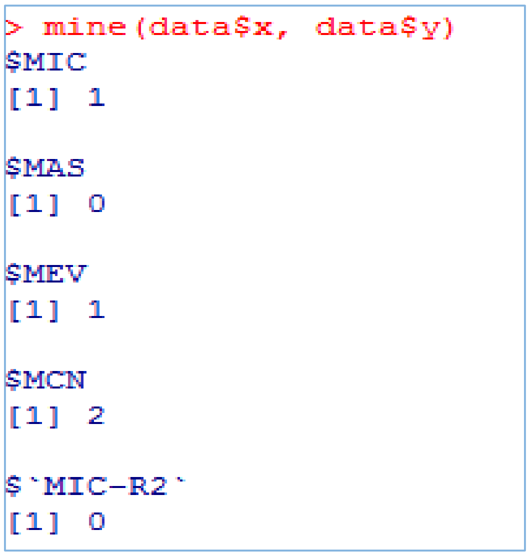

```
> mine(data$x, data$y)
$MIC
[1] 1

$MAS
[1] 0

$MEV
[1] 1

$MCN
[1] 2

$`MIC-R2`
[1] 0
```

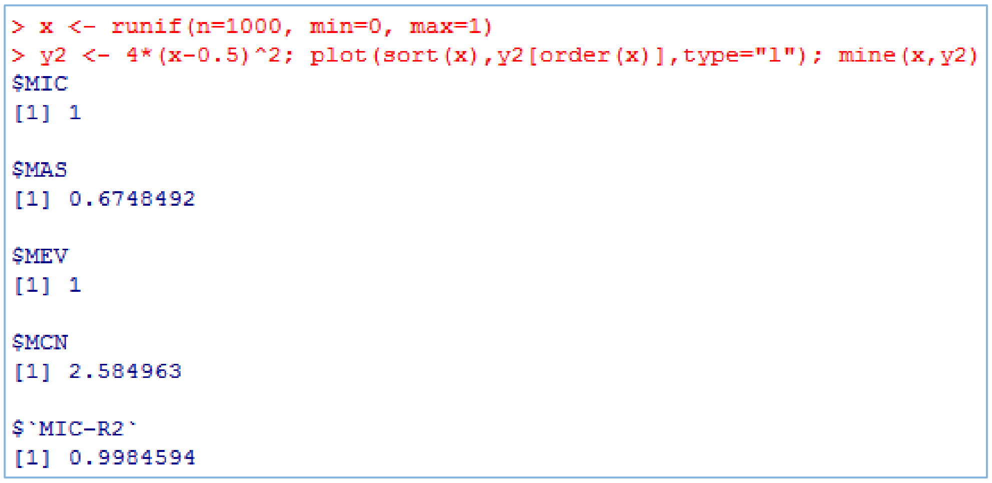
Figure 5.3 Top: The Results in MINE when the Shape was Drawn Directly in MINE (the Ground Truth). Bottom Left: The Linear Relationship Shown in MINE (the Ground Truth). Bottom Right: The Results in MINE When the Test Shape was Drawn in Rhino/Grasshopper.

On the one hand, based on the test results of a linear relationship, we can see that the metrics in MINE show: MIC=1, MAS=0, MEV=1, MCN=2, and MIC− ρ2 ≈ 0. On the other hand, when we have these metrics shown in MINE, we see that the variables have a linear relationship. Additional similar tests were performed, as shown below.

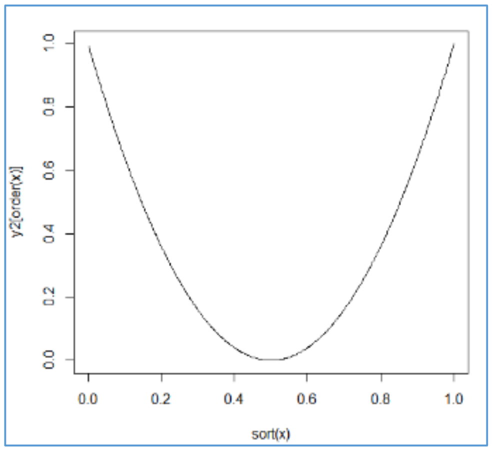
(2) Parabolic relationship (see Figure 5.4)

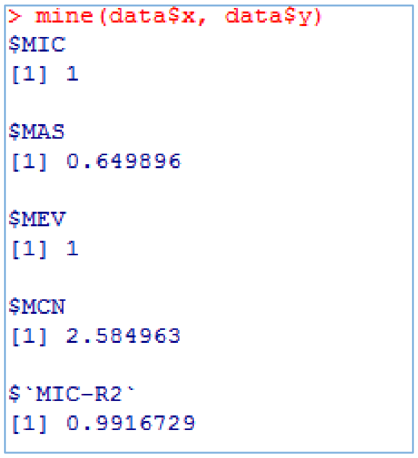
X = 1000 random numbers from 0 to 1

Y2 = 4*(x-0.5)^2



```
> x <- runif(n=1000, min=0, max=1)
> y2 <- 4*(x-0.5)^2; plot(sort(x),y2[order(x)],type="l"); mine(x,y2)
$MIC
[1] 1

$MAS
[1] 0.6748492

$MEV
[1] 1

$MCN
[1] 2.584963

$`MIC-R2`
[1] 0.9984594
```





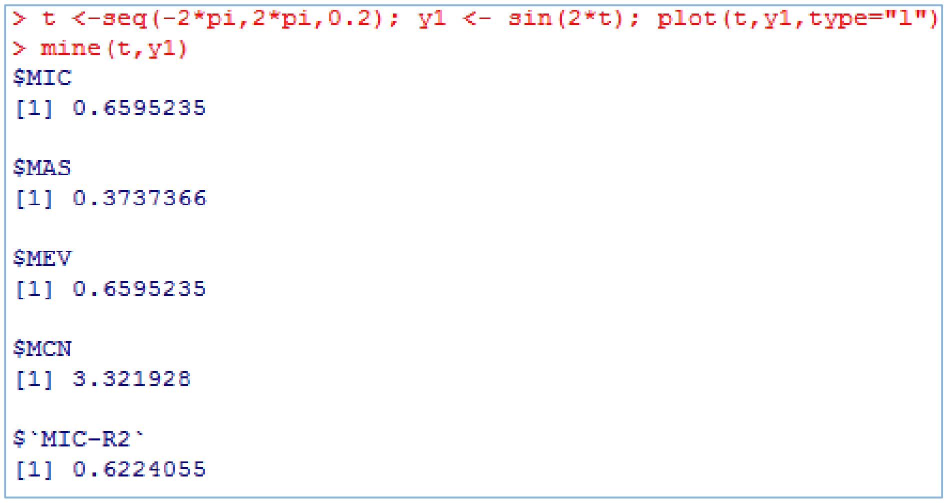
```
> mine(data$x, data$y)
$MIC
[1] 1

$MAS
[1] 0.649896

$MEV
[1] 1

$MCN
[1] 2.584963

$`MIC-R2`
[1] 0.9916729
```

Figure 5.4 Top: The Results in MINE When the Shape was Drawn Directly in MINE (the Ground Truth). Bottom Left: The Parabolic Relationship Shown in MINE (the Ground Truth). Bottom Right: The Results in MINE When the Test Shape was Drawn in Rhino/Grasshopper .

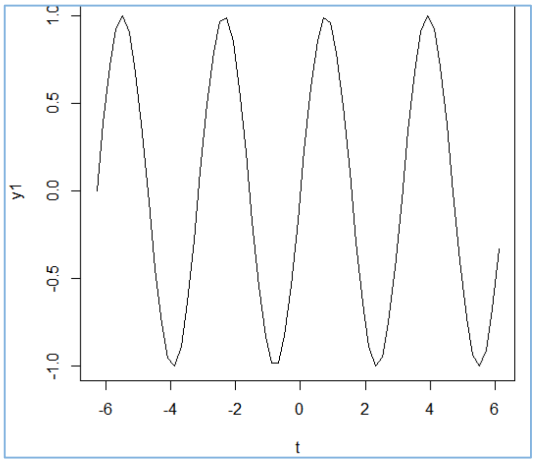
(3) Sinusoidal function (see Figure 5.5)

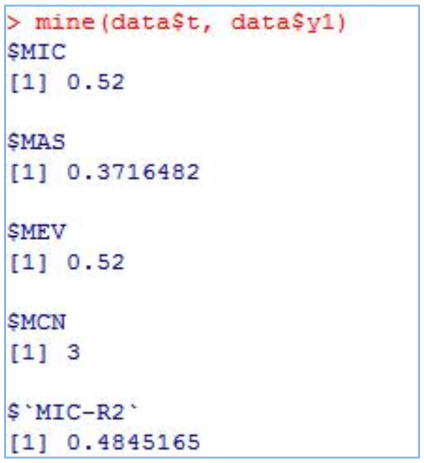
t = points in sequence with interval value 0.2 (as the ground truth), or 0.25 (test case in Rhino/Grasshopper), from -2*pi to 2*pi

y1 = sin (2*t)



```
> t <-seq(-2*pi,2*pi,0.2); y1 <- sin(2*t); plot(t,y1,type="l")
> mine(t,y1)
$MIC
[1] 0.6595235

$MAS
[1] 0.3737366

$MEV
[1] 0.6595235

$MCN
[1] 3.321928

$`MIC-R2`
[1] 0.6224055
```





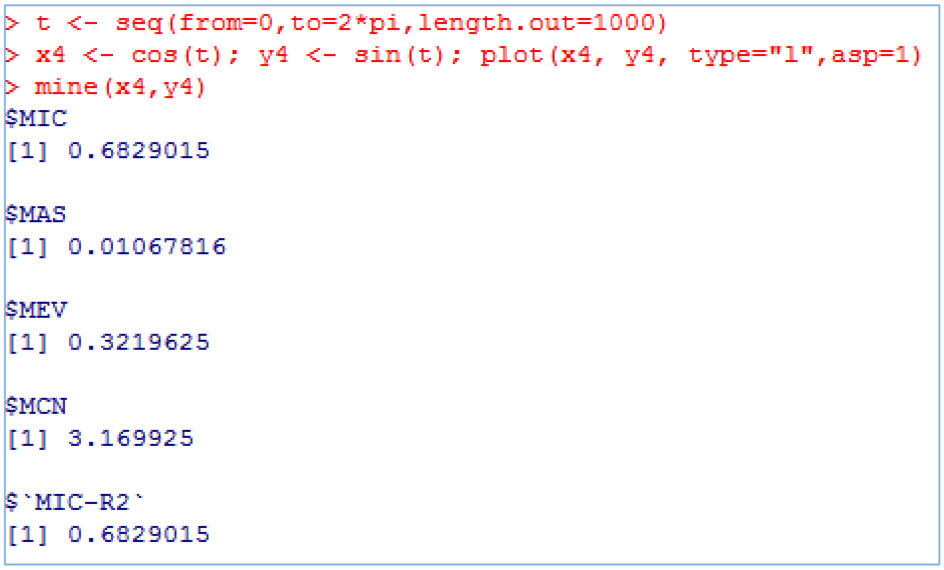
```
> mine(data$t, data$y1)
$MIC
[1] 0.52

$MAS
[1] 0.3716482

$MEV
[1] 0.52

$MCN
[1] 3

$`MIC-R2`
[1] 0.4845165
```

Figure 5.5 Top : The Results in MINE When the Shape was Drawn Directly in MINE (the Ground Truth). Bottom Left: The Sinusoidal Relationship Shown in MINE (the Ground Truth). Bottom Right: The Results in MINE When the Test Shape was Drawn in Rhino/Grasshopper.

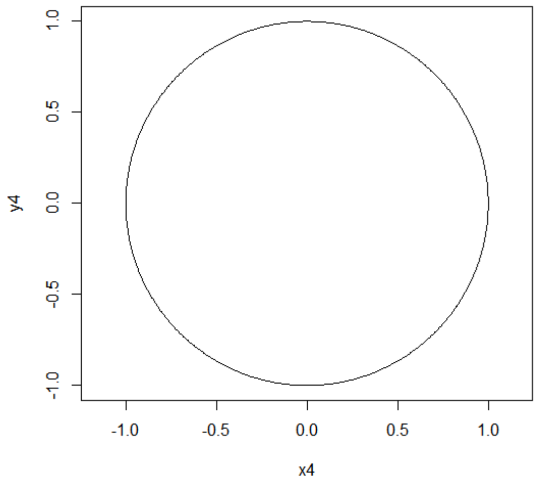
(4) Circle shape (see Figure 5.6)

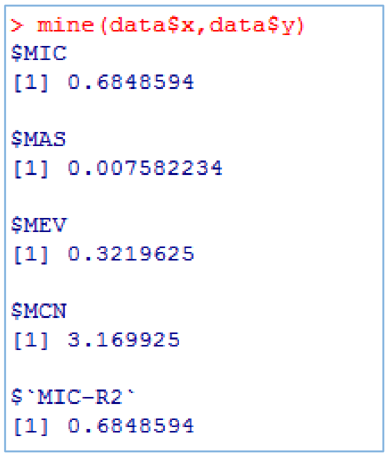
X and Y are the coordinates of points on a circle.

Radius is 1 in the ground true, and 19.5 in Rhino/Grasshopper.



```
> t <- seq(from=0,to=2*pi,length.out=1000)
> x4 <- cos(t); y4 <- sin(t); plot(x4, y4, type="l",asp=1)
> mine(x4,y4)
$MIC
[1] 0.6829015

$MAS
[1] 0.01067816

$MEV
[1] 0.3219625

$MCN
[1] 3.169925

$`MIC-R2`
[1] 0.6829015
```





```
> mine(data$x,data$y)
$MIC
[1] 0.6848594

$MAS
[1] 0.007582234

$MEV
[1] 0.3219625

$MCN
[1] 3.169925

$`MIC-R2`
[1] 0.6848594
```

Figure 5.6 Top: The Results in MINE When the Shape was Drawn Directly in MINE (the Ground Truth). Bottom Left: The Circle Shape Shown in MINE (the Ground Truth). Bottom Right: The Results in MINE When the Test Shape was Drawn in Rhino/Grasshopper.

.

The examples shown above indicate that different shapes/relationships have different metrics values in MINE. Additional tests confirm that the values in MINE are distinctly different for different shapes/relationships, and the values are consistent for the same shapes. When the design optimization (the hospital layout test) was completed for the case study, I was able to categorize the shapes in the optimized design according to the shapes produced using the metrics obtained in MINE, and identify the relationship between the design's performance and the found room layout shape in the optimized design.

5.3.2 Benchmarks

Two important indexes influence the MINE results. One is the sample size, and the other one is the alpha value (α). The value α affects the resolution of the search grid, and thus the computing time (the larger α is, the higher the resolution and the greater the computing time required) (Filosi et al., 2014). The authors of MINERVA have pointed out that α refers to the exponent value in its original Java programming code. The value range for α is from 0 to 1. The default value 0.6 for α (the exponent of the search grid size $B(n) = n^a$) was chosen based on experiences to achieve a plausible approximation without requiring that the process be extremely time consuming (Filosi et al., 2014).

Users should increase α up to 1 if the sample size is small so that the results will be closer to the ground truth. In this project, due to the fact that the case study had a small sample size of 12, α=1 was used to calculate the theoretical values for the benchmark, as well as for the test data (the x and y coordinates of the rooms). Table 5.1 shows the benchmarks of five distinct shapes or relationships with their corresponding values in MINE. These results, obtained from the case study, were compared with the benchmark values to determine if the optimal design (the room layout) was close to any of the shapes that had been tested previously.

Table 5.1 The Benchmarks of Five Different Shapes and Corresponding MINE Metrics Values.

Relationship Type	**MIC**	**MAS**	**MEV**	**MCN**	**MIC-R2**	**Sample Size and α Value**
Linear	1	0	1	2	0	12 points total, α=1
Parabolic	1	0.68	1	2.58	1	
Circle	0.65	0	0.42	3.16	0.65	
Square	0.15	0	0.04	2.58	0.16	
U-shape	0.32	0.27	0.32	2.58	0.32	

5.3.3 Mean Square Error (MSE)

The statistical concept Mean Square Error (MSE) (Lehmann & Casella, 1998) was used to measure the “errors,” which are the differences between the measured MINE results and the previously discussed benchmarks. In other words, the MINE results of each optimized design layout were compared with the benchmark MINE values in each relationship type that had been calculated beforehand (Table 5.1). The goal was to find the expected relationship type in the benchmark for which the optimized design layout had the smallest MSE value (also in the benchmark).

If $\hat{Y}$ is the predicted value, and Y is the true value, the MSE of the predictor is calculated as:

$$\mathrm{MSE} = \frac{1}{n}\sum_{n=1}^{n}(\hat{Y} - Y)^2$$

In Grasshopper, several equations are written in VB code to calculate an MSE value. For example, the equation for calculating the MSE value between predicted MINE values and the linear relationship in the benchmark is:

$$\mathrm{MSE} = \frac{1}{5}\left((\widehat{\mathrm{MIC}} - \mathrm{MIC}_{\mathrm{linear}})^2 + (\widehat{\mathrm{MAS}} - \mathrm{MAS}_{\mathrm{linear}})^2 + (\widehat{\mathrm{MEV}} - \mathrm{MEV}_{\mathrm{linear}})^2 + (\widehat{\mathrm{MCN}} - \mathrm{MCN}_{\mathrm{linear}})^2 + (\widehat{\mathrm{MICR}}2 - \mathrm{MICR2}_{\mathrm{linear}})^2 \right)$$

5.3.4 Implementation

Two major sets of tools are used in this knowledge discovery system, including: (1) Rhino/Grasshopper, with the Galapagos plugin (a single objective optimization tool) and the gHowl plugin (for exchanging information with other applications such as Excel); (2) the MINE application package (for computing MIC values as well as other statistics (http://www.exploredata.net/Downloads/MINE-Application)). Excel is used to transfer information between Grasshopper and the R program.

The diagram in Figure 5.7 lists the programs used, as well as the workflow of this knowledge discovery system. First, after the design optimization is completed in Grasshopper, the chromosomes (in this study, the locations of patient rooms) from the optimized design are exported into an Excel spreadsheet. Then R reads the spreadsheet and calculates the MINE results, which are then saved in another Excel spreadsheet. Next, Grasshopper analyzes the data in the spreadsheet via the MSE method and VB scripting, and puts out the following message in a dialog: [given certain conditions] “the optimal design is a circle” (meaning that the optimal layout for the rooms is a circle shape).

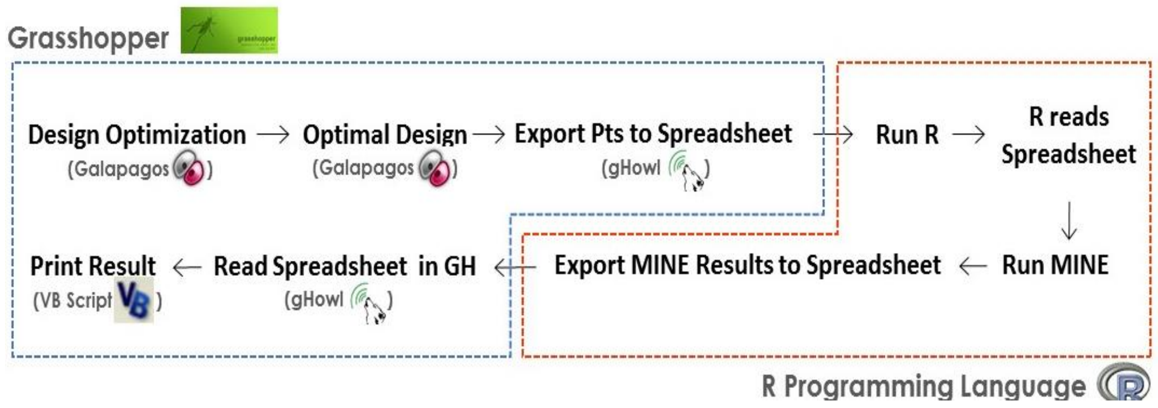
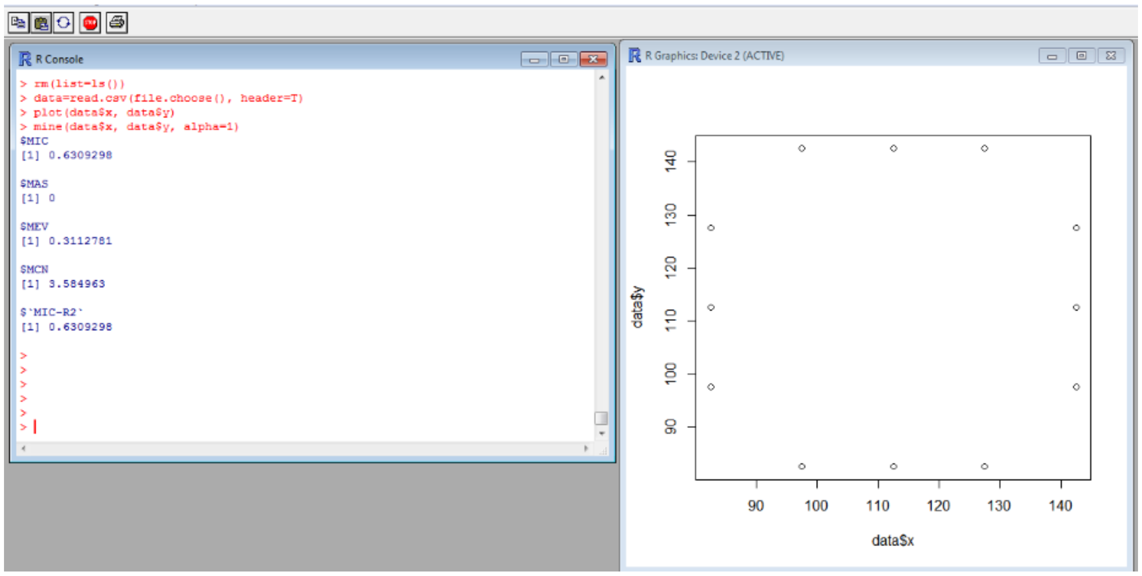


Figure 5.7 The Workflow of the Knowledge Discovery System.

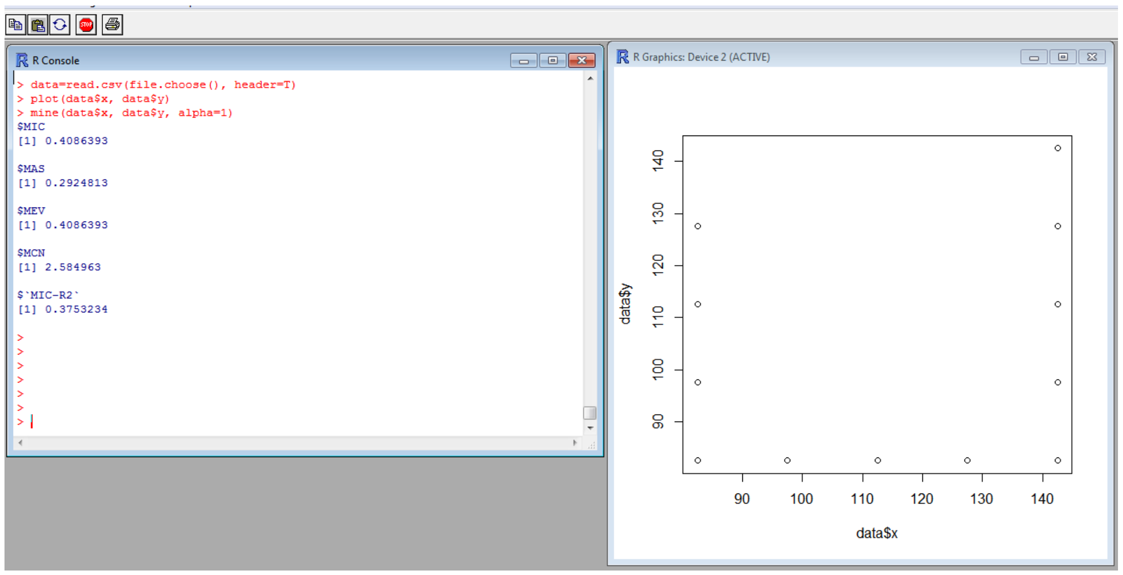
All of the complex calculation processes in the workflow are automatic. There are a few manual starts for different programs (e.g., starting the R program after the optimization process is completed in Grasshopper), but they can also be automated with improved program interoperability.

5.3.5 Results

I applied the methods to the nursing unit layout optimization examples with two different sets of weights for the fitness functions. Figure 5.8 shows the MINE results after the coordinates of the 12 rooms in the optimal solution were imported into MINE.



```
> rm(list=ls())
> data=read.csv(file.choose(), header=T)
> plot(data$x, data$y)
> mine(data$x, data$y, alpha=1)
$MIC
[1] 0.6309298

$MAS
[1] 0

$MEV
[1] 0.3112781

$MCN
[1] 3.584963

$`MIC-R2`
[1] 0.6309298

>
>
>
>
>
>
```

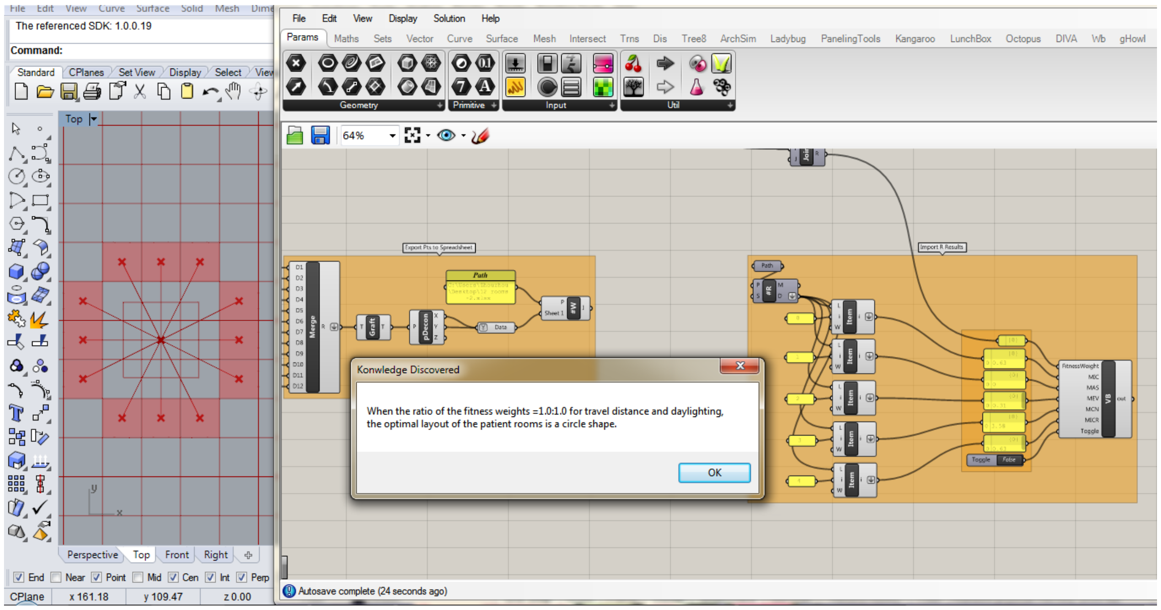


```
> data=read.csv(file.choose(), header=T)
> plot(data$x, data$y)
> mine(data$x, data$y, alpha=1)
$MIC
[1] 0.4086393

$MAS
[1] 0.2924813

$MEV
[1] 0.4086393

$MCN
[1] 2.584963

$`MIC-R2`
[1] 0.3753234

>
>
>
>
>
>
>
```

Figure 5.8 The MINE Results of the Nursing Unit Layout Optimization With Different Weights for the Fitness Functions: (1) Fitness $_{\text{Travel Distance}}$: Fitness $_{\text{Daylighting}}$ = 1:1 (Top); (2) Fitness $_{\text{Travel Distance}}$: Fitness $_{\text{Daylighting}}$ = 1:2 (Bottom). The Small Circles in the Graphs Represent the Points/Locations of the Rooms Obtained From the Optimization Results**.**

Figure 5.9 shows the message boxes as outputs from Grasshopper. The results indicate that when the optimization design problem has equal weights for travel distance and daylighting, the system finds the optimized layout shape to be best interpreted as a circle (for which the MSE is the smallest among all the shapes tested. The second best was a square shape.). The system automatically pops up the following message as discovered knowledge: “*When the ratio of the fitness weights = 1.0:1.0, for travel distance and daylighting, the optimal layout of the patient rooms is a circle shape*.”

When the ratio of the weights of travel distance to daylighting is 1:2, the final optimized design/room layout shape is best interpreted as a U-shape (for which the MSE is the smallest among all the shapes tested). The automatic output message is: “*When the ratio of the fitness weights = 1.0:2.0, for travel distance and daylighting, the optimal layout of the patient rooms is a U shape.*”


The referenced SDK: 1.0.0.19

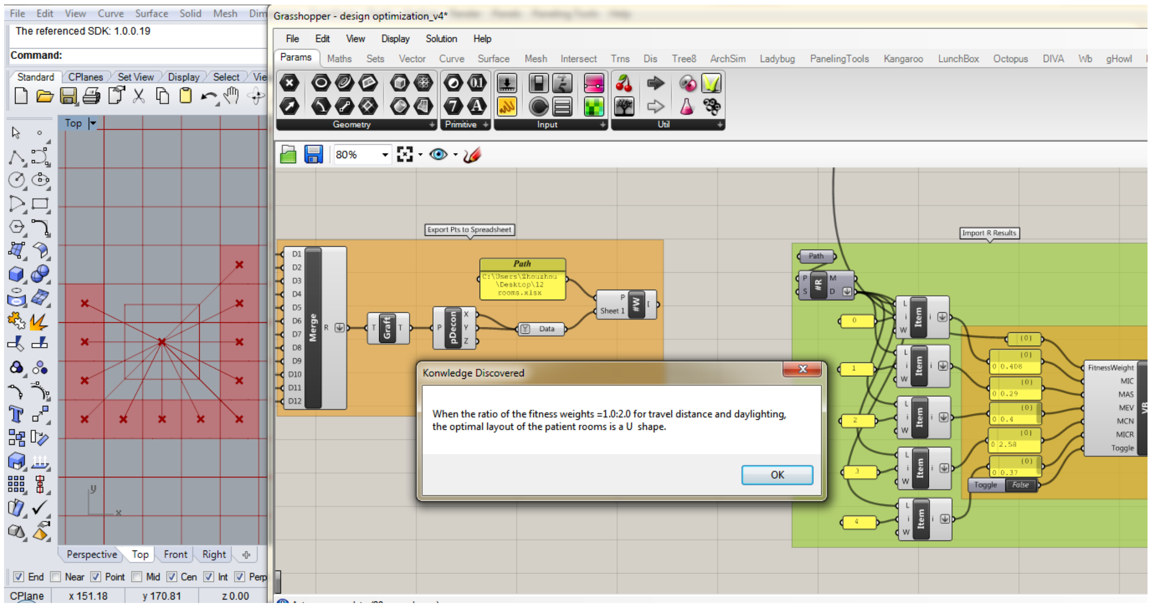
Command:

Konwledge Discovered

When the ratio of the fitness weights =1.0:1.0 for travel distance and daylighting, the optimal layout of the patient rooms is a circle shape.

OK

Autosave complete (24 seconds ago)


Grasshopper - design optimization_v4*

The referenced SDK: 1.0.0.19

Command:

Konwledge Discovered

When the ratio of the fitness weights =1.0:2.0 for travel distance and daylighting, the optimal layout of the patient rooms is a U shape.

OK

Figure 5.9: The Simple Messages Representing Generated Design Knowledge in the Case Study are Automatically Printed in Grasshopper for the Nursing Unit Layout Optimization with Different Weights for the Fitness Functions: (1) Fitness $_{\text{Travel Distance}}$: Fitness $_{\text{Daylighting}}$ = 1:1 (Top); and (2) Fitness $_{\text{Travel Distance}}$: Fitness $_{\text{Daylighting}}$ = 1:2 (Bottom).

5.4 Conclusions and Discussions

When MIC is used in the optimization process, the system is able to automatically identify some simple shapes in the final form of the design (such as a circle, square, etc.) and generate correlations between the shapes as design parameters and the corresponding design performance. These correlations or knowledge can be used to form design guidelines for specific design problems in future design projects with similar design variables and objectives. In the future, when more optimization and knowledge generation are conducted, general knowledge can be discovered to form general design guidelines and reduce the need for the costly computing of additional optimizations.

As mentioned earlier, the MINE result is affected by two parameters: the sample size and the alpha value (α). Additional experiments were carried out to investigate how both parameters influence the results. These experiments show that:

1. when the sample size is small, one should use $\alpha = 1$ to get a result that is closer to the theoretical value unless the user is expecting a simple relationship type (such as linear) where no differences show between the small and large α values even if the sample size is small (see Table 5.2). More differences can be seen when the relationship becomes more complex.

Table 5.2 Comparisons of the MINE Results With a Small Sample Size of 12 and α Values of 0.6 (Top) and 1 (Bottom).

Relationship Type	**MIC**	**MAS**	**MEV**	**MCN**	**MIC-R2**	**Sample Size and α Value**
Linear	1	0	1	2	0	12 points total, α=0.6
Parabolic	0.31	0	0.31	2	0.31	
Circle	0.08	0	0.08	2	0.08	
U-shape	0.1	0	0.1	2	0.07	
Square	0	0	0	2	0	

Relationship Type	**MIC**	**MAS**	**MEV**	**MCN**	**MIC-R2**	**Sample Size and α Value**
Linear	1	0	1	2	0	12 points total, α=1
Parabolic	1	0.68	1	2.58	1	
Circle	0.65	0	0.42	3.16	0.65	
U-shape	0.41	0.29	0.4	2.58	0.37	
Square	0.15	0	0.04	2.58	0.16	

2. when the sample size is large (such as 100), changing α from 0.6 to 1 won't make much of a difference in the results (see Table 5.3). Increasing α will result in more computing time. However, the extra execution time in MINE is only approximately 1 second in the present tests. A longer amount of additional time is expected when the sample size is larger.

Table 5.3 Comparisons of MINE Results with a Large Sample Size of 100 and α Values of 0.6 (Top) and 1 (Bottom).

Relationship Type	**MIC**	**MAS**	**MEV**	**MCN**	**MIC-R2**	**Sample Size and α Value**
Linear	1	0	1	2	0	
Parabolic	1	0.68	1	3.9	1	
Circle	0.61	0	0.32	3.16	0.61	100 points total, α= 0.6
U-shape	0.79	0.59	0.79	2.58	0.79	
Square	0.55	0	0.25	3.58	0.55	

Relationship Type	**MIC**	**MAS**	**MEV**	**MCN**	**MIC-R2**	**Sample Size and α Value**
Linear	1	0	1	2	0	
Parabolic	1	0.68	1	3.9	1	
Circle	0.76	0	0.49	3.58	0.76	100 points total, α= 1
U-shape	0.79	0.59	0.79	2.58	0.79	
Square	0.55	0	0.25	3.58	0.55	

3. for simpler relationship types such as those that are linear, sample size and α value do not matter.

Besides using optimization with the MIC technique, other existing and future data mining methods are possible for use in automatically discovering design knowledge using our proposed system.

It is important to understand that the discovered knowledge is based on specific design variables and constraints, and cannot simply be generalized to other design problems without further tests. For example, in nursing unit design optimization, when the number of patient rooms is specific and with certain constraints, the optimized results show that a circular layout of units will provide an optimal solution for the nurse travel distance and daylighting objectives. However, we cannot claim that a circular layout of units is the best solution in other situations. Nonetheless, as more constraints are added into the optimization problems, new knowledge can be generated to refine the previously discovered knowledge. The method of generating new knowledge - correlations between the optimal design performance and the design parameters (decision variables) - will become more generalizable and applicable to discovering knowledge that will resolve real world problems. The generated knowledge will, in turn, improve the efficiency of the optimization process, and even serve to provide guidelines for future designs and reduce the need for simulation and optimization during the design process.

CHAPTER VI

DISCUSSIONS AND CONCLUSIONS

This chapter discusses the *reliability* and *validity* of the methods of this study, and provides a summary of the study, its findings, and future work.

6.1 Reliability and Validity

The *reliability* of a method measures the degree to which the method provides consistent results. To be considered reliable, the same experiment must be able to produce the same results when performed by other researchers. Reliability is crucial to establishing a study as reliable, and being considered reliable enhances the strength of that study's results (Shuttleworth, 2008). An evaluation of the *validity* of a method considers whether that method meets scientific research standards (Shuttleworth, 2008). All studies must be considered both reliable and valid if they are to receive recognition in the scientific community. *Internal validity* indicates the extent to which cause-effect or causal relationships about a study is assured with minimal bias (Brewer, 2000). *External validity* measures the degree to which the results of a study can be broadly generalized in other circumstances (Louis & Jolley, 2012). The internal validity of this study is discussed in Section 6.1.1 and Section 6.1.2. The external validity needs to be verified in future studies because this study uses specific design objectives, fitness functions and tools. More discussion about future study is in Section 6.3.

The reliability of this study can be ensured, due to the following: (1) The methods developed in this study are based on mathematical deduction or computational logics, and the process is presented in the following discussion; (2) not only the methods, but also the variables used in each step of this study are presented to allow other researchers to duplicate the results; (3) no human judgment involved in this study compromises the reliability of this study.

The proposed framework has two steps. The first step is a design optimization process that finds the optimal design based on the given objectives, and the second step is a design knowledge discovery process that can help with future designs by converting the knowledge discovered into design guidelines. The validation of this proposed framework can be broken down into two parts: (1) the validation of the design optimization process - whether the optimal design can be obtained by using design optimization; and (2) the validation of the design knowledge discovery process – whether or not accurate knowledge can be discovered. If both can be verified, then it is safe to say that the proposed process can actually lead to improvement in both design optimization methods and knowledge discovery methods.

The validation of the first step - whether the optimal design can be obtained by using design optimization - is discussed below.

6.1.1 How to Identify the Optimal Design

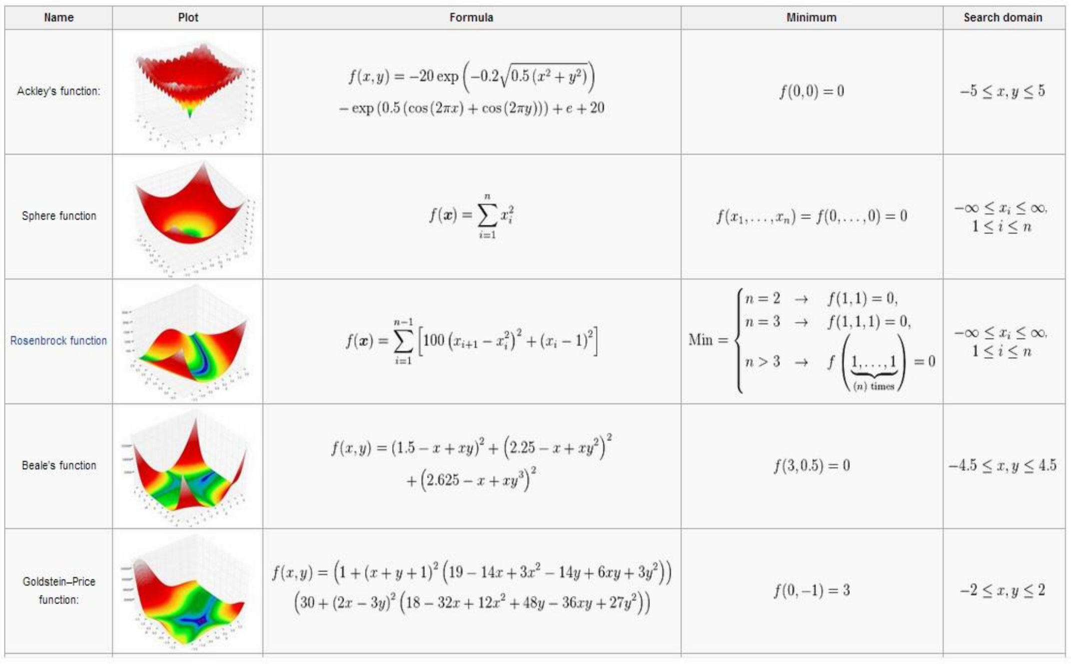
Optimization is the search for the best set of solutions to a system or a problem with explicit objective(s), variables, and constraints (Radford & Gero, 1987). Numerous optimization methods can be used to conduct such a search, such as a gradient search (Salomon, 1998), linear programming and nonlinear programming (Luenberger, 1973), quadratic programming (Frank & Wolfe, 1956), a stochastic search (Goel & Richter, 1974), genetic algorithms (Holand, 1975), discrete methods, etc. A discussion regarding the differences between differential calculus, linear programming, dynamic programming, and genetic algorithms is presented in Chapter 2, Section 2.3.2.

Among the various kinds of optimization methods, some approaches (such as differential calculus, linear programming, and dynamic programming) belong to the area of mathematical optimization. Mathematical optimization searches for the best solution(s) in a set of available alternatives. It is useful in solving analytical problems in mathematics, economics, and computer science (Dantzig, 2010). For analytically solvable problems in mathematical optimization, it is mathematically guaranteed that there is at least one optimal solution or a set of optimal solutions, and the optimal solution(s) can be proven mathematically. As a result, researchers (e.g. Holland, 1992; Forrest, 1993) have suggested using mathematical optimization instead of evolutionary algorithms (such as genetic algorithms) to tackle analytically solvable problems. Because evolutionary algorithms are based on biological evolution, there is no mathematically perfect solution in nature or in any problem of biological adaptation (Marczyk, 2004).

However, mathematical optimization is not suitable for all optimization problems, especially architectural problems, due to the nature of architectural problems that include discrete, nonlinear, and stochastic decision variables (Mackenzie & Gero, 1987). For optimization methods (such as evolutionary algorithms) that cannot be proved mathematically, other methods can be used to validate the optimal solutions to a problem.

1. Test functions. First, common standard test functions can be used to evaluate and test the efficiency and reliability of optimization algorithms (Andrei, 2008). Figure

6.1 displays several standard test functions for single-objective optimization problems. The examples offered here are based on the work of Back (1996), Haupt & Haupt (2004), and Oldenhuis (2009). The first column lists the names of the test functions. The second column outlines the three-dimensional landscapes of the test functions. The third and fourth columns include the objective functions and the optimal (minimized) solutions. The last column is the search domain for the optimal solutions.



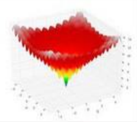
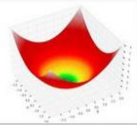
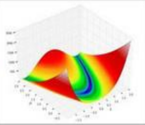
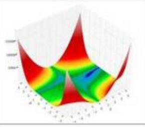
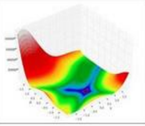
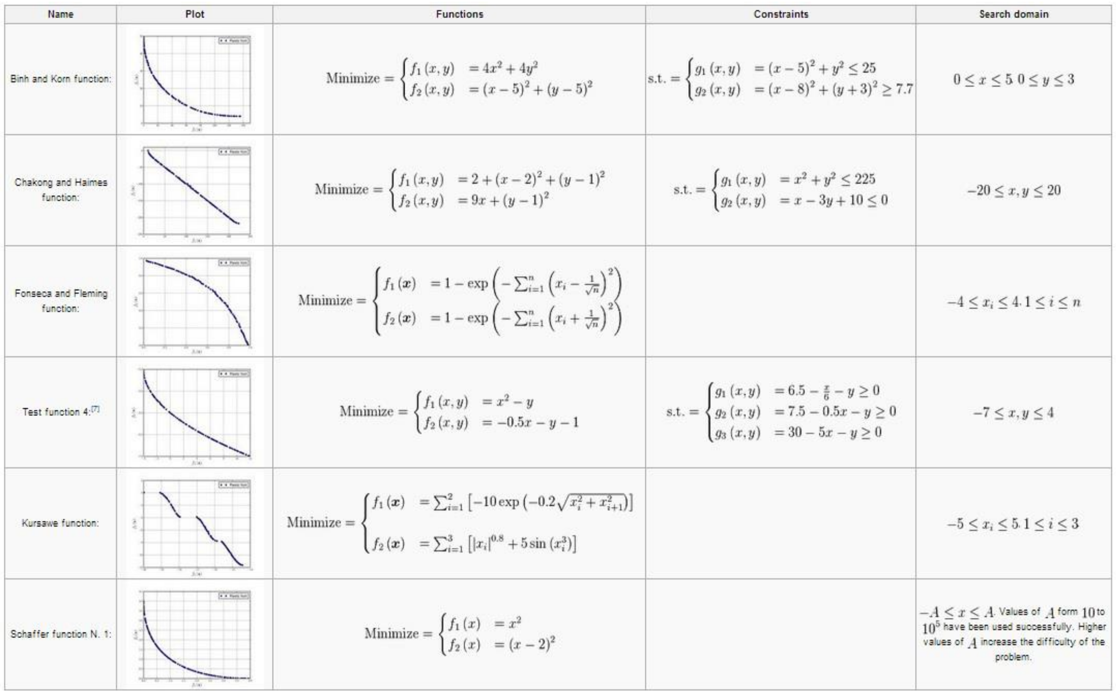
Name	Plot	Formula	Minimum	Search domain
Ackley's function:		$f(x,y) = -20\exp\left(-0.2\sqrt{0.5\left(x^2+y^2\right)}\right) - \exp\left(0.5\left(\cos\left(2\pi x\right)+\cos\left(2\pi y\right)\right)\right) + e + 20$	$f(0,0) = 0$	$-5 \le x, y \le 5$
Sphere function		$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$	$f(x_1, \dots, x_n) = f(0, \dots, 0) = 0$	$-\infty \le x_i \le \infty$, $1 \le i \le n$
Rosenbrock function		$f(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2 + \left(x_i-1\right)^2\right]$	$\text{Min} = \begin{cases} n=2 & \rightarrow & f(1,1)=0, \\ n=3 & \rightarrow & f(1,1,1)=0, \\ n>3 & \rightarrow & f\left(\underbrace{1,\dots,1}_{(n)\text{ times}}\right)=0 \end{cases}$	$-\infty \le x_i \le \infty$, $1 \le i \le n$
Beale's function		$f(x,y) = \left(1.5-x+xy\right)^2 + \left(2.25-x+xy^2\right)^2 + \left(2.625-x+xy^3\right)^2$	$f(3, 0.5) = 0$	$-4.5 \le x, y \le 4.5$
Goldstein–Price function:		$f(x,y) = \left(1+\left(x+y+1\right)^2\left(19-14x+3x^2-14y+6xy+3y^2\right)\right)\left(30+\left(2x-3y\right)^2\left(18-32x+12x^2+48y-36xy+27y^2\right)\right)$	$f(0,-1) = 3$	$-2 \le x, y \le 2$

Figure 6.1 Test Functions for the Single-Objective Optimization Problem (Image Source: Wikipedia: http://en.wikipedia.org/wiki/Test_functions_for_optimization).

Figure 6.2 includes some common test functions in multi-objective optimization problems. The examples are taken from Deb (2001) and Binh and Korn (1997). The first column lists the names of the functions. The second column describes the plot of the

Pareto optimal sets. The third and fourth columns depict the objective functions and the constraints. The last column outlines the search domain of the optimal solutions.



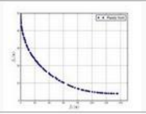
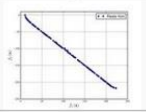
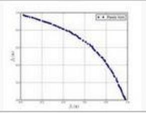
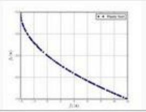
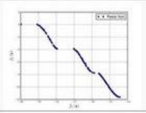
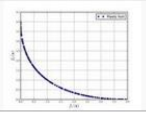
Name	Plot	Functions	Constraints	Search domain
Binh and Korn function:		$\text{Minimize} = \begin{cases} f_1(x,y) & = 4x^2 + 4y^2 \\ f_2(x,y) & = (x-5)^2 + (y-5)^2 \end{cases}$	$\text{s.t.} = \begin{cases} g_1(x,y) & = (x-5)^2 + y^2 \leq 25 \\ g_2(x,y) & = (x-8)^2 + (y+3)^2 \geq 7.7 \end{cases}$	$0 \leq x \leq 5, 0 \leq y \leq 3$
Chakong and Haimes function:		$\text{Minimize} = \begin{cases} f_1(x,y) & = 2 + (x-2)^2 + (y-1)^2 \\ f_2(x,y) & = 9x + (y-1)^2 \end{cases}$	$\text{s.t.} = \begin{cases} g_1(x,y) & = x^2 + y^2 \leq 225 \\ g_2(x,y) & = x - 3y + 10 \leq 0 \end{cases}$	$-20 \leq x, y \leq 20$
Fonseca and Fleming function:		$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) & = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) \\ f_2(\boldsymbol{x}) & = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \end{cases}$		$-4 \leq x_i \leq 4, 1 \leq i \leq n$
Test function 4:[7]		$\text{Minimize} = \begin{cases} f_1(x,y) & = x^2 - y \\ f_2(x,y) & = -0.5x - y - 1 \end{cases}$	$\text{s.t.} = \begin{cases} g_1(x,y) & = 6.5 - \frac{x}{6} - y \geq 0 \\ g_2(x,y) & = 7.5 - 0.5x - y \geq 0 \\ g_3(x,y) & = 30 - 5x - y \geq 0 \end{cases}$	$-7 \leq x, y \leq 4$
Kursawe function:		$\text{Minimize} = \begin{cases} f_1(\boldsymbol{x}) & = \sum_{i=1}^{2}\left[-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right] \\ f_2(\boldsymbol{x}) & = \sum_{i=1}^{3}\left[\lvert x_i\rvert^{0.8} + 5\sin\left(x_i^3\right)\right] \end{cases}$		$-5 \leq x_i \leq 5, 1 \leq i \leq 3$
Schaffer function N. 1:		$\text{Minimize} = \begin{cases} f_1(x) & = x^2 \\ f_2(x) & = (x-2)^2 \end{cases}$		$-A \leq x \leq A$. Values of A form 10 to 10^5 have been used successfully. Higher values of A increase the difficulty of the problem.

Figure 6.2 Test Functions for Multi-Objective Optimization Problems (Image Source: Wikipedia: http://en.wikipedia.org/wiki/Test_functions_for_optimization).

More test functions can be found in (Bingham, 2014). The above mentioned test functions are useful for validating new algorithms or comparing new algorithms to existing algorithms. In my study, I used the Beale's function and the Goldstein-Price function to validate Galapagos, the single-objective optimization tool. Figure 6.3 and Figure 6.4 show the two functions and Figure 6.5 and Figure 6.6 show the corresponding results in Galapagos. The results indicate that Galapagos works well for single-objective optimization and can quickly find the accurate minimum values in both test functions.

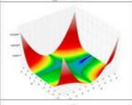
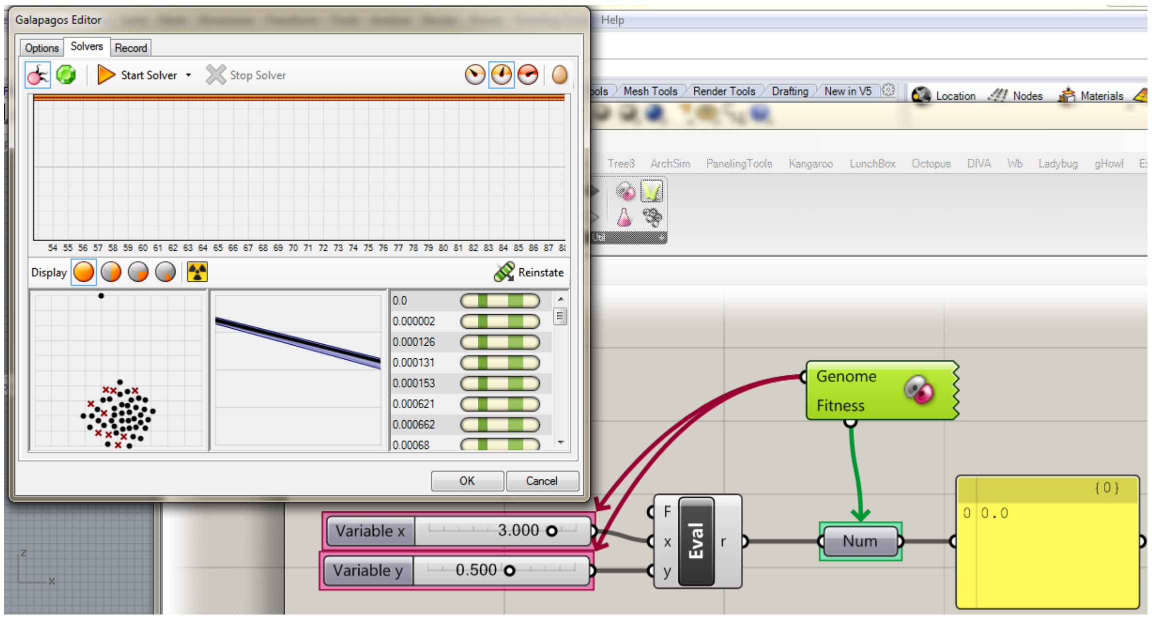
Name	Plot	Formula	Minimum	Search domain
Beale's function		$f(x,y) = (1.5 - x + xy)^2 + \left(2.25 - x + xy^2\right)^2 + \left(2.625 - x + xy^3\right)^2$	$f(3, 0.5) = 0$	$-4.5 \le x, y \le 4.5$

Figure 6.3 Beale's Test Function for Single-Objective Optimization (Image Source: Wikipedia: http://en.wikipedia.org/wiki/Test_functions_for_optimization).



```
0.0
0.000002
0.000126
0.000131
0.000153
0.000621
0.000662
0.00068
```

Variable x 3.000
Variable y 0.500
0 0.0

Figure 6.4 The Minimum Value Found by Galapagos in the Beale's Test Function (x = 3, y = 0.5, and f(x,y) = 0).

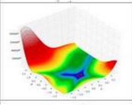
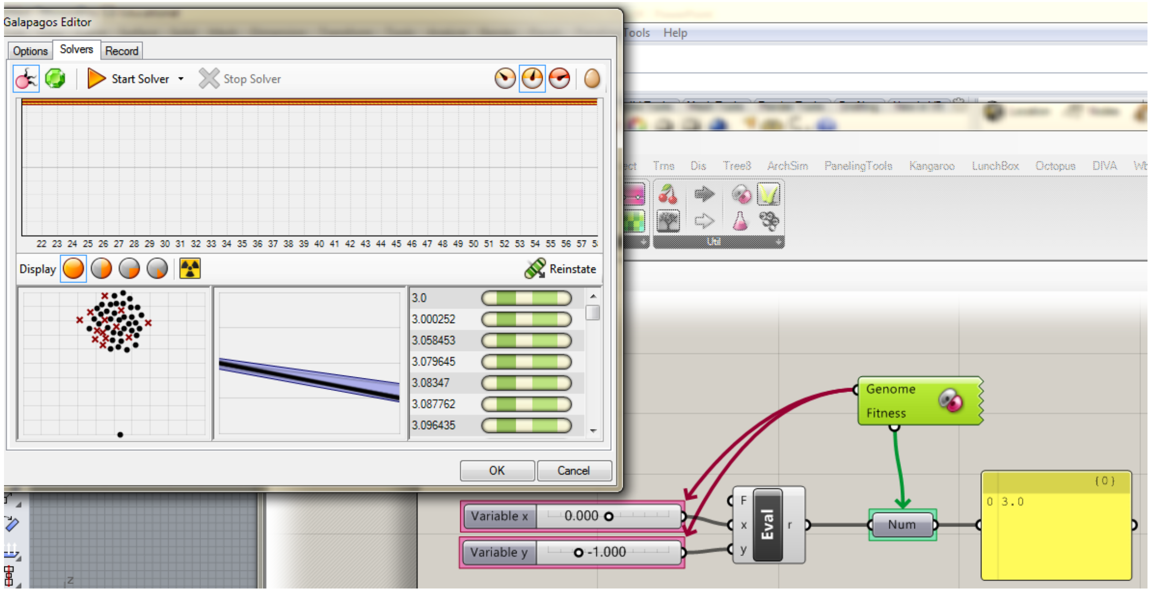
Name	Plot	Formula	Minimum	Search domain
Goldstein–Price function:		$f(x,y) = \left(1 + (x+y+1)^2 \left(19 - 14x + 3x^2 - 14y + 6xy + 3y^2\right)\right) \left(30 + (2x - 3y)^2 \left(18 - 32x + 12x^2 + 48y - 36xy + 27y^2\right)\right)$	$f(0, -1) = 3$	$-2 \le x, y \le 2$

Figure 6.5 Goldstein-Price Test Function for Single-Objective Optimization (Image Source: Wikipedia: http://en.wikipedia.org/wiki/Test_functions_for_optimization).

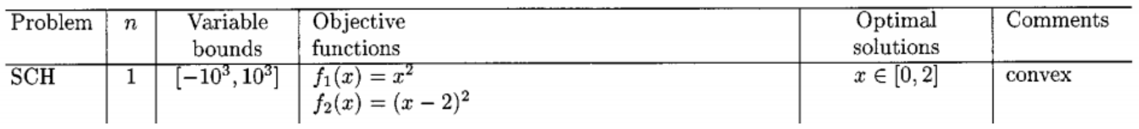


```
3.0
3.000252
3.058453
3.079645
3.08347
3.087762
3.096435
```

```
Variable x   0.000
Variable y   -1.000
0 3.0
```

Figure 6.6 The Minimum Value Found by Galapagos in the Goldstein-Price Test Function (x = 0, y=-1, and f(x,y) = 3).

Octopus, the multi-objective optimization tool used in this study, is validated by the SCH test function. The SCH test function was used in Deb et al. study (2002) to verify the multi objective Genetic Algorithm NSGA-II. There is one variable in the SCH test function; the bounds of that variable are from -10^3 to 10^3. This problem has two objective functions, and there is no constraint. More details about the SCH test function can be found in Figure 6.7.



Problem	n	Variable bounds	Objective functions	Optimal solutions	Comments
SCH	1	$[-10^3, 10^3]$	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$	$x \in [0, 2]$	convex

Figure 6.7 SCH Test Function for Multi-Objective Optimization (Image Source: Deb, et al. 2002).

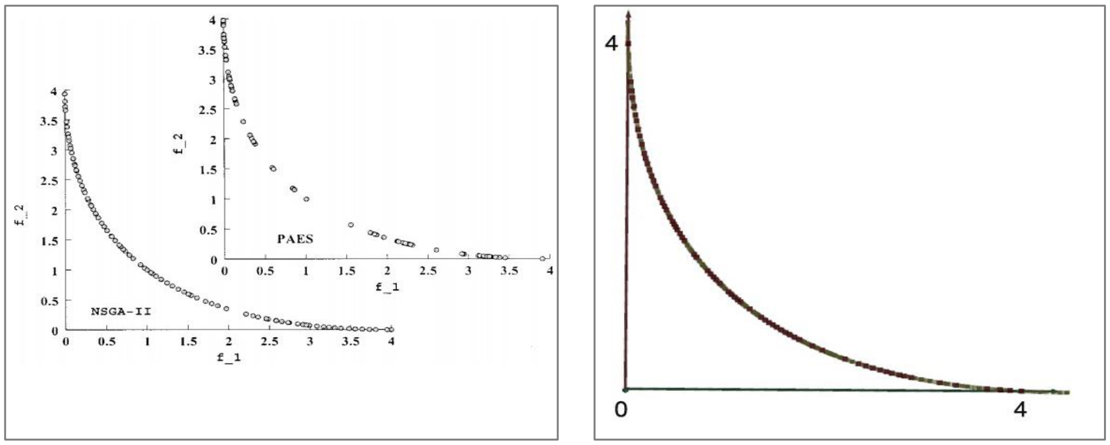


Figure 6.8 Left: The Pareto Optimal Set Found by NSGA-II and PAES for the SCH Test Function (Image Source: Deb, et al. 2002). Right: The Pareto Optimal Set Found by Octopus for the SCH Test Function.

What we found when comparing the SCH test function results among NSGA-II, PAES (see Figure 6.8, left), and SPEA-2 genetic algorithm that is used by Octopus is that Octopus finds a similar spread of solutions (see Figure 6.8, right). The reliability and efficiency of the algorithms for multi-objective optimization used in this study can be ensured.

2. The simplified version of the problem and exhaustive search. Test functions can be used to verify optimization algorithms. However, for a particular optimization problem we must ask: how do we verify that the results are the real optimized solutions (especially in healthcare design) when design problems are usually very complex and involve multiple design objectives? For these kinds of complex problems, the optimal solutions cannot be known in advance in order to verify the results. However, a complex problem can be converted into a simplified version of the original problem and experiment upon the methods and workflow with this simplified problem. This simplified problem then can be verified by converting it into an analytically solvable problem, or by using an exhaustive search. It is generally impossible to use an exhaustive search for a complex problem because the search space is too big and it may take a prohibitively long time for computers to calculate the solution. However, the search space for a simplified problem is much smaller (compared to the original problem) so an exhaustive search is feasible to verify these methods. After carefully verifying the methods, we can apply them to complex problems and trust the results to be optimal.

3. A Case Study. A simplified study – a parametric form-finding for a nursing unit design – is used in this research to verify the methods. The design objectives are to: (1) minimize nurses' travel distance from the nurses' station to each patient room; and (2) obtain an optimal level of daylight illuminance in all patient rooms, based on the LEED standard (healthcare supplement) (USGBC 2009). The two objectives are converted into a single objective by using a weighted sum of the objective functions with pre-defined (architects' subjective) weights. For any nursing unit layout solution, the fitness scores for travel distance and daylight illuminance can be defined using the function below.

$$\textit{Overall Fitness} = \textit{Weight}_{\textit{travel distance}} \times \textit{Fitness}_{\textit{travel distance}} + \textit{Weight}_{\textit{daylighting}} \times \textit{Fitness}_{\textit{daylighting}}$$

The optimized results are not surprising (see Figure 6.9). The image on the left is the result when using equal weights for travel distance and daylighting. The image on the right is the result when the weights of the travel distance and daylighting are at a 1:2

ratio. If we increase the weight of daylighting performance from 1 to 2, then fewer rooms will appear on the north side of the nurses' station because a room will receive the least amount of daylight when it is on the north side with a north-facing window. The result – U shape layout – confirms the expectation.

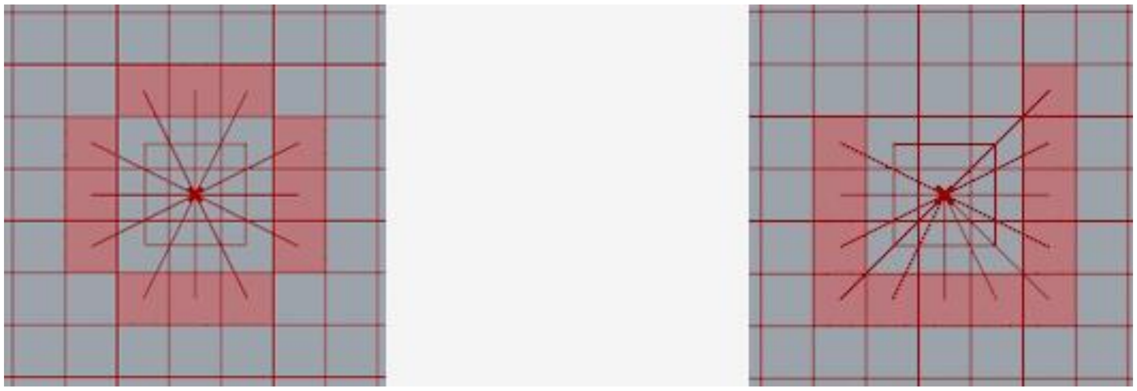


Figure 6.9 The Final Results of the Nursing Unit Layout Optimization with Different Weights in the Fitness Functions: (1) Fitness $_{\text{Travel Distance}}$: Fitness $_{\text{Daylighting}}$ = 1:1 (Left) (2) Fitness $_{\text{Travel Distance}}$: Fitness $_{\text{Daylighting}}$ = 1:2 (Right).

Although we can say that the final results are consistent with our intuitive expectations, intuition cannot be used to verify these results. Exhaustive search can be used in order to validate whether or not the results have truly been optimized. However, compared to GA, exhaustive search may require a long time to complete. There are 12 variables in the simplified study. The number of possible locations for the first variable is 225-9=216 (225 is the total number of cells, and there are 9 reserved cells for the nurses' station and 8' corridor); and the number of possible locations for the second variable is 216-1=215 (216 minus the location of the first variable). Therefore, the total population is $216\times215\times214\times\ldots\times205 = 7.6\times10^{27}$. It will take the computer a long time to complete the calculation, though theoretically the optimal result can be verified mathematically. For simpler problems with less alternatives, optimal solutions can be found through exhaustive search. In this study, the exhaustive search for the case study optimization was not conducted for verification because of the time limitation. Instead, the optimization tool (Galapagos)'s validation and qualitative examination of the optimization results of the case study were used for validation of the methods.

6.1.2 How to Ensure the Discovered Knowledge is Accurate

The purpose of the second part of this study is to find design knowledge - correlations between optimal solutions and design parameters - using design optimization and techniques such as Maximal Information Coefficient (MIC) (Reshef et al., 2011). MIC has been proven in its generality (MIC captures relationships with data of sufficient size) and equitability (different types of data with the same noisy relationships receive similar results in MIC) (Reshef et al., 2011). In this study, MIC is used to allow the computer to automatically identify some simple shapes of the final forms (spatial layouts) of the design (such as a circle or a U shape.), and generate the correlation of each shape with the corresponding design performance.

The validation of the design knowledge discovery process using the MIC technique is presented in a series of case study tests with simple forms or relationships (such as linear relationship, nonlinear relationship, parabolic relationship, sinusoidal relationship, circle, and square). Please refer to Chapter 5, Section 5.3.2 for more details about and a greater discussion of this series of tests. The purpose of testing on different shapes is to verify that MIC will give distinct values to different forms or relationships. The examples shown in Section 5.3.2 indicate that different shapes/relationships will have different metrics values in MINE. In this study, when the design layout optimization is finished, the shapes of the optimized design can be classified into one of the shapes/relationships using the values in MINE. By doing this, we can be sure that no incorrect knowledge can be generated.

6.2 Conclusion and Discussion

Design optimization in the early stages of architectural design received wide attention in recent years. This study presents computational methods for creating and improving a closed loop of design optimization and knowledge discovery in architecture, which aims at addressing some of the shortcomings of traditional design optimization.

The first part of this study – design knowledge-assisted optimization improvement – presents two techniques: offline simulation and Divide & Conquer (D&C). They demonstrate great time savings in building simulation and optimization process, and can provide a larger GA search space in the same amount of time, which offers a better chance of finding the optimal design. The second part of this study – optimization-based knowledge discovery – describes a new design knowledge discovery system where design knowledge can be discovered from optimization through an automatic data mining approach. The discovered knowledge has the potential to further help improve the efficiency of the optimization method, thus forming a closed loop of improving optimization and knowledge discovery.

The method of D&C was proved mathematically before undertaking the task, and the method of offline simulation was supported by comparing the time spent before and after the use of the method. The validation of the use of data mining technique was demonstrated in a series case study tests with simple forms or relationships. A simplified nursing unit layout design was used as a case study in order to verify the proposed methods.

It should be noted that the main purpose of this study is to develop and test new methods and prototypes in order to improve design optimization and optimization-based design knowledge discovery. The case study used here was merely to validate the methods and to demonstrate the work process. Because of that, the design objectives (minimize walking distance and maximize daylighting in a patient unit), the fitness function (see Chapter 4, Section 4.4.4), and tools (Rhino, Grasshopper, DIVA, Galapagos, Octopus, MIC, etc.) can be substituted with different or more sophisticated corresponding elements. The methods such as reusing offline simulation, breaking down a complex problem into easier problems, and using the data mining technique to extract design knowledge will remain the same for different design problems. These proposed methods can be applied to fields other than architectural design after being carefully tested. More tests are needed to investigate the generalizability of the methods. Necessary modification to the methods may be needed for a sophisticated problem or with different tools.

6.3 Limitations and Future Study

There are some limitations in the proposed methods. No mutual feedback exists among genomes in the present proposed technique of offline simulation, because the simulation result of each genome is pre-computed. One possible solution is to find and pre-computer all possible situations in advance, however, this may increase the computing time.

A similar limitation occurs in D&C method, that a newly added genome has no impact on the previously placed genomes. When genomes are not independent of one another, designers may group the sub-problems into sets of sub-problems to enable feedback among genomes within each set of sub-problems. The detailed discussion about these limitations can be found in Section 4.5.

In the design knowledge discovery system, the learned knowledge presented in this study cannot simply be generalized to other design problems because it was limited to specific design variables, constraints and design objectives. However, more generalizable knowledge can be obtained when more tests with different design variables, constraints and design objectives are performed.

In the future, more realistic and sophisticated case studies should be tested. The detailed planning for future studies is described as below.

First, a real world healthcare design project - Camarillo State Hospital Children's Unit Addition Design[1] - will be used as a case study to test the offline simulation and D&C methods, as well as the knowledge discovery method. In this future study, the method for walking distance calculation will be improved, compared to the how it is done in the simplified case study. The new walking distance calculation method will consider nurses' actual walking patterns. The paths to other supporting areas such as medication room, nutrition room, clean utility and soiled utility etc. will also be taken

[1] The Camarillo State Hospital Children's Unit Addition Design program is generously provided by Professor Mardelle Shepley.

into consideration. Path finding algorithms will be used in generating paths. Single-objective optimization will first be used and then the proposed methods will be expanded to Pareto optimization for multiple design objectives to ensure their practicability in the real world architectural design process.

Second, other design objectives, fitness functions, design tools will be tested with the proposed methods to ensure the external validity of this study. More complex case studied and detailed building simulation modeling will be used.

Third, future study is needed to further examine the possible solutions for solving the above-mentioned limitations of both the offline simulation and D&C methods.

Last, the proposed methods and prototypical tools need to be presented to the management-oriented audiences to promote the research of optimization and the improved methods in the industry.

REFERENCES

Abboud, K., & Schoenauer, M. (2002, January). Surrogate deterministic mutation: Preliminary results. In *Artificial Evolution* (pp. 104-116). Springer Berlin Heidelberg.

Ahn, Ki-Uhn., Young-Jin Kim., Deu-Woo Kim., Sung-Hwan Yoon., & Cheol-Soo Par. (2013) Difficulties and issues in simulation of a high-rise office building. In *Proceedings of the 13th Conference of International Building Performance Simulation Association*, pp 842-831.

Anderson, K. S., & Hsu, Y. (1999). Genetic crossover strategy using an approximation concept. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on (Vol. 1). *IEEE*.

Andrei, N. (2008). An unconstrained optimization test functions collection. *Adv. Model. Optim*, 10(1), 147-161.

Ansys, I. (2007). *ANSYS Advanced Analysis Techniques Guide*. Ansys Help.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50-58.

ASHRAE (2007). *ANSI/ASHRAE/IESNA Standard 90.1- 2007, Energy Standard for Buildings Except Low-Rise Residential Buildings*. Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers.

Attia, S., Gratia, E., De Herde, A., & Hensen, J. L. (2012). Simulation-based decision support tool for early stages of zero-energy building design. *Energy and Buildings*, 49, 2-15.

Back, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford Univ. Press.

Beasley, D., Martin, R. R., & Bull, D. R. (1993). An overview of genetic algorithms: Part 1. Fundamentals. *University Computing*, 15, 58-58.

Bellman, R. (1956). Dynamic programming and Lagrange multipliers. Proceedings of *the National Academy of Sciences of the United States of America*, 42(10), 767.

Berry, M. J., & Linoff, G. S. (2004). *Data Mining Techniques: for Marketing, Sales, and Customer Relationship Management*. John Wiley & Sons.

Besserud, K., Skidmore, O., & Merrill, L. L. P. (2008). *Architectural Genomics*. Silicon+ Skin> Biological Process and Computation, 238-245.

Binh, T. T., & Korn, U. (1997, June). *MOBES:* A multiobjective evolution strategy for constrained optimization problems. *Proceedings* of *the Third International Conference on Genetic Algorithms* (Mendel 97) (pp. 176-182).

Bingham, D. (2014, September). *Optimization Test Problems*. Retrieved April 13, 2015, from http://www.sfu.ca/~ssurjano/optimization.html

Buche, D., Schraudolph, N. N., & Koumoutsakos, P. (2005). Accelerating evolutionary algorithms with Gaussian process fitness function models. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions* on, 35(2), 183-194.

Budde, R., Kautz, K., Kuhlenkamp, K., & Zllighoven, H. (2011). *Prototyping: An Approach to Evolutionary System Development*. Springer Publishing Company, Incorporated.

Brewer, M. B. (2000). Research design and issues of validity. *Handbook of Research Methods in Social and Personality Psychology,* 3-16.

Caban, J. J., Bagci, U., Mehari, A., Alam, S., Fontana, J. R., Kato, G. J., & Mollura, D. J. (2012, August). Characterizing non-linear dependencies among pairs of clinical variables and imaging data. *Proceedings of Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE* (pp. 2700-2703). IEEE.

Chakrabarti, S., Ester, M., Fayyad, U., Gehrke, J., Han, J., Morishita, S., ... & Wang, W. (2006). Data mining curriculum: A proposal (Version 1.0). *Intensive Working Group of ACM SIGKDD Curriculum Committee*.

Chen, K. C., & Chen, C. Y. C. (2011). Stroke prevention by traditional Chinese medicine? *A Genetic Algorithm, Support Vector Machine and Molecular Dynamics Approach*. Soft Matter, 7(8), 4001-4008.

Choudhary, R., Malkawi, A., & Papalambros, P. Y. (2003). A hierarchical design optimization framework for building performance analysis. *Proceedings of the 8th IBPSA Conference*, Eindhoven, NL.

Choudhary, R. (2004) *A Hierarchical Optimization Framework for Simulation-Based Architectural Design*. Diss. University of Michigan.

Choudhary, R., & Michalek, J. (2005). Design optimization in computer aided architectural design. Proceedings of CAADRIA, *The Association for Computer-Aided Architectural Design Research in Asia*. New Delphi, India, 149-158.

Claussnitzer, S., Katz, N., Shaxted, M., Park. S.K., & Yori, R. (October 20-22 2014). Workshop – High-throughput computing (HTC) for parametric exploration by SOM. *Proceedings of the Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, October 23-25, 2014. Los Angeles, California.

Clifton, C., & Marks, D. (1996, May). Security and privacy implications of data mining. In *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery* (pp. 15-19).

Cios, K. J., & Moore, G. W. (2002). Uniqueness of medical data mining. *Artificial Intelligence in Medicine, 26*(1), 1-24.

Coffey, B. (2012). *Using building simulation and optimization to calculate lookup tables for control*. PhD Dissertation.

Coffey, B. (2013). Approximating model predictive control with existing building simulation tools and offline optimization. *Journal of Building Performance Simulation*, 6(3), 220-235.

Conner J. M., & Nelson E. C. (1999). Neonatal intensive care: Satisfaction measured from a parent's perspective. *Pediatrics*, 103 (1 Suppl E), 336–349.

Cooper, L., & Cooper, M. W. (1981). *Introduction to dynamic programming* (pp. 197-207). New York: Pergamon Press.

Corbin, C. D., Henze, G. P., & May-Ostendorp, P. (2013). A model predictive control optimization environment for real-time commercial building application. *Journal of Building Performance Simulation*, 6(3), 159-174.

Dantzig, G. B. (2010). *The nature of mathematical programming*. Mathematical Programming Glossary.

Deb, K. (2001*). Multi-objective optimization using evolutionary algorithms* (Vol. 16). John Wiley & Sons.

Deb, K. (2012). *Optimization for engineering design: Algorithms and examples*. PHI Learning Pvt. Ltd.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: *NSGA-II. Evolutionary Computation, IEEE Transactions on*, 6(2), 182-197.

Dettenkofer M., Seegers S., Antes G., Motschall E., Schumacher M. ,& Daschner F. D. (2004). Does the architecture of hospital facilities influence nosocomial infection rates? A systematic review. *Infection Control and Hospital Epidemiology*, 25 (1), 21–25.

DeVinne P (Ed.) (1987) *The American heritage illustrated encyclopedic dictionary*. Boston: Houghton Mifflin.

Eastman, C., P. Teicholz, R. Sacks, & K. Liston. (2011). *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. Wiley.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3), 37.

Farncombe, T., & Iniewski, K. (Eds.). (2013). *Medical Imaging: Technology and Applications*. CRC Press.

Ferris, R. (1996) Introduction. In SS William (ED.), *Reflections on architectural practices in the nineties*. New York: Princeton Architectural Press, pp.8-11.

Filosi, M., Visintainer, R., Albanese, D., Riccadonna, S., Jurman, G., & Furlanello, C. (2014) *Minerva: Minerva: Maximal information-based nonparametric exploration R package for variable analysis*. Retrieved March 23, 2015, from http://cran.r-project.org/web/packages/minerva/, 2014, August 26.

Forrest, S. (1993). Genetic algorithms: principles of natural selection applied to computation. *Science*, *261*(5123), 872-878.

Forrester, A. I., Bressloff, N. W., & Keane, A. J. (2006). Optimization using surrogate models and partially converged computational fluid dynamics simulations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 462(2071), 2177-2204.

Frank, E., Hall, M., Trigg, L., Holmes, G., & Witten, I. H. (2004). Data mining in bioinformatics using Weka. *Bioinformatics, 20*(15), 2479-2481.

Frank, M., & Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2), 95-110.

Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992). Knowledge discovery in databases: An overview. *AI magazine*, 13(3), 57.

Gallas, M. A., & Halin, G. (2013). DaylightGen: From daylight intentions to architectural solutions. *eCAADe* 2, 107-116.

Gen, M. & Cheng, R. (2000). *Genetic algorithms and engineering optimization* (Vol. 7). John Wiley and Sons.

Gerber, D. J., Lin, S. H. E., Pan, B. P., & Solmaz, A. S. (2012, March). Design optioneering: Multi-disciplinary design optimization through parameterization, domain integration and automation of a genetic algorithm. *Proceedings of the 2012 Symposium on Simulation for Architecture and Urban Design* (p. 11). Society for Computer Simulation International.

Gero, J. S. & Louis, S. J. (1995). Improving Pareto optimal designs using genetic algorithms. *Computer ‐ Aided Civil and Infrastructure Engineering*, 10(4), 239-247.

Gole, A. M. (2000). Simulation tools for system transients: an introduction. InPower Engineering Society Summer Meeting, 2000. *IEEE* (Vol. 2, pp. 761-762). IEEE.

Goel, N. S., & Richter-Dyn, N. (1974). *Stochastic models in biology*. Academic Press.

Haber, J. (2010). *Research Questions, Hypotheses, and Clinical Questions*. Retrieved March 27, 2015, from: http://www.us.elsevierhealth.com/media/us/samplechapters/9780323057431/Chapter%2002.pdf

Hall, J., Mani, G., & Barr, D. (1996). Applying computational intelligence to the investment process. *Proceedings of CIFER-96: Computational Intelligence in Financial Engineering. Washington, DC: IEEE Computer Society*.

Harris D. D., Shepley M. M., White R. D., Kolberg K. J. S. ,& Harrell J. W. (2006). The impact of single family room design on patients and caregivers: Executive summary. *Journal of Perinatology*, 26, S38–S48.

Haupt, R. L., & Haupt, S. E. (2004). *Practical genetic algorithms*. John Wiley & Sons.

Hendrich, A., Chow, M., Skierczynski, B., & Lu, Z. (2008). A 36-hospital time and motion study: How do medical-surgical nurses spend their time? *The Permanente Journal*, 12(3), 25–34.

Hensen, J. L., & Lamberts, R. (Eds.). (2012). *Building performance simulation for design and operation*. Routledge.

Hevner, A.R.; March, S.T.; and Park, J. (2004). Design research in information systems research. *MIS Quarterly*, 28, 1, 75 ‑ 105.

Hochman, H. M., & Rodgers, J. D. (1969). Pareto optimal redistribution. *The American Economic Review*, 542-557.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press.

Holland, J. (1992) Genetic algorithms. *Scientific American*, July, p. 66-72.

Hong, T., Chou, S. K., & Bong, T. Y. (2000). Building simulation: an overview of developments and information sources. *Building and Environment*, 35(4), 347-361.

Hu, J., & Karava, P. (2014). Model predictive control strategies for buildings with mixed-mode cooling. *Building and Environment*, 71, 233-244.

http://www.exploredata.net/Downloads [3-27-2015]

http://www.exploredata.net/Downloads/MINE-Application [3-27-2015]

Jin, Y., Olhofer, M., & Sendhoff, B. (2000, July). On evolutionary optimization with approximate fitness functions. *In GECCO* (pp. 786-793).

Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1), 3-12.

Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2), 61-70.

Jo, J. H. & Gero, J. S. (1998). Space layout planning using an evolutionary approach. *Artificial Intelligence in Engineering*, 12(3), 149-162.

Keim, D. A. (2002). Information visualization and visual data mining. *Visualization and Computer Graphics, IEEE Transactions on, 8*(1), 1-8.

Kim, D. S., & Shepley, M. M. (2007). Healthcare architects' professional autonomy: interview case studies. *HERD*, 1(2), 14-26.

Kim, D. S., & Shepley, M. M. (2011). Healthcare design complexity, specialized knowledge, and healthcare architects' professional autonomy. *Journal of Architectural and Planning Research*, 28(3), 194-210.

Kim, H. M. (2001). *Target cascading in optimal system design*. Ph. D. dissertation, University of Michigan.

Kim, H. M., et al. (2001). Analytical Target Cascading in Automotive Vehicle Design. *Proceedings of the 2001 ASME Design Automation Conference*, September 9-12, 2001. Pittsburgh, Pennsylvania.

Kim, H., Stumpf, A., & Kim, W. (2011). Analysis of an energy efficient building design through data mining approach. *Automation in Construction*, 20(1), 37-43.

Kociecki, M., & Adeli, H. (2013). Two-phase genetic algorithm for size optimization of free-form steel space-frame roof structures. *Journal of Constructional Steel Research, 90*, 283-296.

Leedy, P. D., & Ormrod, J. E. (2005). *Practical research: Planning and design*. , Prentice-Hall, Upper Saddle River, NJ.

Lehmann, E. L., & Casella, G. (1998). *Theory of point estimation* (Vol. 31). Springer Science & Business Media.

Liao, Y. H., & Sun, C. T. (2001). An educational genetic algorithms learning tool. *IEEE Trans*. Education, 44(2), 20.

Lobos, D., & Donath, D. (2010). The problem of space layout in architecture: A survey and reflections. *Arquitetura Revista*, 6(2), 136-161.

Luenberger, D. G. (1973). *Introduction to linear and nonlinear programming* (Vol. 28). Reading, MA: Addison-Wesley.

Mackenzie, C. A. & Gero, J. S. (1987). Learning design rules from decisions and performances. *Artificial Intelligence in Engineering*, 2(1), 2-10.

Mahmoodabadi, M. J., Safaie, A. A., Bagheri, A., & Nariman-Zadeh, N. (2013). A novel combination of Particle Swarm Optimization and Genetic Algorithm for Pareto optimal design of a five-degree of freedom vehicle vibration model. *Applied Soft Computing*, 13(5), 2577-2591.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, 15(4), 251-266.

Marczyk, A. (2004, 4 23). *Genetic algorithms and evolutionary computation*. Retrieved from http://www.talkorigins.org/faqs/genalg/genalg.html

McGrail, A. J., Gulski, E., Groot, E. R. S., Allan, D., Birtwhistle, D., & Blackburn, T. R. (2002). Datamining techniques to assess the condition of high voltage electrical plant. *CIGRE Paris WG15, 11*.

Michie, D. (1987, October). Current developments in expert systems. *Proceedings of the Second Australian Conference on Applications of Expert Systems* (pp. 137-156). Addison-Wesley Longman Publishing Co., Inc..

Michie, D. (1989). Problems of computer-aided concept formation. *Applications of Expert Systems*, *2*, 310-333.

Michie, D. 1990. March 15 Interview. *AI Week* 7(6): 7–12

Miller, H. J., & Han, J. (Eds.). (2009). *Geographic data mining and knowledge discovery*. CRC Press.

Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

Mitchell, M., & Jolley, J. (2012). Research design explained. *Cengage Learning*.

Morbitzer, C., Stratchan, P. & Simpson, C. (2003). Application of data mining techniques for building simulation performance prediction analysis. *In: Schellen and van der Spoel, ed. Building Simulation '03, 8th International IBPSA Conference*, Eindhoven, Netherlands, September 18-21, 2003, 911-918.

Muhlenbein, H. (1991). Evolution in time and space-the parallel genetic algorithm. *In Foundations of Genetic Algorithms*.

Moon, Y. I., Rajagopalan, B., & Lall, U. (1995). Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3), 2318-2321.

Negendahl, K., Perkov, T., & Heller, A. (2014). Approaching sentient building performance simulation systems. *Proceedings of eCAADe 2014*. September 10-12, 2014. Newcastle, UK.

NSF, 2009. *The challenge of sustainable energy*. National Science Foundation.

Oldenhuis, R. (2009, 2, 28). *Test functions for global optimization algorithms*. Retrieved from http://www.mathworks.com/matlabcentral/fileexchange/23147-test-functions-for-global-optimization-algorithms

Ong, Y. S., Nair, P. B., & Keane, A. J. (2003). Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4), 687-696.

Ong, Y. S., Nair, P. B., & Lum, K. Y. (2006). Max-min surrogate-assisted evolutionary algorithm for robust design. *Evolutionary Computation, IEEE Transactions on*, 10(4), 392-404.

Papalambros, P. Y. (2002). The optimization paradigm in engineering design: promises and challenges. *Computer-Aided Design*, 34(12), 939-951.

Papalambros, P. Y., & Wilde, D. J. (2000). *Principles of optimal design: modeling and computation*. Cambridge university press.

Pearson, K. (1895). Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58(347-352), 240-242.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45-77.

Piateski, G., & Frawley, W. (1991). *Knowledge discovery in databases*. MIT press.

Poole, D., Mackworth, A. & Goebel, R. (1998). *Computational intelligence: A logical approach,* Oxford University Press, New York

Portugal, V. & Guedes, M. (2012, November). Informed parameterization: optimization of building openings generation. *Plea2012 - 28th Conference, Opportunities, Limits & Needs Towards an Environmentally Responsible Architecture,* Lima, Perú. Retrieved from http://www.plea2012.pe/pdfs/T07-20120130-0053.pdf

Potter, M. A. & De Jong, K. A. (1994). A cooperative coevolutionary approach to function optimization. *In Parallel Problem Solving from Nature—PPSN III* (pp. 249-257). Springer Berlin Heidelberg.

Potter, M. A. & De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1), 1-29.

Pries-Heje, J., Baskerville, R., & Venable, J. (2008). Strategies for design science research evaluation. *ECIS 2008 Proceedings*, 1-12.

Quinlan, J. R. (2014). C4. 5: *Programs for machine learning*. Elsevier.

Rahmani Asl, M., Zarrinmehr, S., & Yan, W. (2013). Towards BIM-based Parametric Building Energy Performance Optimization, *Proceedings of The Association for Computer Aided Design in Architecture (ACADIA),* October 24-27, 2013. Cambridge, Ontario, Canada.

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

Rawat, C. D., Shahani, A., Natu, N., Badami, A., & Hingorani, R. (2012). A genetic algorithm for VLSI floor planning. *International Journal of Engineering Science & Advanced Technology*, 2(3), 412-415.

Renner, G. & Ekárt, A. (2003). Genetic algorithms in computer aided design. *Computer-Aided Design*, 35(8), 709-726.

Reshef, D. N., Reshef, Y. A., Finucane, H. K., Grossman, S. R., McVean, G., Turnbaugh, P. J., ... & Sabeti, P. C. (2011). Detecting novel associations in large data sets. *Science*, 334(6062), 1518-1524.

Rutten, D. (2011, March 7). *Define "Fitness"*.... Retrieved December 10, 2014, from https://ieatbugsforbreakfast.wordpress.com/2011/03/07/define-fitness/

Salomon, R. (1998). Evolutionary algorithms and gradient search: similarities and differences. *Evolutionary Computation, IEEE Transactions on, 2*(2), 45-55.

Sarmanov, O. V. (1962). Maximum correlation coefficient (nonsymmetric case). *Selected Translations in Mathematical Statistics and Probability*, 2, 207-210.

Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1), 72-101.

Schnabel, M. A. (2007). Parametric Designing in Architecture. *In Computer-Aided Architectural Design Futures (CAAD Futures)* 2007 (pp. 237-250). Springer Netherlands.

Shi, X. (2011). Design optimization of insulation usage and space conditioning load using energy simulation and genetic algorithm. *Energy*, 36(3), 1659-1667.

Shuttleworth, M. (Oct 20, 2008). *Validity and reliability*. Retrieved Jan 08, 2015 from Explorable.com: https://explorable.com/validity-and-reliability

Simpson, T. W., Mauery, T. M., Korte, J. J., & Mistree, F. (2001). Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA Journal*, 39(12), 2233-2241.

Spivey, W A. (1962). Linear programming. *Science*, 135(3497), 23-27.

Stavric, M., & Marina, O. (2011). Parametric modeling for advanced architecture. *International Journal of Applied Mathematics and Informatics*, 5, 9-16.

Su, Z., Yan, W. (2014) Improving Genetic Algorithm for Design Optimization Using Architectural Domain Knowledge, *Proceedings of the Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*, October 23-25, 2014. Los Angeles, California.

Test functions for optimization. (2015, March 31). Retrieved April 13, 2015, from https://en.wikipedia.org/wiki/Test_functions_for_optimization

TMHC Staff. (2014, March 24). *30 most technologically advanced hospitals in the world*. Retrieved November 9, 2014, from http://www.topmastersinhealthcare.com/30-most-technologically-advanced-hospitals-in-the-world/

Tuckman, B. W., & Harper, B. E. (2012). *Conducting educational research*. Rowman & Littlefield Publishers.

Ulrich, R. (1984). View through a window may influence recovery. *Science*,224(4647), 224-225.

USGBC. (2009). *Green building design and construction*. (2009 Edition ed.). Washington, DC.

Valenzuela, C. L. & Jones, A. J. (1993). Evolutionary divide and conquer (I): A novel genetic approach to the TSP. *Evolutionary Computation*, 1(4), 313-333.

Vaishnavi, V., & Kuechler, W. (2004). *Design research in information systems*.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3), 611-624.

Vierlinger, R. (n.d.). *Octopus*. Retrieved October 30, 2014, from http://www.food4rhino.com/project/octopus?ufh

von Buelow, P., Falk, A., & Turrin, M. (2010). Optimization of structural form using a genetic algorithm to search associative parametric geometry. *Proceedings of Structure and Architecture*.

Walch, J. M., Rabin, B. S., Day, R., Williams, J. N., Choi, K., & Kang, J. D. (2005). The effect of sunlight on postoperative analgesic medication use: a prospective study of patients undergoing spinal surgery. *Psychosomatic Medicine*, 67(1), 156-163.

Wagner, T. (1993). *A general decomposition methodology for optimal system design*. Ph.D. dissertation. University of Michigan.

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

Watson, R. A. (2002). Compositional evolution: interdisciplinary investigations in evolvability, modularity, and symbiosis. *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN-VIII)* (pp. 161-171). Springer.

Welle, B., Rogers, Z., & Fischer, M. (2012). BIM-Centric Daylight Profiler for Simulation (BDP4SIM): A methodology for automated product model decomposition and recomposition for climate-based daylighting simulation. *Building and Environment,* 58, 114-134.

Witten, I. H., & Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

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

Yu, Z., & Dexter, A. (2009). *Simulation based predictive control of low energy building systems using two-stage optimization*. Proc. IBPSA'09, 1562-1568.

Zhou, Z., Ong, Y. S., Nair, P. B., Keane, A. J., & Lum, K. Y. (2007). Combining global and local surrogate models to accelerate evolutionary optimization. Systems, Man, and Cybernetics, Part C: Applications and Reviews, *IEEE Transactions* on, 37(1), 66-76.

Zhu, X. (Ed.). (2007). *Knowledge discovery and data mining: Challenges and realities.* Igi Global.

Zimring, C., Joseph, A., & Choudhary, R. (2004). The role of the physical environment in the hospital of the 21st century: a once-in-a-lifetime opportunity. *Concord, CA: The Center for Health Design*.

Zimring, C. M., Ulrich, R. S., Zhu, X., DuBose, J. R., Seo, H. B., Choi, Y. S., ... & Joseph, A. (2008). A review of the research literature on evidence-based healthcare design. *Health Environments Research & Design*, 1(3), 61-125.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. Evolutionary Computation, *IEEE* Transactions on, 3(4), 257-271.