6-1-2015

Parametric freeform-based construction site layout optimization

Ibrahim Abotaleb

The American University in Cairo

School of Sciences and Engineering

PARAMETRIC FREEFORM-BASED CONSTRUCTION SITE LAYOUT OPTIMIZATION

A Thesis Submitted to

The Department of Construction and Architectural Engineering

in partial fulfillment of the requirements for
the degree of Master of Science
in Construction Management

By
Ibrahim Salah Eldin Mahmoud Abotaleb

Under the Supervision of

Dr. Khaled Nassar
Associate Professor
Department of Construction and Architectural Engineering
The American University in Cairo

Dr. Ossama Hosny
Professor
Department of Construction and Architectural Engineering
The American University in Cairo

April 2015

Acknowledgments

I would like to express my deepest gratitude to my advisors Dr. Khaled Nassar and Dr. Ossama Hosny for their full support, expert guidance, understanding and encouragement throughout my study and research. In addition, I express my appreciation to Dr. Samer Ezeldin, Dr. Mohamed Marzouk, and Dr. Ezz Eldin Yazid for having served in my committee. Their thoughtful questions and comments were valued greatly.

I would also like to thank Dr. Mohamed AbouZeid for his continuing support, encouragement, and mentoring during my undergraduate and graduate years at the American University in Cairo. Moreover, I would like to express my gratitude to Dr. Mohamed Abdelghany and Eng. Yasser Al-Daidamony for their patience and encouragement on the course of my graduate studies.

Thanks also go to Eng. Islam Mashaly and Eng. Sherif Tarabishy for helping me throughout this academic exploration.

Finally I would like to thank my parents, sister, family and friends for their unconditional love and support that provided me with strength to finish and present this work.

Abstract

Traditional approaches to the construction site layout problem have been focused mainly on rectilinear facilities where the importance proximity measures are mainly based on Cartesian distances between the centroids of the facilities. This is a fair abstraction of the problem; however it ignores the fact that many facilities on construction sites assume non-rectilinear shapes that allow for better compaction within tight sites. The main focus of this research is to develop a new approach of modeling site facilities to surpass limitations and inefficiencies of previous models and to ensure a more realistic approach to construction site layout problems. A construction site layout optimization model was developed that can suit both static and dynamic site layouts. The developed model is capable of modeling any rectilinear and non-rectilinear site shapes, especially splines, since it utilizes a parametric modeling software. The model also has the ability to mimic the "dynamic" behavior of the objects' shapes through the introduction and development of three different algorithms for dynamic shapes; where the geometrical shapes representing site facilities automatically modify their geometrical forms to fit in strict areas on site. Moreover, the model provides different proximity measures and distance measurement techniques rather than the normal centroidal Cartesian distances used in most models. The new proximity measures take into consideration actual movement between the facilities including any passageways or access roads on site. Furthermore, the concept of selective zoning was introduced and a corresponding algorithm was provided; where the concept significantly enhances optimization efficiency by minimizing the number of solutions through selection of pre-determined movement zones on site. Soft constraints for buffer zones around the site facilities were developed as well. The site layout modeling was formulated on commercial parametric modeling tools (Rhino® and Grasshopper®) and the optimization was performed through genetic algorithms. After each of the algorithms was verified and validated, a case study of a real dynamic site layout planning problem was made to validate the comprehensive model combining all of the modules together. Different proximity measures and distance measurement techniques were considered, along with different static and dynamic geometrical shapes for the temporary facilities. The model produced valid near-optimum solutions, a comparison was then made between the layout that is produced with the model and the layout that would have been produced by other models to demonstrate the capabilities and advantages of the produced model.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1 Chapter 1: Introduction

1.1 Site Layout Planning

Site layout planning involves identifying, sizing, and locating necessary temporary facilities on a construction site. Temporary facilities range from simple lay-down areas to warehouses, fabrication shops, maintenance shops, batch plants, and residence facilities (Yeh, 1995; Hegazy & Elbeltagi, 1999). The space available at the construction site is considered an important resource that should be carefully dealt with. In highly congested sites, space becomes a very scarce resource that needs to be carefully planned and efficiently utilized. On the other hand, in large sites having abundant space availability, the positioning of site facilities with respect to each other will greatly influence the efficiency of workflow (Osman et. al., 2003). Despite its importance, site planning is often neglected, and the attitude of engineers has been that it will be done as the project progresses. A well-planned site layout has an significant positive impact on the construction process as it participates in minimizing travel time, promoting safety, enhancing material and equipment transportation and handling, enhancing productivity, and decreasing costs, especially for large projects (Hamiani & Popescu, 1988; Tommelein et al., 1992).

According to Ning et al (2011), the following are the key attributes for successful site layout plans:

1. Efficient movement of materials
2. Efficient tie-in with external transportation
3. Good space utilization and configuration
4. Ease of expansion
5. Satisfaction of safety regulations
6. Effective movement of personnel
7. Efficient operations
8. Low frequency and seriousness of potential breakdowns
9. Security
10. Easy supervision and control

1.1.1 Temporary facilities in site layouts:

The type and number of temporary facilities needed for a specific project must be determined prior to their sizing and location. Table 1 provides a sample list of common temporary facilities that can be used in a project. Every site requires its own unique set of temporary facilities depending on its needs.

Table 1: Sample of possible temporary facilities in site layouts (Elbeltagi, 2014)

No.	Temporary Facilities	No.	Temporary Facilities
1	Job office	20	Fabricated rebar storage yard
2	Owner representatives office	21	Storage yard for lumber
3	Subcontractors office	22	Storage yard for formed lumber
4	First aid office	23	Batch-plant and aggregate storage
5	Information and guard house	24	Craft change-house
6	Electric supply room	25	Machine room
7	Staff/Engineer dormitory	26	Sampling / Testing lab
8	Staff/Engineer family dormitory	27	Pipe jointing yard
9	Labor dormitory	28	Pipe storage yard
10	Labor family dormitory	29	Welding shop
11	Dinning room for labor	30	Parking lot
12	Bathroom for labor	31	Tanks
13	Restroom for labor	32	Long term laydown storage
14	Equipment maintenance shop	33	Electrical shop
15	Parking lot for mechanics	34	Steel fabrication shop
16	Prefabricated rebar storage yard	35	Sandblast shop
17	Rebar fabrication yard	36	Painting shop
18	Carpentry shop	37	Scaffold storage yard
19	Cement warehouse	38	Other material warehouse

Nedzémlyi and Vattai (2014) provide some good guidelines for shaping and sizing the temporary facilities in site layouts. Three examples of their guidelines are provided in this section.

Example 1 - Wood Yard: Although new formwork systems are constantly developed, many construction sites still use traditional formwork systems that require wood yards. A wood yard consists of a cutting yard for cutting the wood pieces, a joining yard for joining the wood pieces, storage yard for new materials and recycled materials, and an assembly yard for assembling and storing the finished pieces. A sample layout of a wood yard is provided in Figure 1.

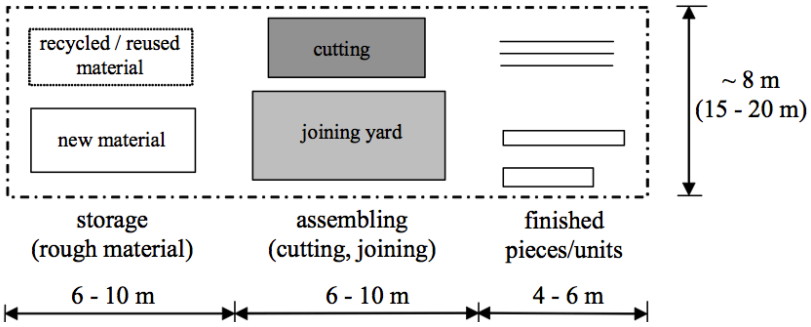

Figure 1: Sample configuration of a wood yard (Nedzémlyi and Vattai, Lecture)

Example 2 - Steel Yard: A steel yard consists of a storage yard for storing the straight rods and the rolls of wire that are shipped to the site, a yard for cutting, aligning, and cutting the steel, and another storage yard for storing the processed steel that is ready for installation (sized straight bars and bent bars ….etc). A sample layout of a steel yard is provided in Figure 2.

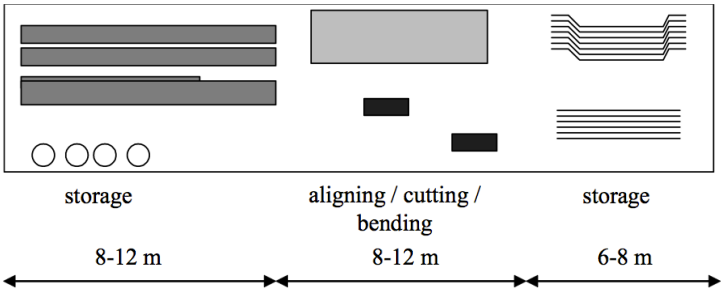

Figure 2: Sample configuration of a steel yard (Nedzémlyi and Vattai, 2014)

Note: All previous research efforts in site layout modeling assume a fixed configuration of the wood yard and the steel yard with a fixed length and width. The purpose of providing these examples is to highlight the fact that the wood yard and the steel yard are not just one compact unit each; but rather a combination of units that can be laid next to each other in different configurations. This research takes this fact into consideration in the modeling process; allowing for the wood and steel yards to have dynamic forms [See Dynamic Rectangles Algorithm, Section 3.2.4] that shape themselves depending on the site conditions; instead of just assuming fixed preset forms.

Example 3 - Caravans: The caravans are essential in all site layouts. They are used for housing the staff's offices (project managers, consultants, contractors, sub-contractors), meeting rooms, dining rooms, cabinets, changing rooms, bathing rooms, labor housing …etc. The caravans are containers that are pre-fabricated. They are transported to the site and installed there using

mobile cranes. The containers come in standard sizes. The single container is 2.44m x 6.06m in size. The double-sized container is 4.94m x 6.05m in size (Nedzémlyi and Vattai, accessed in 2014). The containers can be laid next to each other in any arrangement. Previous efforts in site layout optimization model caravans as rectangles (in the 2-D level) with fixed dimensions, ignoring the many possible arrangements that the caravans can be arranged. This research takes into consideration a variety of alignments for the caravans; where it provides an algorithm [See Offsetted Planar Curves Algorithm, Section 3.2.2] for allowing the caravans to shape their alignment depending on the available shape of the space.

1.1.2 Common problems marking poor site layout planning:

In the absence of a precise site layout plan, the following problems may occur: a) Incorrect allocation of material stacks. This problem may cause double or triple handling of materials to the correct locations. For example: stacking materials too remote from the hoist or not within the radius of the crane; or stacking the materials on areas that would be excavated later; b) Inappropriate allocation of plant and equipment; such as locating the concrete mixer in a place that is inaccessible for the delivery of materials, locating the tower cranes in locations where they would not be able to reach all parts of the works; c) Inadequate allocation of space; causing safety hazards and larger travel time due to cramping.; and d) Wrongful allocation of site huts in relation to their effective use; such as locating the site offices too near from noisy activities or too remote with insufficient overview of the site, or setting the warehouses in an area with inadequate access for loading and unloading or located in insecure area.

1.1.3 Types of construction sites:

Construction sites can be categorized into 4 categories demonstrated in Figure 3: 1) open field, 2) local estate, 3) long and thin, and 4) restricted (Nassar, 2014). In an open field construction site, the available site space is abundant and the works are concentrated in a localized zone that is relatively small; thus providing many options for allocating the temporary facilities without many restrictions. In the local estate construction site, the construction zones where the works are undergone are scattered along the site, yet small in size. So although the available area for temporary site facilities is large, there are many constraints regarding the transportation and safety due to the small pathways and passages between the construction zones. This type of site

layout is very frequent in residential projects. The long and thin construction site is mainly the type of layout for roads. It has special considerations in the site layout planning which are beyond the scope of this research. In the restricted site layouts, the construction zone is taking the bulk area of the project thus leaving a very limited area for setting the temporary facilities. In this type of layout, it is important to model the site facilities accurately with minimizing the interpolations and the approximations to not waste area. The model developed in this research is highly recommended for restricted sites.

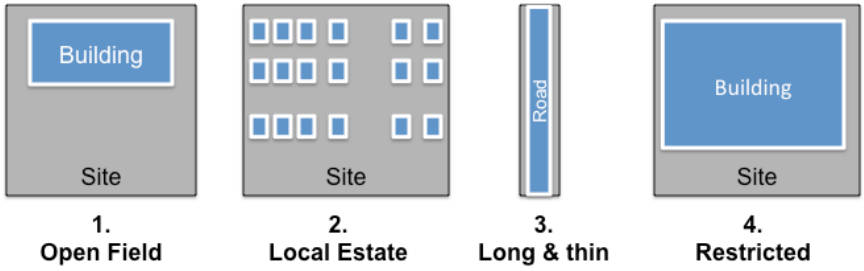

Figure 3: Types of construction sites (Nassar, 2014)

1.1.4 Static site layouts vs. Dynamic site layouts:

In simple words, a static site layout is where the location of the temporary facilities is not changed throughout the duration of the project; once a temporary facility is set in a place, it stays in that place till the project ends or until dismantled earlier, but it does not move from a place to another place. In dynamic site layouts, as shown in Figure 4, the temporary site facilities change their locations at different stages of the projects, depending on the site conditions and the execution plan. Optimizing dynamic layouts takes into consideration additional criteria such as relocation costs. The algorithms provided in this research suit both static and dynamic site layouts.

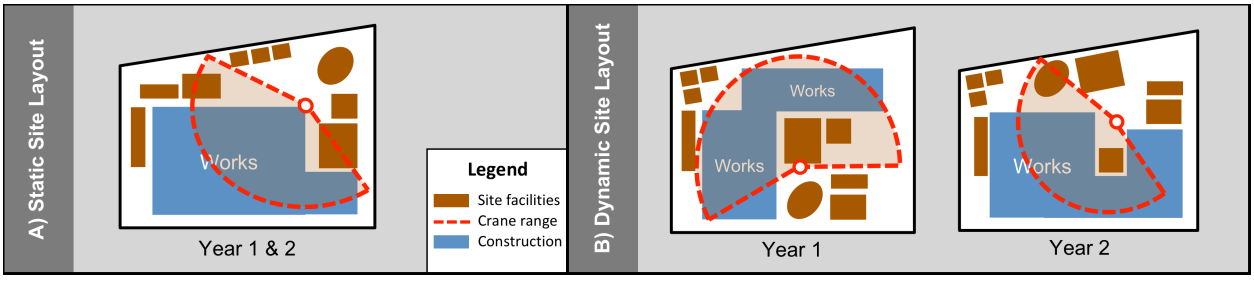

Figure 4: Static and dynamic site layouts

1.2 Previous Research in Site Layout Optimization

The problem of site layout planning has been studied by researchers using mainly two techniques; heuristic techniques and mathematical optimization models to produce the optimum solutions. Heuristic methods produce good but not optimal solutions, and cannot be fairly be adopted for large complex projects. Mathematical techniques are capable of solving more complex site layout planning problems with different types of variables and constraints (Osman et al., 2003). Mathematical techniques involve the identification of one or more goals that the site layout strives to achieve, the modeling and mathematical term for the end goal is the "objective function". The goal is to optimize the objective function by either maximizing or minimizing it through the alterations of the different model variables. Several mathematical techniques and models were developed to solve site layout problems. Most famous of them are linear programming (Zouein & Tommelien, 1999), genetic algorithms (Li & Love, 1998; Hegazy & Elbeltagi, 1999; Zouein et al., 2002; Osman et al., 2003; • Zhou et al., 2009; Khalafallah and El-Rayes, 2011; Nguyen, 2014), artificial neural networks (Yeh, 1995), artificial bee colony (Yahya & Saka, 2014), artificial ant colony (Ning et al., 2011), fuzzy logic (Tam et al., 2001; Elbeltagi & Hegazy, 2001; Xu & Li, 2012), simulated annealing (Andayesh & Sadeghpour, 2014), and other hybrid optimization techniques (El-Rayes & Said, 2009; Ning et al., 2011; Lien and Cheng, 2012); Andayesh & Sadeghpour, 2013). (See Figure 5)

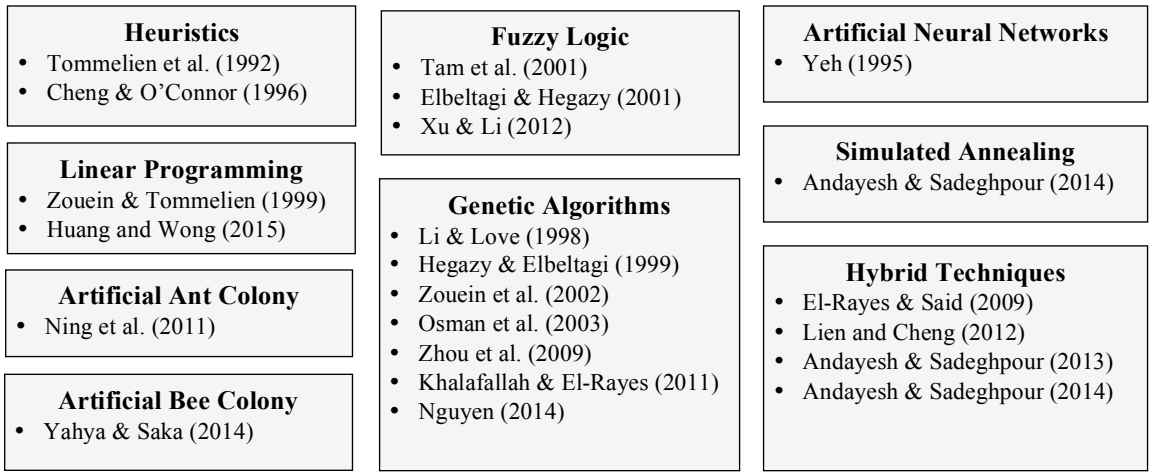

Figure 5: Techniques of tackling site layout planning problems

Research in the field of site layout planning is targeted on either developing the modeling techniques for the site facilities and obstacles, finding better optimization techniques to find optimum solutions, investigating better proximity measures, examining and comparing different objective functions, or investigating the effect of time and scheduling on the different site layout variables. This research is oriented towards ***finding better algorithms for modeling the site layout facilities and obstacles*** since the research in this area has not been widely developed as shown in Figure 6. An accurate representation of the construction site is important for site layout modeling, as it enables the development of more realistic and efficient layouts. Li & Love (1998) modeled the site facilities in predetermined locations; where the number of predetermined places should be equal to or greater than the number of predetermined facilities. So there is no flexibility in moving the site facilities, except in the pre-determined locations. The variable of each facility would just be its location number. Hegazy & Elbeltagi (1999) presented a more flexible modeling for the site facilities by using a two-dimensional grid, where each facility is modeled as a number of grid units that add up to the facility area. Osman et al. (2003) provided a CAD-based model for site facilities with almost the same concept as Hegazy & Elbeltagi's model (1999) by modeling the dividing the site into perpendicular grids and enclosing the facilities inside the grids. Andayesh & Sadeghpour (2013) represented site facilities by their minimum bounding circles to facilitate the optimization process. However, this representation also results in a waste of space. Yahya & Saka (2014) presented a model that modeled facilities as rectangular geometries that allow for horizontal and vertical alignment with site boundaries that are represented as lines with specified slopes.

The use of grids, circles, and rectangles as shown in Figure 6 simplifies the search procedure by decreasing the number of possible choices for the position of objects. However, in reality, the construction site and facilities can acquire any shape and can be located in any place. In strict construction sites, it can be safely claimed that the previously mentioned models face difficulties, that sometimes result in not finding valid results at all, in finding solutions in strict site layouts due to the inefficiency of modeling the facilities through wasted area difference between the "actual" and the "modeled" shapes of the facilities. Moreover, all previous models assumed fixed "static" shapes for the site facilities; while in reality, shapes of site facilities are "dynamic" in the sense that their geometries change in every run enabling them to be squeezed into the available tight spaces. For example, a pile of sand might acquire a circular or an elliptical or even a spline

geometrical plan depending on the available space; this flexibility of selecting what shape to acquire is not available in current models.

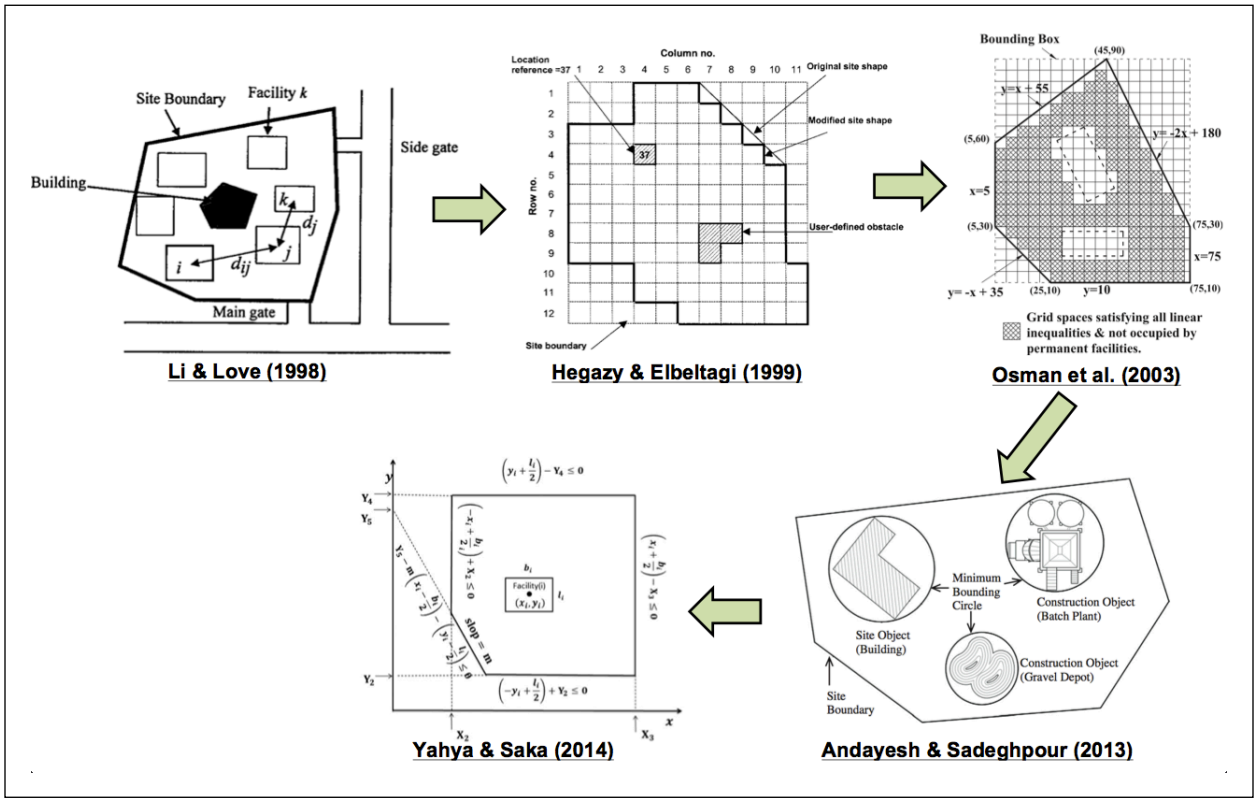

Figure 6: Development of modeling of site facilities throughout the years

To further highlight the inefficiency of the currently available site layout optimization model, a small demonstration is made in by modeling a spline shape (such as a swimming pool or a pile of sand) with all 4 available modeling techniques. If the shape was modeled using the orthogonal grids algorithm (such as in Hegazy & Elbeltagi, 1999; Osman et al, 2003) the area gap between the modeled shape and the real shape would be about 37%; which is a significant waste of area. If the shape was modeled using the rectangular algorithm (such as in Yahya & Saka, 2014) the area gap between the modeled shape and the real shape would be about 46%; which is a significant waste of area as well. Although the rotated angle algorithm was not confirmed that it was used before, but even if it was used to model the test subject, the wasted area would be 27% of the model area. If the shape was modeled using the circular algorithm (such as in Andayesh & Sadeghpour, 2013) the area gap between the modeled shape and the real shape would be about 38%; which is still significant waste of area.

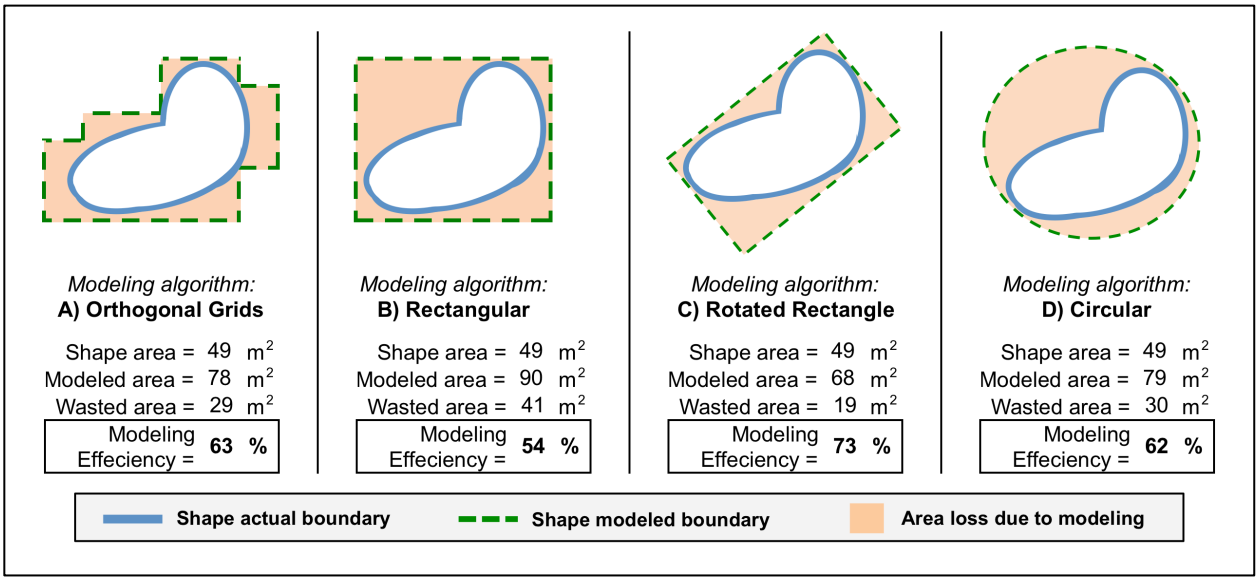

Figure 7: Modeling inefficiency of the currently available techniques for modeling site facilities and obstacles

1.2.1 Problem Statement

The geometrical representations for site facilities in pervious research efforts result in noteworthy inefficiencies due to the rectilinear and simple approximations of complex and non-rectilinear shapes resulting in area loss and misuse of site area; such misuse cannot be afforded in strict construction sites. In strict construction site layouts it is important to be able to model site facilities and obstacles as close as possible to reality to be able to produce valid solutions by the optimization model.

1.3 Research Objectives

The objectives of this research are to:

1. Present new algorithms for accurately modeling regularly and irregularly shaped site facilities, and for mimicking their dynamic behavior to ensure a more realistic approach to construction site layout problems.
2. Provide algorithms for new proximity relationships and distance measurement techniques between the different site facilities.
3. Introduce the concept of selective zoning that significantly minimizes the model running time by limiting the areas of movements for the different facilities to only the valid zones.
4. Formulate a full site layout optimization model that utilizes all of the introduced algorithms in a parametric modeling software.

1.4 Research Methodology

The following points summarize the used methodology in pursuing this research:

1. Study the previous site layout optimization models and investigate their modeling inefficiencies.
2. Specify the problem statement and set the research objectives.
3. Demonstrate the capabilities of using a new parametric modeling platform in modeling complex and irregularly shaped static geometries, thus minimizing the area losses and inefficiencies in modeling; which solves a part of the problem stated in the problem statement.
4. Develop three different algorithms for "dynamic" geometrical shapes and provide instructions on their formulation on the used parametric modeling platform. The purpose of the dynamic geometrical shapes is to enhance the model's ability to find near-optimum solutions in strict sites with irregular spaces.
5. Verify and validate the developed algorithms of the dynamic geometrical shapes.
6. Define the concept of selective zoning and develop an algorithm for it. The algorithm development is to be followed by its verification and validation.
7. Develop an algorithm for collision and overlapping prevention constraint using an area union concept. The algorithm development is then followed by its verification and validation.
8. Develop an algorithm for in-site constraint, which ensures facilities are inside the site boundaries throughout the model runs, using an area union concept. The algorithm development is to be followed by its verification and validation.
9. Develop an algorithm for buffer zones around the facilities as soft constraints. The algorithm development is to be followed by its verification and validation.
10. Present an algorithm for the Shortest Walk distance as an added distance measurement technique and demonstrate its formulation on the used parametric modeling platform. Afterwards, a comparison is to be made between it and the direct Cartesian distance measurement technique.

11. Present four different proximity relationships (Center-to-Center, Point-to-Point, Center-to-Point, and Side-to-Side) and demonstrate their formulation on the used parametric modeling platform
12. State the optimization procedure and the different parameters (variables, objective function, constraints, and optimization technique).
13. Verify the capability of the developed algorithms to be integrated into one model by testing them in a case study of site layout optimization problem.
14. Validate the integrated model by analyzing the logic of the results and comparing them to results of previous optimization models from the literature.

1.5 Thesis Organization

This thesis is organized into five chapters. Chapter 1 provides a general introduction about site layout planning, the software (Grasshopper®) that is used for the model development, and the problem statement ending with the research objectives. Chapter 2 presents a review of the problem solving techniques for the site layout problem and the previous research efforts made in that topic. Chapter 3 discusses in details the model development and its different newly introduced algorithms that serve for the purpose of the research objectives. The model uses a commercial parametric software for developing the algorithms that has never been used for site layout optimization purposes, this chapter provides detailed demonstrations about the different developed algorithms with examples for verification. Chapter 4 presents a case study where the developed algorithms are applied to a real construction project and the results are analyzed to validate the model. Chapter 5 summarizes and concludes the research and provides recommendations for future research in the site layout optimization field. The thesis structure is shown in Figure 8.

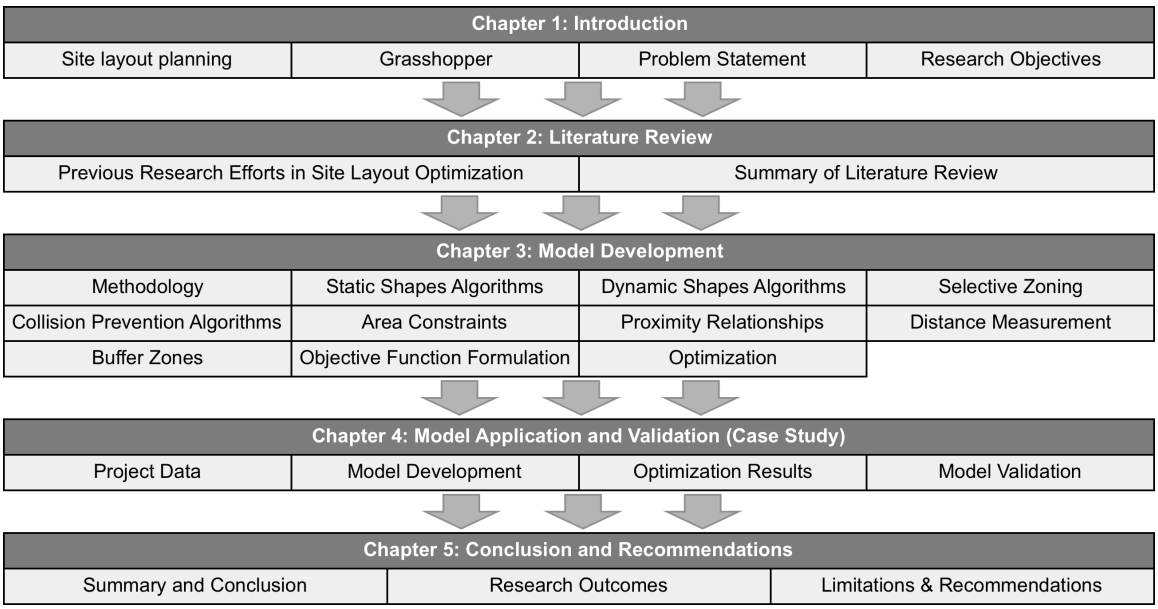

Figure 8: Thesis organization

Chapter 2

Literature Review

2 Chapter 2: Literature Review

2.1 Problem-solving Techniques in Site Layout Planning

Many problem-solving techniques have been utilized in site layout optimization such as mathematical modeling, knowledge based systems, artificial intelligence, and evolutionary algorithms. There is no settlement between researchers on a certain problem solving technique that proves to be more suitable than others, since all of them have different advantages and shortcomings. Site layout optimization techniques can be categorized into two categories: heuristic and mathematical techniques.

2.1.1 Heuristic techniques

Due to weak capabilities of the computing powers at the time, early research in site layout planning depended on heuristic designs. Tommelien et al. (1992) provided an expert system for construction site layout planning named SightPlan. SightPlan models how people place facilities in the construction site and encodes the domain knowledge applied in the process using common lisp. Inputs for SightPlan are: 1) dimensions and locations of permanent site facilities, 2) dimensions and locations of access roads, 3) dimensions of temporary site facilities, 4) constraints on the temporary facilities regarding their location relevant to the locations of the permanent facilities, and 5) zones dividing the construction site area into smaller areas. According to Tommelien et al. (1992), SightPlan is highly case-specific and requires a lot of development to be generic enough to be applied at a wider range of construction sites since it was only tested in industrial site layouts with under-constrained systems.

An automated site layout system named ArcSite was presented by Cheng & O'Connor (1996). The system was for the objective of automating the planning tasks for laying out temporary site facilities with the objective of minimizing construction conflicts and improving efficiency. ArcSite used an integration of a database management system (DBMS) and a geographic information system (GIS) as inputs. ArcSite consists of knowledge specific to construction site layout, temporary facilities databases, Arc/Info databases, and algorithms for integrating and automating temporary facility layout design. The system offers a methodology to systematically acquire and interpret experts' knowledge and experience in site planning, then it uses the concept of searching by elimination to develop a heuristic approach to model the process of human

decision-making and generate the potential locations for the temporary facilities. The procedures developed in ArcSite to develop its knowledge-based heuristic algorithm are classified into three phases as shown in Figure 9: 1) compilation of the experts' knowledge of site layout planning, 2) interpretation of the obtained knowledge, and 3) translation of the knowledge into the system's implementation forms.

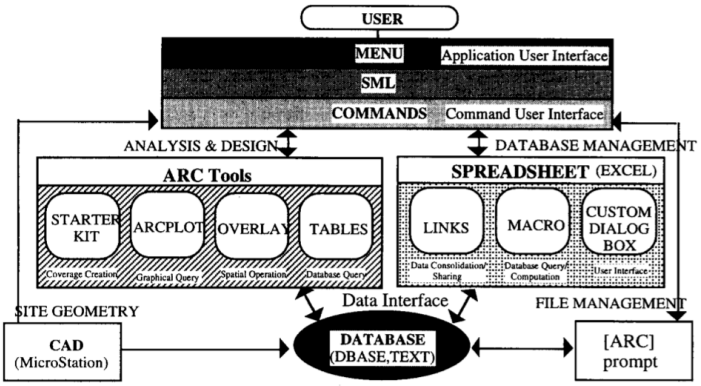

Figure 9: ArcSite system architecture (Cheng & O'Connor, 1996)

According to the authors, one of the disadvantages of ArcSite is that it only consists of up to 25 temporary facilities. Another disadvantage is that ArcSite does not have the ability to draw objects such as polygons, lines, or points on the site.

2.1.2 Mathematical techniques

Mathematical techniques are capable of solving more complex site layout planning problems with different types of variables and constraints. However, as the variables and constraints increase, the computational powers needed to solve the problems increase as well, and sometimes that increase is exponential. Mathematical techniques involve the identification of one or more goals that the site layout strives to achieve. The modeling and mathematical term for the end goal is the "objective function". The goal is to optimize the objective function by either maximizing or minimizing it through the alterations of the different model variables.

Several mathematical techniques and models were developed to solve site layout problems. Most famous of them are linear programming, genetic algorithms, artificial neural networks, ant colony, fuzzy logic, and dynamic programming.

2.1.2.1 Linear Programming:

Linear programming is a type of mathematical modeling with the focus of allocating limited resources to known goals through mathematical formulations to meet a certain objective such as minimizing losses or maximizing efficiency. One of the most distinct characteristics of linear programming is that the object function and the constraint functions are linear in nature. Linear programming is simple in its nature and is used in many engineering applications; especially in problems with few and uncomplicated constraints. There is not much research allocated in utilizing linear programming in site layout planning problems due to the complexity and the non-linear behavior of such problems.

Zouein & Tommelien (1999) presented a site layout model employing linear programming in solving their dynamic site layout. The objective function is to minimize (Inter-facility Transportation Cost + Facility Relocation Costs). The model works as combining heuristic rules with linear programming functions. A resource would be selected heuristically one at a time, then the model would calculate sets of valid positions that satisfies all constraints. Consequently, a linear equation is solved to find the optimal location of each resource in the pursuit of minimizing the objective function. The model performs the assignment process with time frames in chronological order. So, the assignment of a resource is dependent on the assignment of the preceding resource; thus the presented algorithm does not provide optimum solutions, but rather feasible solutions. The model also is purely a mathematical approach, with very limited graphical modeling capabilities, and according to the author, it can fail to detect overlaps between geometrical entities. The resulting sequence of layouts, if one is found, is suboptimal in terms of the stated objective. This is nonetheless a desirable outcome when solving a problem for which no closed-form mathematical solution exists.

2.1.2.2 Genetic Algorithms:

Genetic algorithms (GA) belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Computer scientists originally studied evolutionary in the 1950s and 1960s as a way of reaching optimal solutions for engineering using the concepts of evolution. Genetic Algorithms were conceived by John Holland in the 1960s and

were developed by Holland and his students and colleagues at the University of Michigan in the 1960s and the 1970s (Mitchell, 1996).

As defined by Mitchell (1996) "*GA is a method for moving from one population of "chromosomes" (e.g., strings of ones and zeros, or "bits") to a new population by using a kind of "natural selection" together with the genetics−inspired operators of crossover, mutation, and inversion. Each chromosome (solution) consists of "genes" (e.g., bits), each gene being an instance of a particular "allele" (e.g., 0 or 1). The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones. Crossover exchanges subparts of two chromosomes, roughly mimicking biological recombination between two single−chromosome ("haploid") organisms; mutation randomly changes the allele values of some locations in the chromosome; and inversion reverses the order of a contiguous section of the chromosome, thus rearranging the order in which genes are arrayed*"

GA engages 3 types of operators: selection, crossover, and mutation. 1) Selection is where the chromosomes in the population are selected for reproduction (breeding). The fitter chromosomes are more likely to reproduce. 2) Crossover is the mating of two chromosomes where some genes from a fit chromosome are taken and combined with other genes from another fit chromosome to produce two offspring that are a mix of the genes of the parent fit chromosomes. The crossover operator roughly mimics biological recombination between two single−chromosome (haploid) organisms. 3) Mutation is where some genes are mutated in the breeding by randomly flipping some of the bits in a chromosome. Mutation can occur at each bit position in a string with a very small probability that is controlled by the user. Each iteration is called a generation. The entire set of generations is called a run. The methodology of how GA operates is provided in the flowchart in Figure 10.

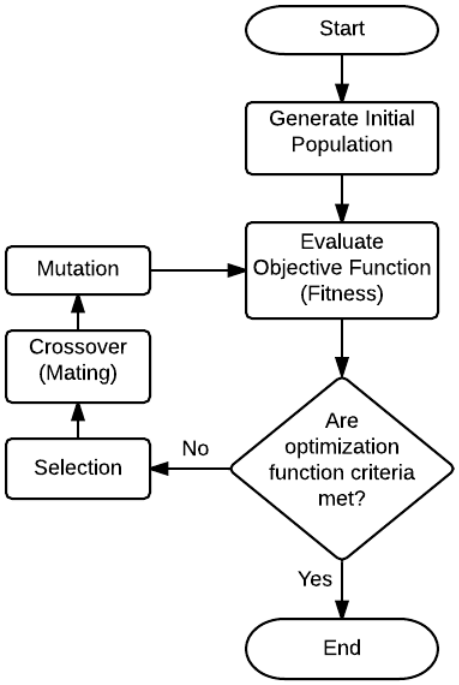

Figure 10: Procedures of Genetic Algorithms

Advantages and disadvantages of evolutionary algorithms: Evolutionary algorithms are slow when it comes to solving an optimization problem with several millions of valid solutions. This is not the shortcoming of the algorithm itself, but rather the limitation of the computing power available at the time of running. Nowadays, problems become more and more complex and the number of variables becomes excessive, thus requiring an immense computational and processing powers. Another drawback of evolutionary algorithms is that if the mutation rate is not high enough, the evolutionary solver might not reach a global optimum and reaches a local optimum instead. On the other hand, if the mutation rate is very high, it would take the solver a much longer time to reach an optimum solution. Moreover, evolutionary algorithms do not guarantee an "optimum" solution; especially in cases of excessive variables where it would need infinite time to try all combinations and obtain the "optimum" solution. Instead, they obtain "near-optimum" solutions much more quickly. So, the user must know that the obtained solution is most probably the "near-optimum" solution in large problems. In order to minimize the running time of evolutionary algorithms, it is preferred to feed the first population with a good valid solution with high fitness if possible. This makes the breeding more likely to produce better solutions quickly.

On the good side, there are many benefits for evolutionary algorithms that make them unique amongst other computational methods. Evolutionary algorithms are remarkably flexible and are able to tackle a wide variety of problems. Evolutionary algorithms can tackle problems that are under-constrained or over-constrained or otherwise even poorly formulated. Furthermore, because the run-time process is progressive, intermediate answers can be harvested at practically any time. Unlike many dedicated algorithms, Evolutionary solvers spew forth a never-ending stream of answers, where newer answers are generally of a higher quality than older answers. So even a pre-maturely aborted run will yield something which could be called a result. It might not be a very good result, but it will be a result of sorts. According to Rutten (2010), "*Evolutionary Solvers allow -in principle- for a high degree of interaction with the user. This too is a fairly unique feature, especially given the broad range of possible applications. The run-time process is highly transparent and browsable, and there exists a lot of opportunity for a dialogue between algorithm and human. The solver can be coached across barriers with the aid of human intelligence, or it can be goaded into exploring sub-optimal branches and superficially dead-ends*." According to Fonseca and Fleming (1998), evolutionary algorithms became favorable over other multi-objective mathematical approaches because of its population-based search and efficiency in discontinuous and non-differential problems.

GA have been applied to solving the facility layout problem in the area of production facilities (Tanaka and Yoshimoto 1993; Tate and Smith 1993, 1995; Hamamoto et al. 1999) and to solving the ‘‘construction site-level facility layout’’ (Li and Love, 1998). In both applications, the layout problem was modeled as a location-allocation problem, which consists of allocating a set of predetermined facilities into a set of predetermined sites where the smallest site can accommodate the largest facility.

One of the early researches to tackle the site layout optimization using GA was made by Li and Love (1998), where they presented an investigation of applying the genetic algorithm (GA) system to search for the optimal solution for a construction site-level layout problem. Their model was tested on an example of a site with 11 facilities with the objective function of minimizing the total traveling distance TD of site personnel between facilities according to Equation 1 (Li and Love, 1998):

$$TD = \sum_{i=1}^{n} \sum_{x=1}^{n} \sum_{j=1}^{n} \delta_{xi} f_{xi} d_{ij} \quad \text{..... [Eqn. 1]}$$

Where,

N: Number of facilities
δ_{ij}: permutation matrix variable
f_{ij}: frequency of trips by personnel between facilities i and j
d_{ij}: Distance between locations m and n

One limitation in their model is that in the problem description, locations have to be predetermined as shown in Figure 11. The number of predetermined places should be equal to or greater than the number of predetermined facilities. So there is no flexibility in moving the site facilities, except in the pre-determined locations. The model does not take into consideration geometrical features and deals with the facilities as objects that can be put in any of the pre-determined locations regardless of their sizes and geometrical shapes.

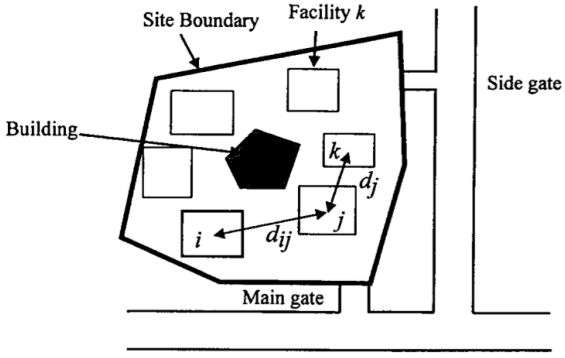

Figure 11: Predetermined locations for site facilities (Li and Love, 1998)

Hegazy and Elbeltagi (1999) presented an evolution-based site layout planning model (EvoSite) that utilized GA to optimally place facilities within a construction site. Their model was more generic, comprehensive, and flexible than the one presented by Li and Love in 1998. EvoSite models any irregular user-defined site using a two-dimensional grid, where each facility is modeled as a number of grid units that add up to the facility area. Such representation is made on spreadsheet modeling, where each grid unit is represented on a cell. A facility is placed on the grid as a number of units (cells) as shown in Figure 12. EvoSite provides three alternative methods for placing a facility on the site grid, starting from the facility's location reference: (1) horizontal; (2) vertical; and (3) rectangular.

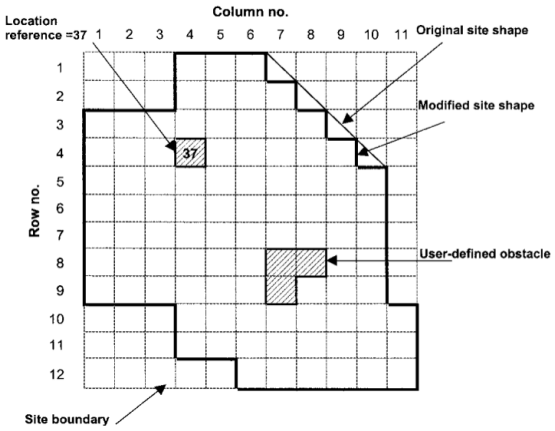

Figure 12: Site representation of the EvoSite model (Hegazy and Elbeltagi, 1999)

The basis of specifying the closeness relationships representing how should close should the facilities be place in relationship to each other are not highly discussed EvoSite model, but rather left generic to the user's preferences. However, for the case study in the model, an exponential relationship with desired closeness were used. The objective function of the EvoSite model is to minimize the travel distance according to Equation 2 (Hegazy and Elbeltagi, 1999).

$$\text{(objective function)} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} R_{ij} \quad \text{. . . . . [Eqn. 2]}$$

Where,

n: Number of facilities
R_{ij}: Desired proximity weight value between facilities i and j
d_{ij}: Distance between locations m and n

Despite the flexibility provided by EvoSite model at the time of its presentation, it has some shortcomings such as the long time of calculations and the hardships of laying large site facilities after laying the smaller ones in a scattered manner; leading to adding some heuristic rules of laying the large facilities first before running the model to place the small facilities; which takes away its advantage of being a generic model. Another shortcoming is that EvoSite deals only with geometrical shapes formed of square grids, so modeling a spline or another complex geometrical shape would be inefficient since a lot of area would be wasted while interpolating the edges to be incorporated in square grids.

Zouein et al. (2002) presented a model applying genetic algorithms for solving the site layout problem, as characterized by rectangular facilities with proximity requirements between them

and with geometric constraints that further restrict their relative positions. The objective function is to minimize the transportation cost of materials. Object positions around the site are the variables. The GA operators are programmed so that to maximize the probability of locating valid position for the objects. The used GA were highly problem oriented, and the research focused more on the investigation of the modified GA than on developing an integrated site layout planning system. While testing the model, the authors commented that in most cases where the total- objects-to-site-area ratio did not exceed 60%, the algorithm returned close to optimal solutions in a reasonable time. However, in problems with higher total-objects- to-site-area ratio the algorithm failed to find ‘‘good’’ and in some cases feasible solutions. Another drawback of the model is that it only models rectangular facilities in rectangular site boundaries with no other varieties. It also does not provide different choices for proximity measures.

Osman et al. (2003) presented a CAD-based site layout optimization model that consisted of an genetic algorithm optimization engine and a geometric input / output interface. The link between the optimization engine and the geometrical data contained in AutoCAD™ drawings was made through AutoCAD™ VisualBasic™ Applications. All site-related geometrical data were programmed to be detected as an orthogonal 2-D grid. The CAD-based GA approach was extended to include the changes that take place in the construction site throughout the project lifespan, thus also enabling solving for dynamic site layouts. The model was tested on an actual 24,000 m^2 construction site. The model produced a site layout that accomplished nearly a 25% saving in total layout cost compared to the layout actually adopted. The dynamic site layout optimization algorithm was based on the Mini-Min approach. The Mini-Min approach considers all possibilities for choosing the critical phase. It performs the dynamic optimization of all phases N_{phase} times, N_{phase} being the number of phases. It calculates the total costs for all phases N_{phase} times and chooses the trial having the least cost as the Minimum-Minimum solution. The presented model provided a good example of the benefit of the interaction between visual software and programming in solving site layout problems, however, the model depended on perpendicular grids to model the layout boundaries and facilities as shown in Figure 13, thus posing inefficiencies in the use of site space causing the resulting of no valid solutions in heavily crowded sites.

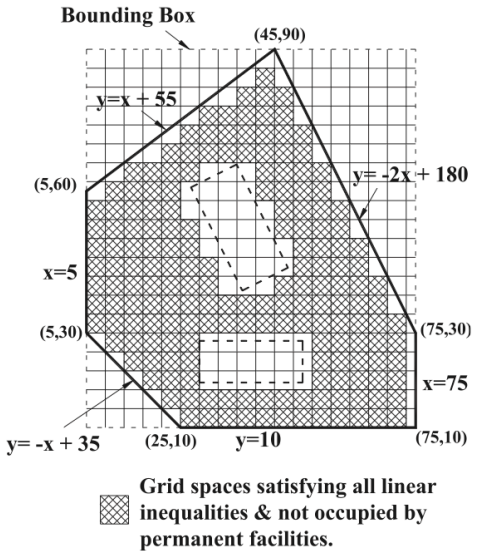

Figure 13: Site representation of the model by Osman et al. (2003)

2.1.2.3 Artificial Bee Colony Algorithm

The artificial bee colony (ABC) algorithm is one of the most recent swarm intelligence based algorithms introduced by Dervis Karaboga in 2005. The artificial bee colony algorithm mimics the foraging behavior of honey bees. Yahya and Saka (2014) proposed a multi-objective artificial bee colony (MOABC) via Levy flights algorithm to determine the optimum construction site layout. The model is intended to optimize the dynamic layout of unequal-area under two objective functions. The first objective function is to minimize the total handling cost of interaction flows between facilities as shown in Equation 3 (Yahya and Saka, 2014). Whereas the second objective function is to minimize safety hazards/environmental concerns facilities as shown in Equations 4 and 5 (Yahya and Saka, 2014).

$$f_1 = \text{Min}\sum_{p=1}^{np}\sum_{i=1}^{n}\sum_{j=1}^{n} C_{ijp} r_{ij} P_{ijp} \quad \text{..... [Eqn. 3]}$$

$$f_2 = \text{Min}\sum_{p=1}^{np}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{-1}{SE_{ijp}} e_{ij} P_{ijp} \quad \text{..... [Eqn. 4]}$$

$$e_{\text{ij}} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \quad \text{..... [Eqn. 5]}$$

Where,

(x_i,y_i) & (x_j,y_j): Cartesian coordinates of the centroids of facility i and j; which are treated as design variables.
e_{ij} : Euclidean distance between centroids of facilities i and j.
r_{ij} : Modified rectangular distance between centroids of facilities i and j that considers the presence of obstruction.
SE_{ijp} : Closeness relationship values for safety & environmental concerns.

One of the advantages of the model is that it takes into consideration the obstacles while calculating distances between facilities, unlike other previous models. Figure 14 demonstrates the two different distance measurement techniques in Yahya and Saka's model (2014); the Cartesian distance and the modified rectangular distance. Yet, the paper did not describe the exact algorithm of calculating the modified rectangular distance. The significance of the MOABC model rises from the fact that it is the first site layout problem to be modeled using the artificial bee colony algorithm, which is the optimization technique, however, it does not introduce new advances in the core essence of site layout planning itself. The facilities are still modeled as rectangular shapes and the model's efficiency was not tested on strict construction sites; but it is expected to fail to obtain feasible results in heavily strict construction sites due to the Cartesian linear modeling of the facilities, as all other site layout optimization models.

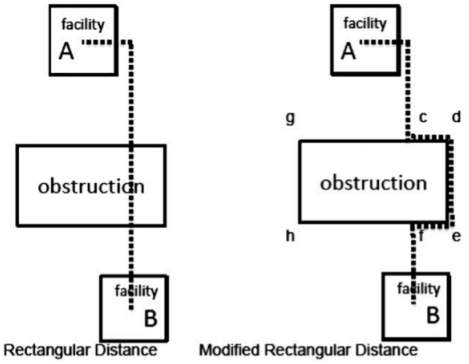

Figure 14: Traveling distance measurement in Yahya and Saka's model (2014)

2.1.2.4 Artificial Neural Networks:

Yeh (1995) presented a discrete combinatorial optimization problem using the Annealed Neural Networks for construction site layout. His model used the method of assigning the facilities at predetermined locations whilst satisfying a set of constraints. His model combined the algorithm of Hopfield neural networks, which are suitable for solving discrete combinatorial optimization problems, and simulated annealing. The main drawback of using Hopfield's neural network is its limitation in escaping local minima. That is why the simulated annealing was used in parallel to it. Simulated annealing is known to finding global minimum by combining gradient decent with a random process; however it requires high computational powers.

2.1.2.5 Fuzzy Logic:

Fuzzy logic was first introduced by Zadeh (1965), who defined it by: ‘‘*A fuzzy set is a class of objects with a continuum of grades of member- ship. Such a set is characterized by a membership (characteristic) function which assigns to each object a grade of membership ranging between zero and one*.’’

Tam et al. (2002) tackled the construction site layout problem from an evaluative angle rather than a problem solving angle by presenting a nonstructural fuzzy decision support system NSFDSS that integrates both experts’ judgment and computer decision modeling, making it suitable for the appraisal of complicated construction problems. The NSFDSS follows three steps: 1) decomposition; for wich the problem is structured into elements of different elements of different levels working downward from the goal on the top through criteria bearing to the goal on the second level and then to subcriteria on the third level, and so on, 2) comparative judgment, where constructed pairwise comparisons are made of the relative importance of the elements with respect to the shared criterion or property on the level above, giving rise to the corresponding matrix and 3) synthesis of priorities, by multiplying local priorities with the priority of their corresponding criterion on the level above, and weighting each element on a level according to the criteria it affects. The NSFDSS is not an optimization model that tries to find optimum solutions for site layout problems, but rather an evaluation tool for site layouts made by project managers. So, there was no emphasis on finding new techniques of modeling site facilities or formulating site layout problems.

Xu and Li (2012) formulated a fuzzy random multi-objective decision-making model for dynamic site layout optimization. The model utilized the following two objective functions: 1) minimizing the total cost of site layout; and 2) maximizing the distance between the ‘high-risk’ facilities and the ‘high-protection’ facilities to reduce the possibility of safety or environmental accidents. In the model, The interaction cost and the operating cost of facilities are regarded as fuzzy random variables. The fuzzy random uncertainty in the mathematical modeling of construction site layout planning is considered. The model is not focused on modeling the geometries of the site facilities and their geometrical constraints. Moreover, the model assumes predetermined locations for the objects; where objects are only allocated in these predetermined locations; which is not realistic.

2.1.2.6 Ant Colony:

Multi-objective optimization, also referred to as Pareto-based ant colony optimization, is where there is no one single objective function, yet there are multiples of them, and they might be conflicting. So, there is no one optimum solution, but rather a stream of good solutions with trade-offs between two ore more objective functions where the user has the final decision to select the suitable solution from the stream of good solutions. A good example of multi-objective optimization problem is the decision to buy a car with an objective function to minimize cost and at the same with another objective to maximize comfort. Usually, static site layout problems involve only one objective function; on the other hand, dynamic site layout problems involve multi-objective optimization due to their different layout stages.

Ning et al. (2011) proposed a decision-making system to solve dynamic and unequal-area, multi-objective optimization construction site layout problems. The system has two objective functions: 1) minimizing the likelihood of accidents happened to improve the safety level (f1), and 2) minimizing the total handling cost of interaction flows between the facilities associated with the construction site layout (f2), which is based on interaction relationship between the facilities. Two optimization models, one is single-objective-function model and the other is multi-objective-function model, are employed to solve multiple objective site layout planning problems using the methodology shown in Figure 15.

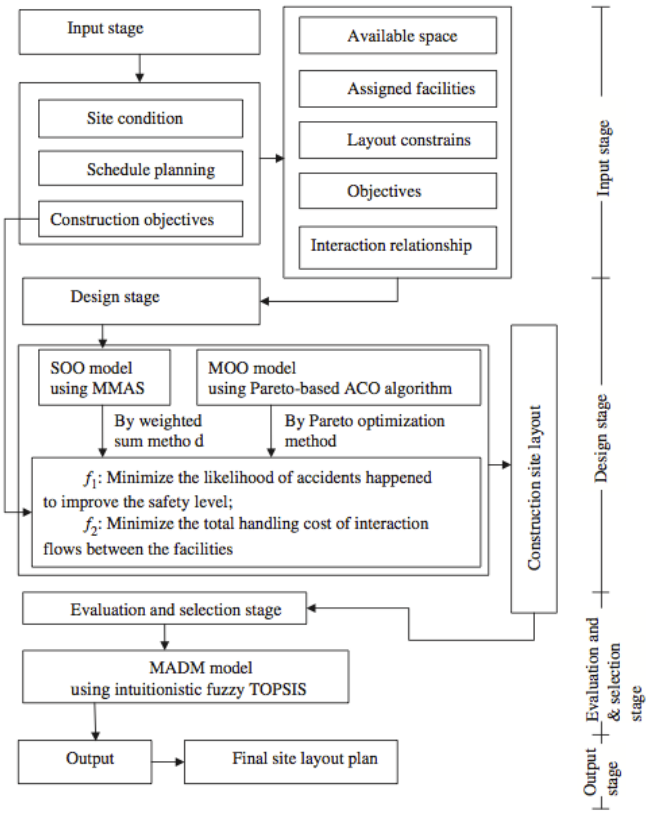

Figure 15: Methodology of the decision-making system presented by Ning et al. (2011)

2.2 Summary of Literature Review

2.2.1 Proximity measures

The proximity relationships represent the preference of the site planner in locating the facilities near or far from one another. There are quantitative and qualitative methods for specifying these proximities. The quantitative methods consider the actual transportation time and cost of personnel and materials between the facilities. The qualitative method consider subjective numerical proximity weight to represent the desired inter-facility proximity (close or far), or, in other words, closeness relationships. The developed model suits both methods as it generic. The user has the flexibility to input the proximity weights that he desires. For demonstration purposes in this research, the qualitative method is used. The method used in this research is the qualitative method. If the two facilities are required to be close to each other, the proximity weight between them would have a high value. As stated in the literature review, several scales have been adopted to represent the proximity weights representing the qualitative importance of closeness of objects in numerical values. presents a summary of the used qualitative closeness matrices used in the different research efforts mentioned in the literature. The same table shows in the last column the proximity weights used in the current research, where a negative value represents the "undesirable" proximity between facilities; which was selected to be the negative of the value representing the "important" proximity.

Table 2: Different proximity measures used in previous research

Desired relationship between facilities	Proximity Weight [1]	Proximity Weight [2]	Proximity Weight [3]	Proximity Weight [4]	Proximity Weight [5]
[A] Absolutely Necessary	6^5=7,776	81	7500	81	***81***
[E] Especially Important	6^4=1,296	37	1500	27	***37***
[I] Important	6^3=216	9	250	9	***9***
[O] Ordinary Closeness	6^2=36	3	50	3	***3***
[U] Unimportant	6^1=6	1	10	1	***1***
[X] Undesirable	6^0=1	0	1	0	***-9***

[1] Hegazy & Elbeltagi (1999), [2] Askin et al. (1993) and Osman et al. (2003), [3] Elbeltagi (Lecture), [4] Yahya & Saka (2014), [5] Current research (2015)

Since the objective function is to minimize the overall score (which is the multiplication of the proximity weight and the inter-facility distances), as the proximity weight increases, the model

targets to decrease the distance to minimize the score. The drawback in the models with no negative values in the "undesirable" proximity measures is that when two facilities are required to be far from each other the model would not really focus on getting them far from each other. Accordingly, those models focus on minimizing the distances between facilities with strong relationships without really focusing on maximizing the distances between the facilities with undesirable relationships. That is why in this research, any two facilities with undesirable relationship would have a proximity weight of a negative value, so as the distance between those two facilities increases the smaller the score. Accordingly, the model would focus on both minimizing the distances between facilities with strong relationships and maximizing the distances between the facilities with undesirable relationships.

2.2.2 Different objective functions

provides a summary of the different objective functions used in the site layout optimization models of the previous research mentioned in the literature.

Table 3: Different objective functions used in previous research

Research	Objective Function
Yeh (1995)	Min. Cost of facility construction + Interactive cost between facilities
Li & Love (1998)	Min. frequency of trips made by construction personnel
Zouein & Tommelien (1999)	Min. Proximity weight + Relocation weight
Hegazy & Elbeltagi (1999)	Min. proximity weight on an exponential scale
Tam et al. (2001)	Min. total transportation costs of resources between facilities
Elbeltagi & Hegazy (2001)	Min. proximity weight on an exponential scale
Zouein et al. (2002)	Min. proximity weight
Khalafallah & El-Rayes (2002)	1) Max. construction safety, 2) Max. construction-related aviation safety 3) Max. construction-related security level, 4) Min. while minimizing all relevant site layout costs
Cheung et al. (2002)	Min. total transportation costs of resources between facilities
Osman et al. (2003)	Min. proximity weight
El-Rayes & Said (2009)	Min. total site layout cost (transportation + relocation)
Ning et al. (2011)	1) Min. total handling cost of interaction flows between facilities. 2) Min. safety hazards/environmental concerns.
Xu & Li (2012)	
Yahya & Saka (2014)	
Andayesh & Sadeghpour (2014)	Min. proximity weight

Chapter 3

Model Formulation

3 Chapter 3: Model Formulation

3.1 Background and Model Methodology

The outcome of this research is the creation of a series of new algorithms for modeling site layout facilities and the production of an optimization model using these algorithms to find optimum locations, angles, geometrical formations, and arrangements for the different facilities. The developed site layout optimization model is mainly formed of four modules: 1) inputs module, 2) calculations module, 3) optimization module, and 4) output module, each of the modules consists of sub-modules as shown in Figure 16.

In the inputs module, the user inputs the a) site boundaries, where the boundaries can assume any geometrical rectilinear or non-rectilinear forms, b) geometries of the site obstacles such as the fixed facilities and access roads, c) geometries of the static and dynamic representations of the temporary facilities, d) desired proximity weights between the facilities, e) desired proximity relationships between the facilities, and f) desired distance measurement technique between the facilities.

In the calculation module, the movement and rotation parameters are set and the model calculates the actual distances between the facilities and the objective function. In this module, the model also has sub-modules that a) ensure non-overlapping between objects, b) ensure the objects to be always inside the site boundaries, c) put soft constraint for buffer zones around the facilities, and d) ensure the dynamic geometrical shapes to be within the given area range.

In the optimization module, the model takes uses Genetic Algorithms described in Section 2.1.2.2 to reach the near-optimum solution.

In the output module, the model outputs the final near-optimum site layout plan and its corresponding layout score.

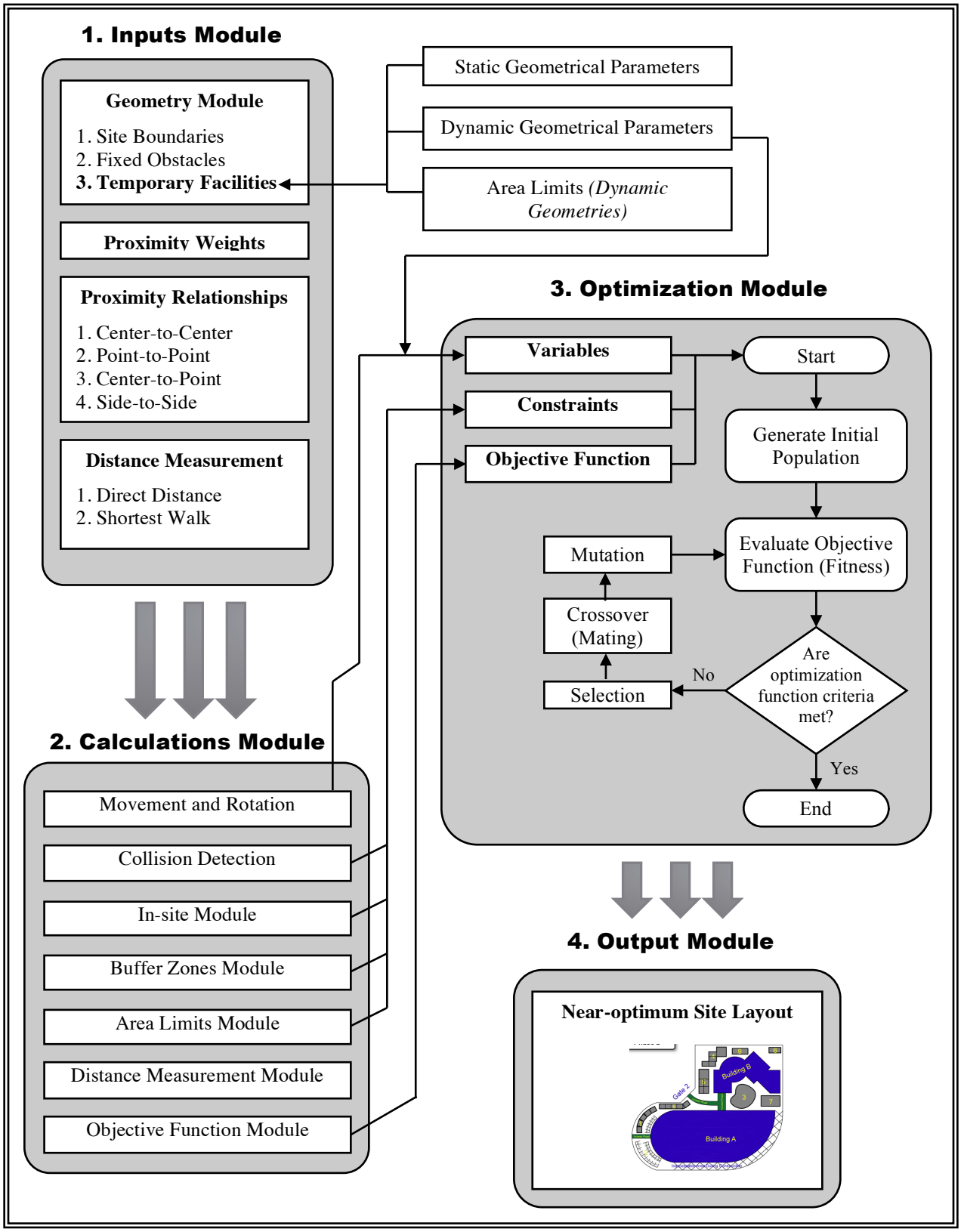

Figure 16: Methodology of the developed optimization model

Several attempts have been investigated to model irregular static and dynamic shapes using normal platforms such as Microsoft Excel® and Matlab® but excessive difficulties where faced as they required intensive scripting capabilities. **Grasshopp er®** has been found to be the most suitable modeling platform for executing the developed algorithms in this research for its powerful visual programming capabilities. For that reason, the modeling capabilities of Grasshopper® were used to execute the algorithms described in this research. So users who wish to execute the discussed algorithms and/or add further developments are preferred to have introductory knowledge of the basic Grasshopper® tools and interface, which is not difficult to obtain. Since there are no found records of using Grasshopper® in construction site layout planning before, this research is considered to introduce researchers to the use of Grasshopper® as the parametric modeling platform (graphical algorithm editor) in site layout optimization; for that reason, formulation of the developed algorithms is extensively demonstrated using different screen shots from the software showing the different visual programming components and parameters of the software that are corresponding to the algorithms.

Grasshopper® is a plug-in for Rhinoceros 3D®, referred to as Rhino®, modeling software. Rhino® is mainly used by architects to facilitate drawing and modeling complex geometrical shapes that are difficult to model using other related software such as AutoCAD 3D® and Revit Architecture®. Grasshopper® is a plug-in for Rhino® that enables parametric modeling; allowing users to model and perform analysis on even more complex shapes in Rhino® by inputting parametric data and formulating parametric relationships instead of drawing. Grasshopper® also has many add-ins for different architectural and environmental analysis modules such as light analysis, heat analysis, ventilation analysis. Grasshopper® combines the mostly graphical approach of working in Rhino® with the powerful algorithmic techniques found in scripting. The benefit of using Grasshopper® is that users don't need to have a high level of scripting or programming experience to generate complex models. Figure 17 provides an example of the capabilities of Grasshopper®.

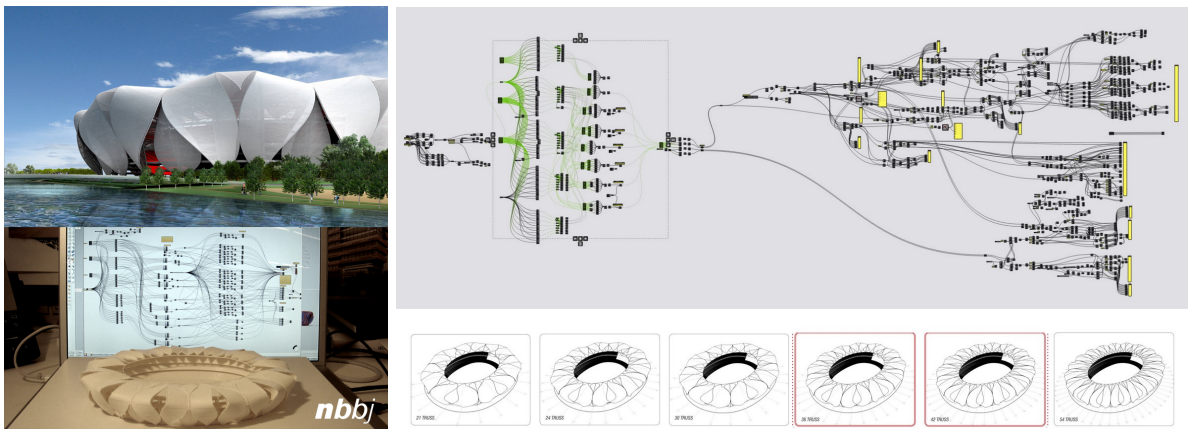

Figure 17: The Hangzhou Stadium covering fully modeled by Grasshopper® in a set of algorithmic steps (Miller, 2009)

The technical term for a Grasshopper® file is a "definition". The objects that make up a Grasshopper file fall into two main classes: ***Parameters*** and ***Components*** (Figure 18). Parameters store data whereas components process data. Components are generally divided into three main parts: 1) a name that is usually located in the middle bar of the component, 2) input grips, on the left side of the component, that take input data to use in whatever operation the component handles, and 3) output grips, on the right side of the component, that contain whatever information is passed on after the component processes it's input. Output grips are sometimes connected to the input grips of other components, chaining them together into additional operations. A parameter has just one input and output grid. Since the parameter only stores data it never needs more than one of each.

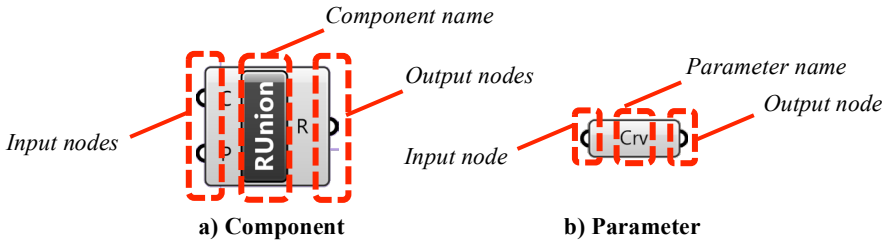

Figure 18: Components and Parameters in Grasshopper®

Galapagos is an add-in tool in Grasshopper® that employs GA in finding near-optimum solutions within Grasshopper®. Galapagos was developed by David Rutten, a developer with McNeel & Associates, and was revealed in 2010 in a lecture by himself in a lecture named "Computing Architectural Concepts" at the Architectural Association in London (Aweida, 2011). The use of Galapagos itself is simple and does not require the user to have strong knowledge

about GA. Galapagos requires the user to input two main inputs: 1) the variables, and 2) the objective function. The user also has the ability to set the mutation rate, the objective (minimization or maximization of the objective function), the population number, and the stopping criteria. Galapagos performs several runs by changing the variables in every run; in each run the objective function is calculated and analyzed. The runs are iterated until the stopping criteria takes place (either by reaching a user-set number of runs, by reaching a user-set stagnant results, or by user-set total running time).

The following shall be noted while using Galapagos:

- Variables connected to Galapagos must be ***Number Slider*** parameters.
- Objective function connected to Galapagos must be a ***Number*** parameter.
- Unlike other GA tools such as Evolver in Microsoft Excel, there are no hard constraints in Galapagos. However, this can be maneuvered by setting very high penalties for constraints in Grasshopper®.
- Galapagos is a single-objective optimization tool. In the case of multi-objective optimization problems it is recommended to use other optimization plug-ins or to convert the problems into single-objective optimization function by funding an equation linking the multi-objectives and converting them into one objective.

3.2 Modeling the Site Facilities

The site layout facilities acquire different geometrical shapes as seen from a top view. The first step of modeling the site facilities is to model their geometrical shapes. Previous site layout optimization models assumed static shapes for the facilities, meaning that the shapes are the same in all iterations. So if a facility is modeled as a square, it stays a square with the same dimensions during all iteration of the optimization model. In this research, an addition is made by introducing dynamic shapes; where a facility is assumed to acquire a certain shape and in every iteration it changes its shape given certain constraints until it reaches a shape that fits in the unoccupied land in the site.

Types of shapes used for modeling the site facilities (Figure 19):

- *Static (fixed) Shapes:*
  1. Rectangle
  2. Triangle
  3. Circle
  4. Irregular polygon
  5. Ellipse
  6. Free-form (splines, bezire curves …etc)

- *Dynamic (non-fixed) shapes:*
  1. Offsetted Planar Curves
  2. Dynamic Freeforms
  3. Dynamic Rectangles

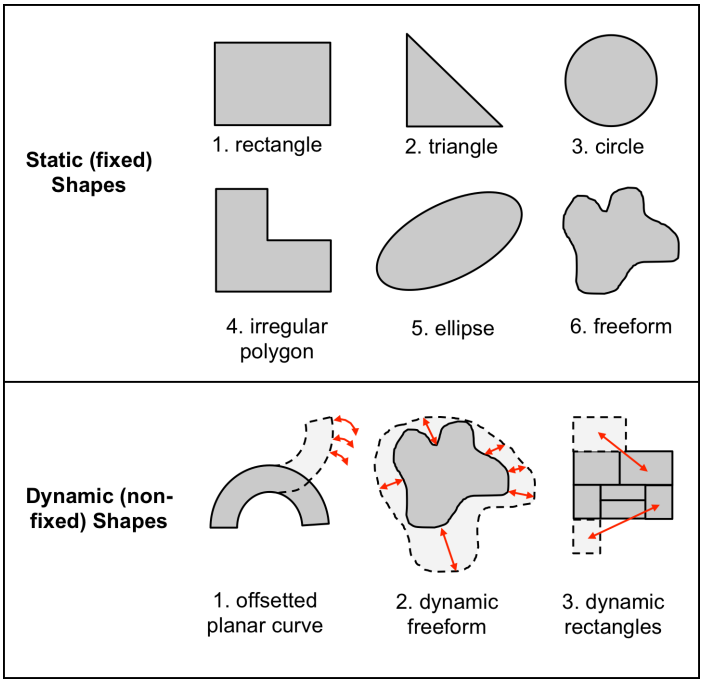

Figure 19: Different shapes used for modeling site facilities (objects)

The following sub-sections describe each of the static and dynamic shapes and their formulation in Grasshopper®. The formulation of the static shapes modeling the site facilities is very simple on Grasshopper® and does not require the use of many parametric components. However, the formulation of the dynamic shapes requires full understanding of their algorithms, so in each of the sub-sections discussing the dynamic shapes, the following order was made: 1) introduction; covering the use of the shape and the algorithm behind it, 2) steps of modeling; covering how the algorithm is executed on Grasshopper®, 3) variables; stating the variables of the dynamic shape,

4) flexibility; showing samples of the different forms of the dynamic shapes by changing the variables, and 5) verification; providing a verification model to verify the formulated algorithm and ensure the absence of any modeling errors.

3.2.1 Modeling Static Shapes

A static shape is defined in this research as a geometrical shape with certain parameters; where the parameters do not change in the different iterations, so the initial form and size of the shape is exactly as its final form and size. In other words, a static shape is a geometrical shape that its size-changing and form-changing parameters are not variables in the optimization model. It can be safely acclaimed that all previous research in the site layout modeling assumed static shapes.

3.2.1.1 Modeling Rectangular Surfaces

Rectangular surfaces are the mostly used geometrical shapes in site layout modeling. They can be used to model many of the site facilities such as storage areas, caravans, workshops, parking lots, external restrooms …etc. To create a rectangle on the used parametric modeling software, Grasshopper®, the ***Rectangle*** component is used to create a rectangle with known parameters and places it in the origin (0,0,0) of the layout. The different parameters of the rectangle are specified by the ***Panel*** parameter and connected to the input nodes of the ***Rectangle*** component. The ***Rectangle*** component requires 4 inputs: 1) P; which is the plane of the rectangle, 2) X; which is the dimension of the rectangle in the X-direction, 3) Y; which is the dimension of the rectangle in the Y-direction, and 4) R; which is the rectangle corner fillet radius. The plane of the rectangle is already preset to the world XY plane. For the dimensions in the X and Y directions, the user inputs the range of these dimensions in a ***Text Panel***. So, if the dimension in X-direction is 4m, the user inputs "-2 to 2" in the ***Text Panel*** connected the input node marked X in the ***Rectangle*** component. For sharp edges, the radius of the corner fillet is 0, while for round edges the radius of the corner fillet is larger than 0 as demonstrated in the comparison shown in Figure 20.

The use of the ***Surface*** parameter is after creating the curve (rectangle, circle, freeform …ec). Creating the curve just draws the outer boundaries, while connecting the curve to the ***Surface*** parameter creates a filled surface with an area inside the boundaries of the curve; hence the rose filling of the geometrical shapes in most of the figures. So in all of the geometrical shapes modeling methodologies, the ***Surface*** parameter is used.

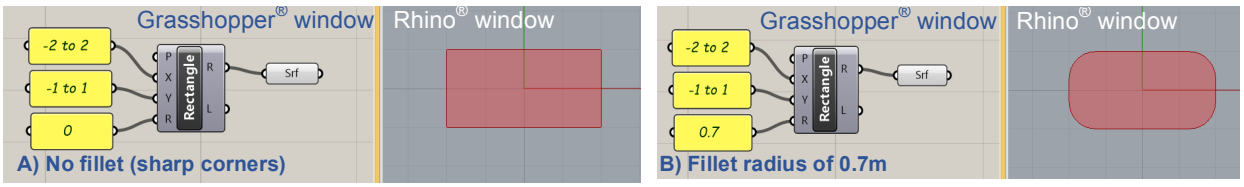

Figure 20: Creating a rectangle using the Rectangle component in Grasshopper®

3.2.1.2 Modeling Triangular Surfaces

Triangular surfaces are not used frequently in site layout modeling. However, it is still beneficial to be able to model triangular facilities as triangular instead of fitting them into rectangular models. There is no tool in Grasshopper® for the sole purpose of creating triangles; but there is a tool for creating polygons. A triangle is a polygon with 3 sides. The ***Polygon*** component is used in this case as shown in Figure 21.

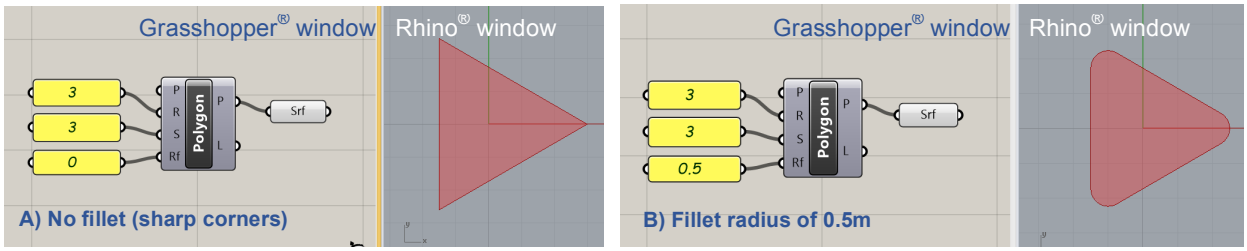

Figure 21: Creating a triangle using the Polygon component in Grasshopper®

The different parameters of the triangle are specified by the ***Panel*** parameter (colored in yellow in Figure 21) and connected to the input nodes of the ***Polygon*** component. The ***Polygon*** component requires 4 inputs: 1) P; plane of the polygon, 2) R; radius of polygon (distance from center to tip), 3) S; number of sides of the polygon, and 4) Rf; polygon corner fillet radius. The polygon plane is already preset to the world XY plane. The number of sides is 3.

3.2.1.3 Modeling Circular Surfaces

Very few site layout models actually model circular site facilities as circles; the rest model them as squares. Circular surfaces are perfect for modeling circular tanks in the site. The most convenient component for creating circular surfaces is the ***Circle CNR*** component.

The ***Circle CNR*** component creates a circle defined by center, normal vector of base plane, and radius (Figure 22). The center coordinates and the radius are specified by the user exactly as the

radius is specified in the ***Circle*** component. The normal vector of the base plane is preset to (0,0,1); which means that the plane of the circle is the XY plane.

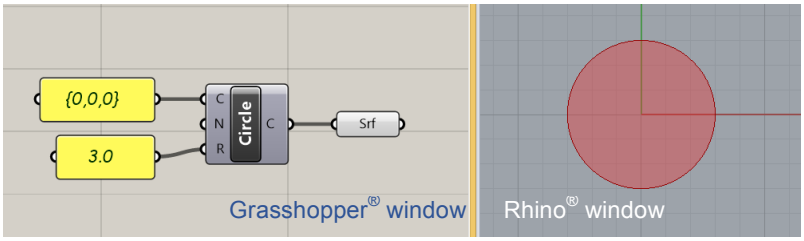

Figure 22: Creating a circle using the Circle CNR component in Grasshopper®

3.2.1.4 Modeling Irregular Polygonal Surfaces

An irregular polygon is a polygon with unequal sides and unequal angles. In reality, most of the site layout facilities are irregular polygons, but site layout models model them usually as rectangles; which causes a waste of area and presence of inefficiencies in modeling. One of the main benefits of using Grasshopper® is the ability to model irregular polygons as is, without having to make unnecessary fittings or interpolation.

To create an irregular polygon with any number of sides, the coordinates of its vertices have to be specified first using the ***Point*** parameter as shown in Figure 23. The points are then connected together by a polyline through connecting the ***Point*** parameters to a ***Polyline*** component. It is very important to ensure that the formed polygon is closed by the changing value of the input node marked by C to 1. This algorithm enables the creation of all possible shapes of irregular polygons.

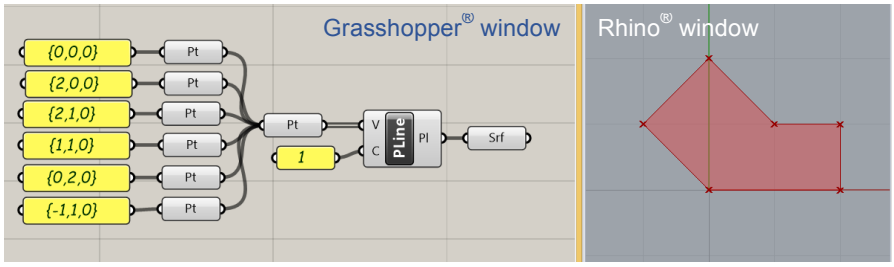

Figure 23: Sample of creating an irregular polygon using the Polyline component in Grasshopper®

3.2.1.5 Modeling Elliptical Surfaces

The ***Ellipse*** component in Grasshopper® creates an ellipse defined by a base plane and 2 radii. The ***Ellipse*** component requires 3 inputs: 1) P; plane of the ellipse, 2) R1; radius in X-direction,

and 3) R2; radius in Y-direction. The ellipse plane is automatically preset to the world XY plane. The user specifies the ellipse radii by a text ***Panel*** as shown in Figure 24.

Figure 24: Creating an ellipse using the Ellipse component in Grasshopper

3.2.1.6 Modeling Freeform Surfaces

Freeforms are the most realistic ways of modeling some temporary facilities such as the material piles; especially that materials such as sand and gravels occupy non-linear space. So, it is highly inaccurate to model these material piles as rectangles or linear polygons. A significant addition of this research to the ongoing site layout modeling research is the ability to model one of the most flexible geometrical shapes; which is the freeform.

Modeling freeform surfaces is not a very difficult task by normal parametric programming software such as Excel® and Matlab® since each point on the curve can be obtained by specific equations depending on the interpolation technique (polynomial, Bezier, NURBS …etc). However, the difficulty comes in developing an algorithm for avoiding collision and overlapping between the freeform surfaces and the other surfaces because the number of points on the freeform surface are infinite and each of the points is at a different distance from the shape centroid. Accordingly, programming the freeform surfaces using scripting was not utilized, and the visual programming using the graphical algorithm editor Grasshopper® was used for its capabilities in simplifying coding and replacing long written scripts into graphical modules.

Creating a freeform on Grasshopper® starts be creating the control points. Creating the control points it made through using the ***Point*** parameter for each point, connected to it the corresponding coordinates as shown in Figure 25. There are several ways of interpolating between points; each type of interpolation results in a different curve. One of the interpolation components is the ***Interpolate*** component; which enables the user to interpolate between the control points at any degree while ensuring that all the points are on the interpolated curve boundary as shown in Figure 25. The main inputs of the Interpolate component are 1) V; the

control points, 2) D; degree of interpolation, and 3) P; a Boolean value for whether the curve is close or open (1 for closed and 0 for opened).

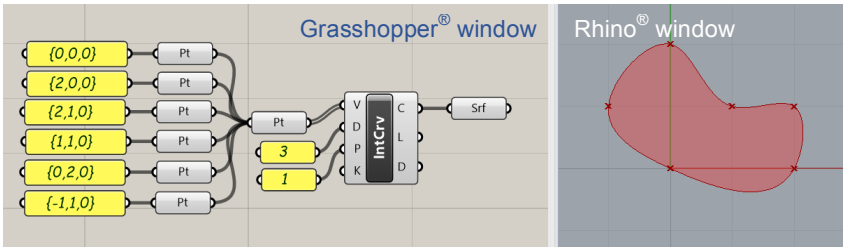

Figure 25: Creating a freeform surface using the Interpolate component in Grasshopper®

Another famous method of interpolating between points in the NURBS curve. The Non-Uniform Rational B-Splines, NURBS, is a mathematical parametric model generating and representing curves and surfaces. Typically, each point of the curve is computed through the weighted sum of a number of control points. The weight of each point varies according to the governing parameter. To interpolate between points using NURBS curve on Grasshopper®, the control points are specified exactly the same as in the normal interpolation technique. Then the points are connected to the ***NURBS Curve*** component along with the required degree and the Boolean 1 specifying that the curve is a closed one. Figure 26 shows the formulation of a sample freeform by using the NURBS curve interpolation. Mathematical representations of other interpolation techniques such as the cubic Bezier curve interpolation and quadratic Bezier curve interpolation are provided in Equations 10 and 11 respectively.

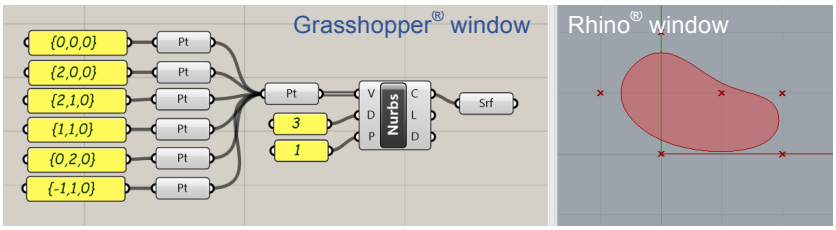

Figure 26: Creating a freeform surface using the NURBS Curve component in Grasshopper®

3.2.2 Modeling Offsetted Planer Curves

3.2.2.1 Introduction to the Offsetted Planar Curves

In this method, the purpose is to form a geometrical shape that ensures a certain minimum width while flexible enough to squeeze itself into tight areas and non-uniform boundaries. In such geometrical shape, the width is fixed all over the span. In the Offsetted Planer Curves Method, a main curve – referred to as the ***Spine*** – is formed from several control points; where these points define the shape of the spine using any type of interpolation available in the software. Note: Grasshopper® provides many options for point interpolation such as Bezier curves, NURB curves, polyline, arc interpolation, curve interpolation by many possible degrees … etc. The Offsetted Planer Curve is suitable for modeling caravans in site; since caravans are placed next to each other where their width is fixed. In strict sites with tight available spaces, laying caravans in a straight line is not a luxury, so it is important to provide a tool that not only models the possibility of irregular layout of caravans, but also provide a proposal of the near-optimum form while ensuring the minimum width and a certain range of plan area.

3.2.2.2 Mathematical Representation of OPC

The Offsetted Planar Curve is constructed by forming a planar curve **c***(t)* that is offsettd in directions that are perpendicular to its direction. Forming the planar curve in this research is made by forming 5 points; a base point P_1 and 4 variable points shown as follows:

$$P_2 = P_1 + V1 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{[Eqn. 6]}$$

$$P_3 = P_2 + V3 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{[Eqn. 7]}$$

$$P_4 = P_3 + V4 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{[Eqn. 8]}$$

$$P_5 = P_4 + V5 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \textit{[Eqn. 9]}$$

Where, Vi is the movement vector between P_{i-1} and P_i.

For cubic Bezier curve interpolation between the points, the path traced by the function **c**(t), given any four points is as follows (Michiel, 2011):

$$\mathbf{c}(t) = (1-t)^3 P_{i-1} + 3(1-t)^2 t P_i + 3(1-t)t^2 P_{i+1} + t^3 P_{i+2} \quad \ldots\ldots \textit{[Eqn. 10]}$$

Where, $t \in [0,1]$, and $2 \leq i \leq 3$

For quadratic Bezier curve interpolation between the points, the path traced by the function **c**(t), given any three points is as follows (Michiel, 2011):

$$\mathbf{c}(t) = (1-t)^2 P_{i-1} + 2(1-t)tP_i + t^2 P_{i+1} \quad \text{.....[Eqn. 11]}$$

Where, $t \in [0,1]$, and $2 \leq i \leq 4$

Other interpolation techniques can be used to interpolate between the control points such as NURB curves, B-splines, third degree polynomial interpolation …etc.

After obtaining the parametric representation of a planar curve **c***(t)* = (x(*t*) , y(*t*)) the curve shall be oriented by increasing values of the parameter *t*. The unit normal vectors **n***(t)* are computed then these vectors are obtained by rotating the oriented tangent vectors **c'**(*t*) = (x'(*t*) , y'(*t*)) counterclockwise through 90 degrees. Then they are normalized to unit length through the length of ***c'(t)*** Equation 12 (Pottmann et al., 2007):

$$\boldsymbol{n}(t) = \frac{(-y'(t), x'(t))}{\sqrt{x'(t)^2 + y'(t)^2}} \quad \text{.....[Eqn. 12]}$$

The offset $\mathbf{c}_d(t)$ at distance *d* to ***c(t) is obtained as:***

$$\mathbf{c}_d(t) = \mathbf{c}(t) \pm d \cdot \mathbf{n}(t) \quad \text{.....[Eqn. 13]}$$

3.2.2.3 Steps of Modeling Surfaces Using Offsetted Planer Curves

Step 1: Defining the points of the spine

The points of the spine are defined using the ***Construct Point*** component. In order to minimize the number of variables in the spine coordinates, the points are made in relation to each other. Meaning that each points depends on the point preceding it; just like the spine of any living organism. For example, as shown in Figure 27, the first point in the spine is defined using the ***Construct Point*** component with a specified x and y coordinates using the ***Number Slider*** parameter. The second point is defined using a combination of the ***Move*** component and the ***Vector XYZ*** component. This makes the second point dependent on the first point. The physical

meaning of this definition is: point 2 is the projection of point 1 that is moved along a vector with the inputted X, Y, and Z distances.

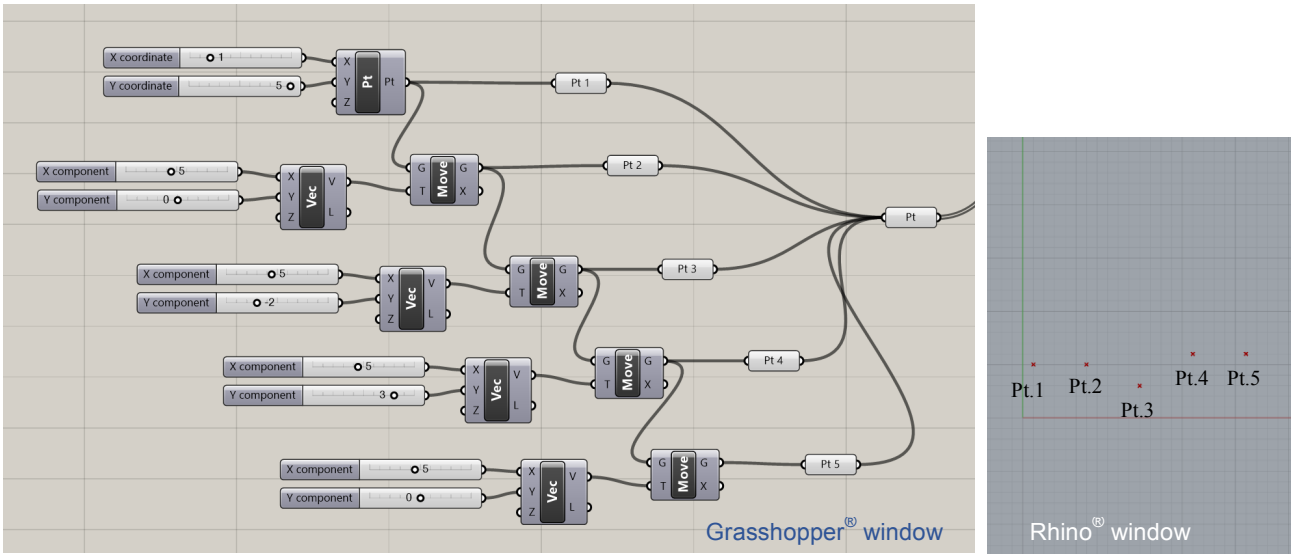

Figure 27: Formulation algorithm of the spine points for the Offsetted Planar Curves

In point 2, the range of the movement in the X-direction starts from 1 to 10 according to the ***Number Slider*** attached to the ***Vector XYZ*** component. The range can be changed at any time by the user. The range of the movement in the y-direction starts from -5 to 5 according to the ***Number Slider*** attached to the ***Vector XYZ*** component. The range can be changed at any time by the user. Figure 28 shows the range of each point relevant to the point preceding point.

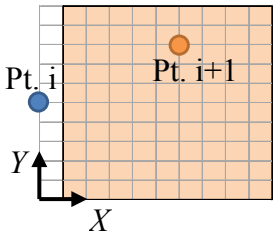

Figure 28: The orange zone represents the movement zone of any point in the spine with reference to its preceding point

Table 4 shows a set of symbols used to specify the relationship of the upper and lower limits of the movement vectors (movement zones) of each of the spine points. It also shows a set of proposed numbers to facilitate the calculations to the users. These numbers are the ones used to verify the algorithm formulation on Grasshopper®. The minimum length of the curve is 4 meters

while the maximum is 40 meters in the X-projection. Figure 27 shows the formulation of step 1 in the algorithm of forming the spine on Grasshopper®.

Table 4: Movement ranges of the spine points (Note: movement is relative to the first point)

	Generic values*				**Proposed values**			
	X-direction		Y-direction		X-direction		Y-direction	
	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
Control Point 2	1	A	-B	B	1	10	-5	5
Control Point 3	1	A	-B	B	1	10	-5	5
Control Point 4	1	A	-B	B	1	10	-5	5
Control Point 5	1	A	-B	B	1	10	-5	5
Values in this table represent the ranges in the number sliders of the vectors forming the spine points. Each number represents a distance between each point and the point preceding it. ** Values of A and B are specified by the user*								

Step 2: Forming the Surface

After the spine points are formed, the ***Interpolate (IntCrv)*** component is used to form a curved line connecting the spine points. The formed curved line shall be referred to as "the spine". The spine can be connected using other curve interpolation components such as ***Bezeir Span*** or ***Nurbs Curve*** or ***PolyArc***. The different combination of the spine points coordinates results in a very flexible range of spine shapes from straight lines to 4th degree curves. Examples of the flexibility in connecting the spine points are shown in Figure 29; where different interpolation components were used for the same reference points of the spine. For example, a user might prefer to connect the spine points with lines instead of third degree curves since the caravans are linear in shape.

The spine is then offsetted, perpendicular to its axis, using the Offset tool as shown in Figure 27. For demonstration purposes the offset value used was 2 meters in each direction, resulting in a total shape width of 4 meters. These two curved lines resulting from the offset shall be referred to as "the offsetted curves". The ***Ruled Surface*** component is then used to form a surface between the offsetted curves.

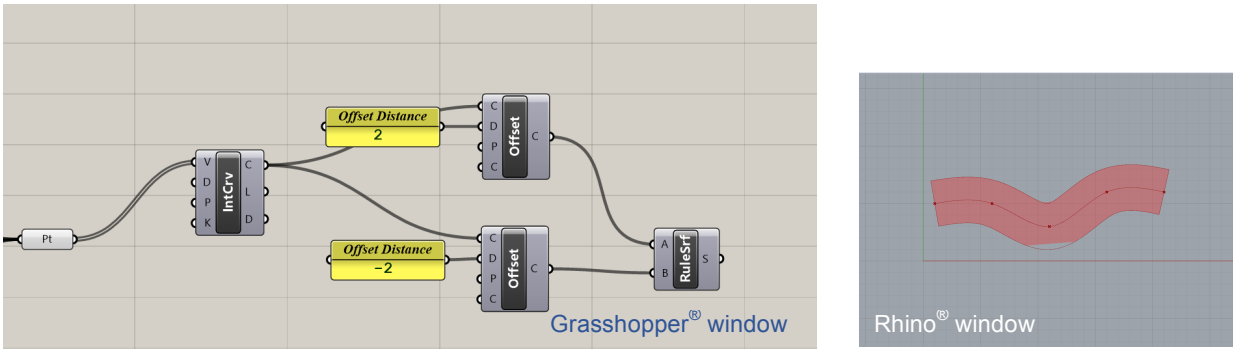

Figure 29: Grasshopper® window showing the formulation algorithm of the Offsetted Planar Curves

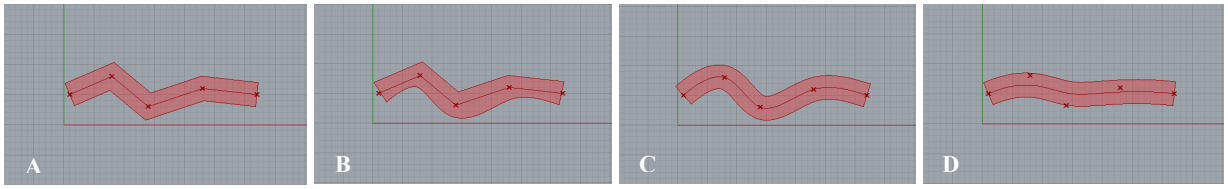

Figure 30: The different shapes resulting from changing the interpolation technique while keeping the control points constant. a) interpolating using polyline, b) interpolating using a combination of polyline and 3rd degree curve, c) interpolating using 3rd degree curve interpolation, d) interpolating using NURBS curve.

Step 3: Adding Rotation and Selective Zoning

To enable the resulting surface to rotate, its formulation is connected to the ***Rotate*** component and the rotation angle is kept as a variable. Object rotation is discussed in Section 3.3.2. In order to minimize the number of possible solutions to reduce the processing time, the resulting surface is bound to the selective zoning algorithm, where it cannot move outside of some pre-specified zones. The selective zoning algorithm and formulation are discussed in Section 3.2.5.

3.2.2.4 Variables of the OPC

The different variables of the OPC are shown in Table 5. Changing the values of the movement vectors of points 2, 3, 4, and 5 in the X and Y directions results in changing the geometrical form of the OPC such as the length and the different angles while keeping the offset width constance throughout the OPC length. The variables responsible for changing the location and rotation of the OPC as a whole (whithout changing its form) are variables number 1, 2, 11, and 12 in Table 5.

Table 5: Variables of the Offsetted Planar Curves Algorithm

	Variables	Recommended Range	Number of possible inputs
1	X coordinate of point 1	0 to 5	6
2	Y coordinate of point 1	0 to 5	6
3	Movement in X direction of point 2	1 to 10	10
4	Movement in Y direction of point 2	-5 to 5	11
5	Movement in X direction of point 3	1 to 10	10
6	Movement in Y direction of point 3	-5 to 5	11
7	Movement in X direction of point 4	1 to 10	10
8	Movement in Y direction of point 4	-5 to 5	11
9	Movement in X direction of point 5	1 to 10	10
10	Movement in Y direction of point 5	-5 to 5	11
11	Surface Rotation	0 to 180 (increments of 30^{o})	7
12	Selective Zoning	1 to 4	4

3.2.2.5 Flexibility of the OPC

The OPC algorithm produces very flexible surfaces that can take any curved shapes with fixed widths as shown in Figure 31. The flexibility of this method allows the modeled OPCs to fit into complicated zones that are difficult, and almost impossible, to model using traditional modeling tools. .In the example used for demonstration, the produced surface has a maximum length of 40 meters and a minimum length of 4 meters. The user can easily change that range by changing the range of the relative X-direction vector movement of the spine points. With the numbers stated in Table 5 there are $1.48x10^{8}$ different shapes that can be formed using the OPC algorithm.

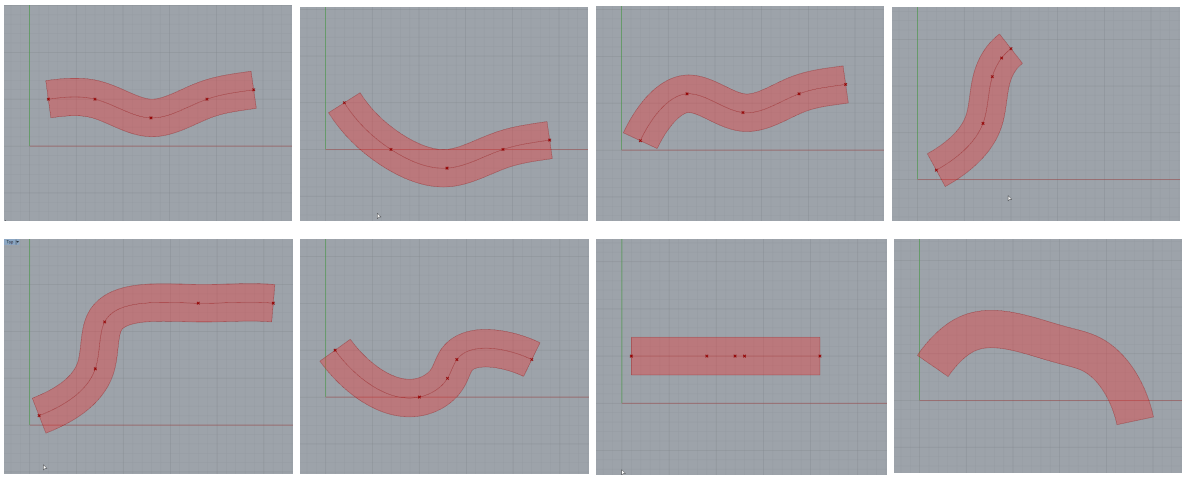

Figure 31: Examples of different possibilities of surfaces modeled using the offsetted planer curves method

Producing real-life caravan layouts on site taking the shapes of the produced geometries from the OPC algorithm might result in internal area waste for the un-evenly distributed areas for the offices. To overcome this waste, it is recommended to create the geometries in the OPC algorithm with additional 10% of area in the model formulation.

3.2.2.6 Verification and Validation of the OPC algorithm

In order to verify the integrity of the Offsetted Planer Curves Method and ensure the absence of any modeling flaws or bugs, a small site layout optimization model was created where the site had four obstacles A, B, C, and D – colored in rose in Figure 32. The obstacles are non-rectilinear geometrical shapes and spaced relatively close to each other so that it is very difficult to lay the test object – colored in blue – using traditional linear modeling methods. The objective function is to minimize the distance between the centroid of the test object and the centroid of obstacle B. The variables were the coordinates of the control points of the spine and the rotation of the offsetted planer surface. The two constraints were: 1) avoid overlapping between any of the objects on the layout, and 2) keep the offsetted planer surface area between 80 m^2 and 120 m^2. Figure 32 shows a snapshot of the formulation of the full verification model on Grasshopper®. The OPC algorithm is said to be verified if it runs without programming mistakes and produces results. The OPC algorithm is said to be validated if the outputted results by the model are as they were expected where the logical solution is reached without interferences.

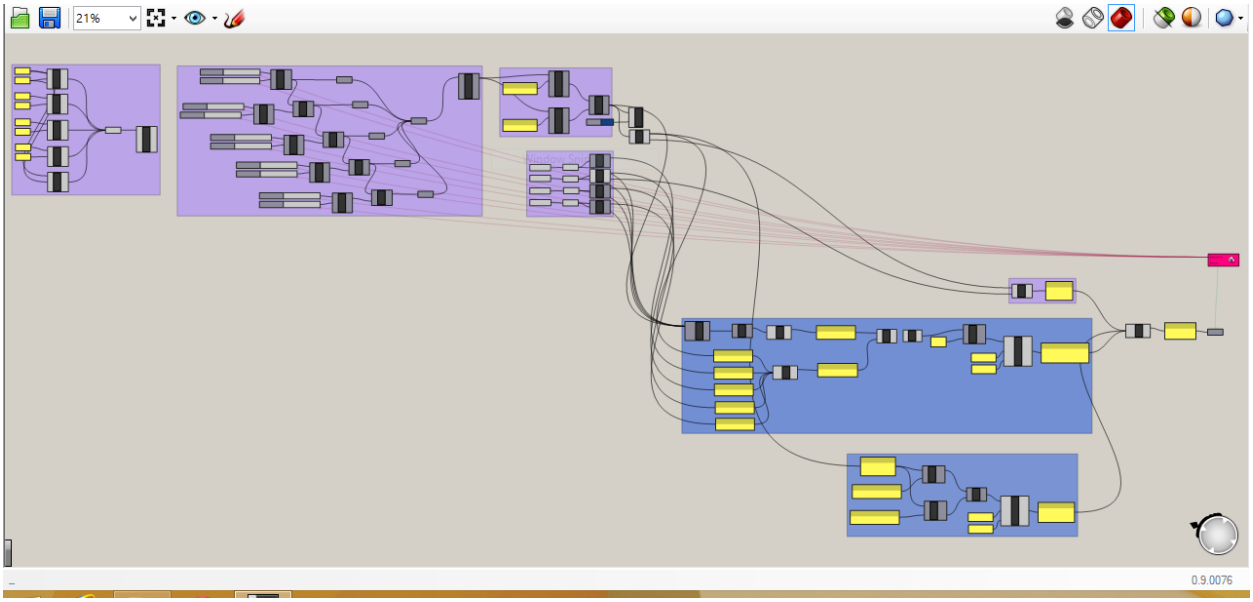

Figure 32: Grasshopper® window showing the full formulation for the verification of the Offsetted Planer Curves Algorithm.

The optimization engine obtained many valid results with no bugs or errors. Four of the valid results are shown in Figure 33 and Table 6. In each of the solutions, the shape of the object is different due to its adaptive and highly flexible nature. Since the model is verified, it is safe to state that it can work on a larger scale on a real site layout problem in parallel with other modules and object shapes. The Offsetted Planar Curves algorithm is not recommended to be used in non-strict site layouts since it requires relatively lengthy running time due to the large number of its variables. Normal static shapes are recommended for non-strict site layouts. However, in strict site layouts, static geometrical shapes may fail in providing valid solutions due to their inefficiencies in the area usage and their failure in adaptation with the surrounding site obstacles. Since this simple model ran without programming errors or running bugs, the OPC algorithm is verified. The solution with the lowest score that is reached by the model is logical and as anticipated; where the centroid of the test object is most close to that of obstacle B while maintaining enough distance to avoid collision; accordingly the OPC algorithm is validated.

Table 6: Verification of the Offsetted Planar Curves Modeling (4 sample solutions)

	Numeric values in the connected number slider							
	Sample Solution 1		Sample Solution 2		Sample Solution 3		Sample Solution 4	
	X	Y	X	Y	X	Y	X	Y
Point 1	37	12	2	17	0	24	7	31
Point 2	7	0	6	-3	3	5	5	2
Point 3	6	0	6	-1	4	3	4	2
Point 4	4	0	4	-1	7	3	7	-1
Point 5	4	0	4	-2	4	1	5	-5
Rotation	90 Degrees		0 Degrees		0 Degrees		0 Degrees	
Score	43.5		28.3		12.5		**8.6**	
Area (m^2)	84		85.8		90.6		97.1	

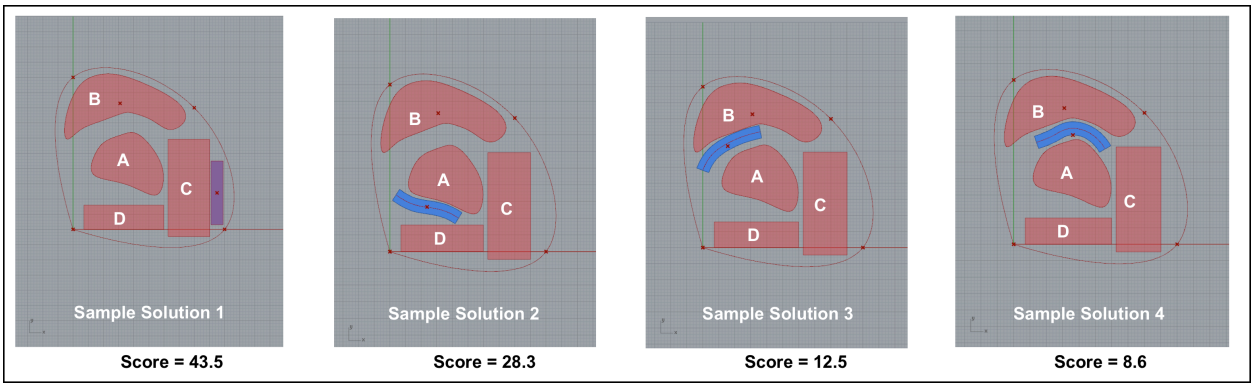

Figure 33: Rhino® window showing possible valid solutions of the verification of the Offsetted Planar Curves Modeling

If the caravans were modeled as normal rectangles as in the literature, the only solution would be sample solution number 1 in Figure 33, which was going to be the near optimum according to literature; however by utilizing the OPC algorithm, other solutions were found and closer to the optimum. So, not only the verification model does verify and validate the algorithm, but it also provides a vivid demonstration of the capabilities of the OPC algorithm.

3.2.3 Modeling Dynamic Freeforms

3.2.3.1 Introduction to the Dynamic Freeforms

A freeform is a curve that has control points connected together using one of many available curves such as 3rd degree polynomial interpolation, Bezeir curve, Nurb curve …etc. A Dynamic freeform is defined in this research as a freeform with flexible control point; meaning that the coordinates of the control points are variables in the optimization model, thus allowing for many flexibilities.

In this algorithm, the purpose is to form a geometrical shape that is very flexible– not limited by a specified shape – to have the ability to be shaped and squeezed into narrow complex areas. Such modeling algorithm is very beneficial for modeling the material piles such as excavation piles and waste piles. Previous site layout optimization models model piles as squares or other linear geometrical shapes. This is not accurate in nature and causes many modeling difficulties such creating waste of space instead of efficiency in using space. In other cases, the models would not find valid solutions in strict site layouts due to the inefficiency of modeling curvilinear shapes and modeling them as approximate linear shapes instead.

Since the material piles are non-uniform and curvilinear in nature, the best way to model them is by curvilinear freeform shapes. Not only this algorithm allows the modeling of freeform shapes, it also allows for changes in the shapes in the different model runs to fit into narrow and complex available spaces on site.

The algorithm works as follows: A point is defined and named the “Reference Point”. Then six surrounding points are defined as “Control Points” of the freeform. The control points are relevant to the reference point. Meaning that the location of each of the control points is defined as the location of the reference point added to a certain vector. The vectors have upper and lower

limits for each of the control points. A curve is then interpolated between the control points shaping the freeform. So in order to move the whole freeform, only the reference point needs to be moved and the rest of the control points will automatically move in the same direction since they are all connected to the reference point. The shape of the freeform is changed by changing the different vectors of the control points. So, the variables are the coordinates of the reference point and the vectors of the control points.

3.2.3.2 Mathematical Representation of Dynamic Freeforms

The mathematical representation of the dynamic freeforms algorithm is as follows:

$$P_1 = P_0 + V1 \quad \text{.....[Eqn. 14]}$$

$$P_2 = P_0 + V2 \quad \text{.....[Eqn. 15]}$$

$$P_3 = P_0 + V3 \quad \text{.....[Eqn. 16]}$$

$$P_4 = P_0 + V4 \quad \text{.....[Eqn. 17]}$$

$$P_5 = P_0 + V5 \quad \text{.....[Eqn. 18]}$$

$$P_6 = P_0 + V6 \quad \text{.....[Eqn. 19]}$$

Where, Vi is the movement vector between P_0 and P_i.

For cubic Bezier curve interpolation between the points, the path traced by the function B(t), given any four points is as follows (Michiel, 2011):

$$B(t) = (1-t)^3 P_{i-1} + 3(1-t)^2 t P_i + 3(1-t)t^2 P_{i+1} + t^3 P_{i+2} \quad \text{....[Eqn. 20]}$$

Where, $t \in [0,1]$, and $2 \leq i \leq 4$

For quadratic Bezier curve interpolation between the points, the path traced by the function B(t), given any three points is as follows (Michiel, 2011):

$$B(t) = (1-t)^2 P_{i-1} + 2(1-t)t P_i + t^2 P_{i+1} \quad \text{.....[Eqn. 21]}$$

Where, $t \in [0,1]$, and $2 \leq i \leq 5$

Other interpolation techniques can be used to interpolate between the control points such as NURB curves, B-splines, third degree polynomial interpolation …etc.

Table 7 shows a set of symbols and relationships used to specify the upper and lower limits of the movement vectors (movement zones) of each of the dynamic freeform control points. It also shows a set of proposed numbers to facilitate the calculations to the users. The proposed values in the table are where the value of A equals to 3, B equals to 10, C equals to 7, and D also equals to 7. The proposed numbers in the table are reflected in Figure 34 where the range of movement of each of the control points relevant to the reference point of the Dynamic Freeform algorithm is represented in a rectangle. According to the current range, there are 48.189×10^9 possible freeform shapes that can be formed using this algorithm; each run in the model constitutes a different shape.

Table 7: Movement ranges of the control points (Note: movement is relative to the reference point)

	Generic values*				**Proposed values**			
	X-direction		Y-direction		X-direction		Y-direction	
	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
Control Point 1	-A	A	0	B	-3	3	0	10
Control Point 2	A	A+C	0	D	3	10	0	7
Control Point 3	A	A+C	-D	0	3	10	-7	0
Control Point 4	-A	A	-B	0	-3	3	-10	0
Control Point 5	-A-C	-A	-D	0	-10	-3	-7	0
Control Point 6	-A-C	-A	0	D	-10	-3	0	7

Values in this table represent the ranges in the number sliders of the vectors forming the control points. Each number represents a distance between the control point and the reference point.
** Values of A, B, C and D are specified by the user*

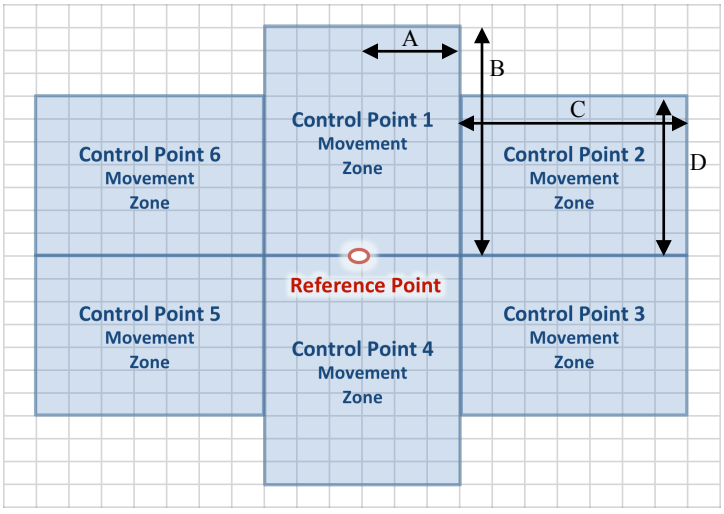

Figure 34: The range of movement of each of the control points relevant to the reference point

3.2.3.3 Steps of Modeling Surfaces using Dynamic Freeforms

Step 1: Defining the reference point

The reference point is defined using the ***Construct Point*** component that is connected to two ***Number Slider*** parameters at its input node marking the X and Y coordinates as shown in Figure 35..

Step 2: Defining the reference points

Each of the reference points is defined by using a combination of the ***Move*** and ***Vector XYZ*** components. The ***Move*** component takes the reference point as its input geometry and takes the ***Vector XYZ*** component output as its input movement vector; thus resulting in a new point (reference point) that it the translation of the reference point in a certain vector. The ***Vector XYZ*** component takes two ***Number Slider*** parameters as its input (one for X-movement and one for Y-movement). This is repeated in all of the six reference points with changes in the values of the maximum and minimum limits of the ***Number Slider*** parameters connected to the ***Vector XYZ*** components. These values are defined in a way not to overlap the zone of movement of any of the control points with each other. A set of equations are shown in Table 8 that are used to specify the relationship of each of the upper and lower limits of the movement vectors (movement zones) of each of the control points. It also shows a set of proposed numbers to facilitate the calculations to the users. These numbers are the ones used to verify the algorithm formulation on Grasshopper®.

Step 3: Forming the Surface

After the control points are formed, the ***Interpolate (IntCrv)*** component is used to form a curved line connecting the spine points. The curve can be connected using other curve interpolation components such as ***Bezeir Span*** or ***Nurbs Curve*** or ***PolyArc***. The ***Surface*** component is then used to form a surface between the boundaries of the formed freeform curve as shown in Figure 35.

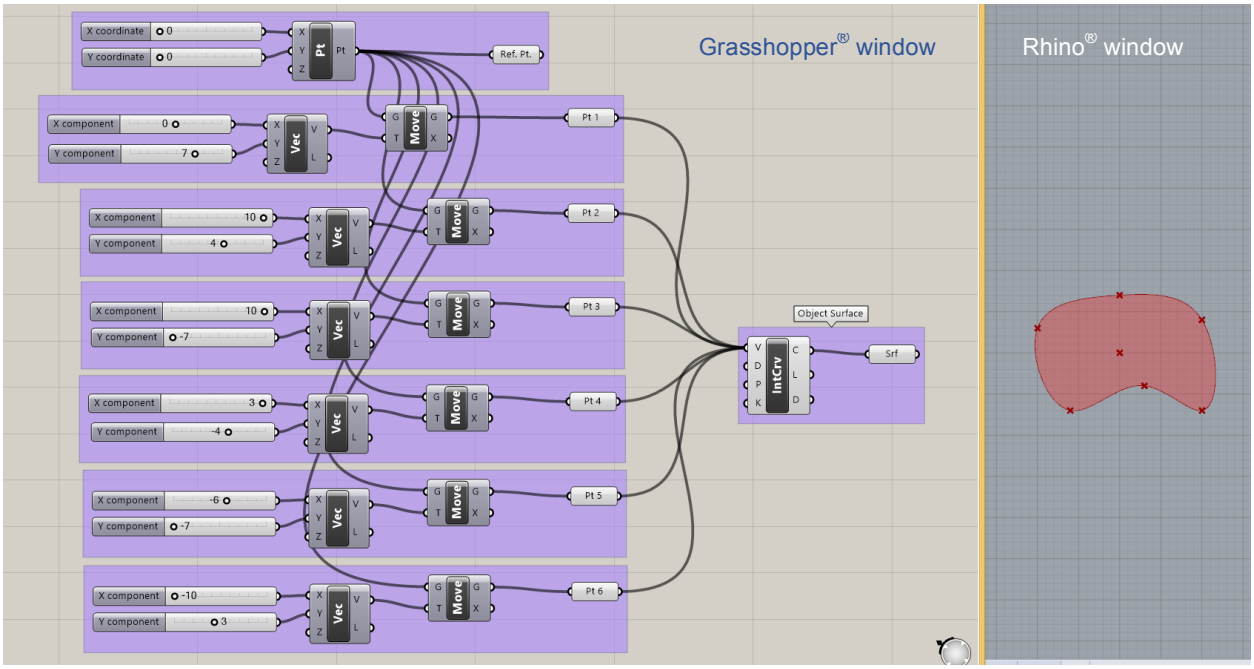

Figure 35: Grasshopper® window showing the formulation algorithm of the Dynamic Free-forms

3.2.3.4 Variables of the Dynamic Freeforms

Table 8 shows the variables in the dynamic freeforms algorithm. The variables and their ranges shown in the table can be modified depending on the project. It is noted that the rotation is not a variable since the flexibility of the produced dynamic freeforms allows them to mimic the behavior of rotation by just changing the values of the control points movement vectors.

Table 8: Variables of the Dynamic Free-forms Modeling (Note: movement is relative to the central reference point)

	Variables	Range	Number of possible inputs
1	X coordinate of reference point	0 to 10	10
2	Y coordinate of reference point	0 to 10	10
3	Movement in X direction of point 1	-3 to 3	7
4	Movement in Y direction of point 1	0 to 10	10
5	Movement in X direction of point 2	3 to 10	8
6	Movement in Y direction of point 2	0 to 7	7
7	Movement in X direction of point 3	3 to 10	8
8	Movement in Y direction of point 3	-7 to 0	7
9	Movement in X direction of point 4	-3 to 3	7
10	Movement in Y direction of point 4	-10 to 0	10
11	Movement in X direction of point 5	-10 to 3	8
12	Movement in Y direction of point 5	-7 to 0	7
13	Movement in X direction of point 6	-10 to -3	8
14	Movement in Y direction of point 6	0 to 7	7
15	Selective Zoning	1 to 4	4

3.2.3.5 Flexibility of the Dynamic Freeforms

The dynamic freeform algorithm provides an unprecedented flexibility marking a breakthrough in the site layout modeling research pool. Figure 36 shows a sample of the different formed shapes using the dynamic freeform algorithm. Each run constitutes a different shape. Shapes formed by the dynamic freeform algorithm can fit into complex narrow areas in the site. With the numbers stated in Table 8 there are 48.189×10^9 different shapes that can be formed using the dynamic freeform algorithm. Figure 36 shows just 8 examples.

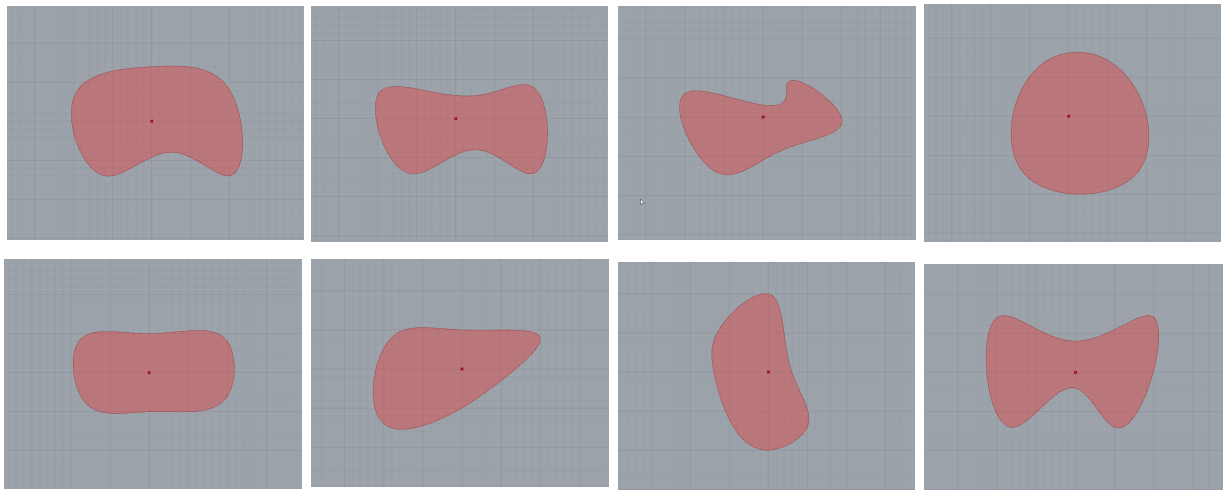

Figure 36: Examples of different possibilities of surfaces modeled using the Dynamic Freeforms algorithm

3.2.3.6 Verification and Validation of DF algorithm

In order to verify the integrity of the Dynamic Freeform algorithm and ensure the absence of any modeling flaws or bugs, a small site layout optimization model was created where the site had four obstacles A, B, C, and D – colored in rose in Figure 37. Some of the obstacles are non-rectilinear geometrical shapes and spaced relatively close to each other so that it is very difficult to lay the test object – colored in blue – using traditional linear modeling methods. The objective function is to minimize the distance between the centroid of the test object, which is the dynamic freeform object, and the centroid of obstacle B. The variables were the coordinates of the reference point and the movement vectors, in both X and Y directions, of the control points of the dynamic freeform. The two constraints were: 1) avoid overlapping between any of the objects on the layout, and 2) keep the freeform surface area between 80 m^2 and 120 m^2. Figure 38 shows a snapshot of the formulation of the full verification model on Grasshopper®.

The purpose of Figure 38 is just to provide a broad view of the full formulation on Grasshopper®. The optimization engine obtained many valid results with no bugs or errors. Four of the valid results are shown in Figure 37 and Table 9. Since this simple model ran without programming errors or running bugs, the DF algorithm is verified. The solution with the lowest score that is reached by the model is logical and as anticipated; where the centroid of the test object is most close to that of obstacle B while maintaining enough distance to avoid collision and maintaining an area between the minimum and maximum limits; accordingly the DF algorithm is validated. Since the model is verified and validated, it is safe to state that it can work on a larger scale on a real site layout problem in parallel with other modules and object shapes.

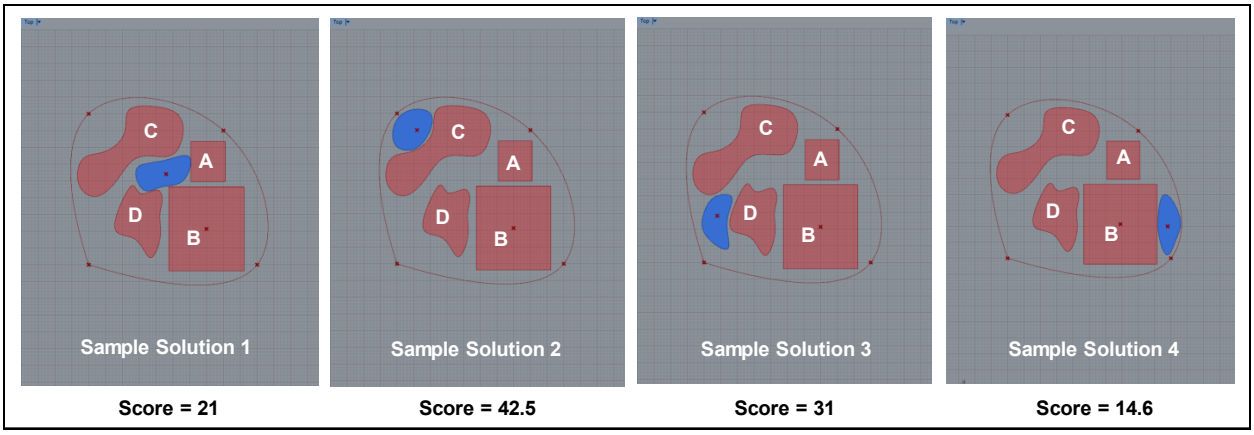

Figure 37: Rhino® window showing possible valid solutions of the verification of the Dynamic Freeforms Algorithm

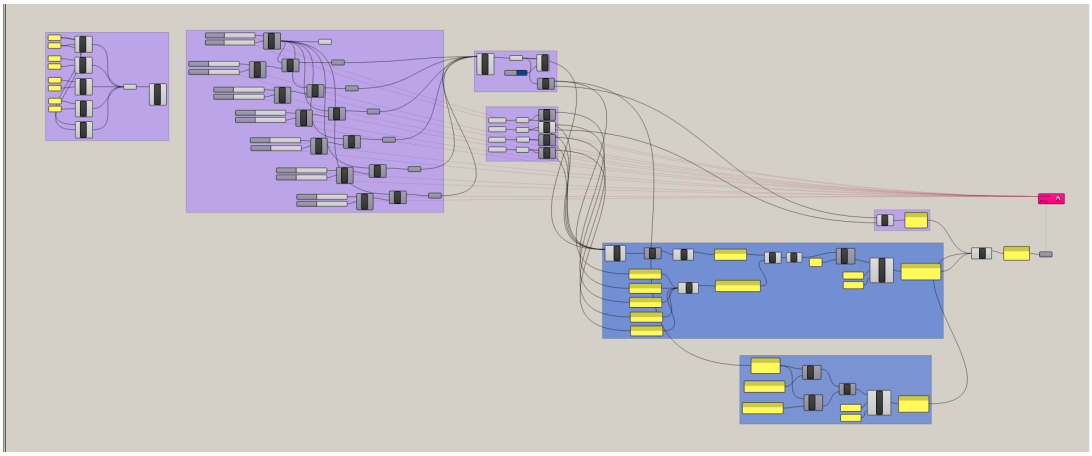

Figure 38: Grasshopper® window showing the full formulation for the verification of the Dynamic Freeforms Algorithm.

Table 9: Verification of the Dynamic Freeforms Algorithm (4 sample solutions)

	Numeric Value in the connected number sliders							
	Sample Solution 1		Sample Solution 2		Sample Solution 3		Sample Solution 4	
	X	Y	X	Y	X	Y	X	Y
Reference Point*	23	27	6	40	4	14	49	10
Control Point 1**	-1	4	-2	6	0	6	-1	20
Control Point 2**	7	5	4	5	4	6	4	1
Control Point 3**	5	-2	3	-2	3	-1	3	-3
Control Point 4**	-2	-4	-2	-6	3	-10	0	-9
Control Point 5**	-6	-5	-6	-5	-4	-5	-3	0
Control Point 6**	-9	1	-7	1	-4	1	-3	1
Score	21		42.5		31		**14.6**	
Area (m^2)	119.8		113.2		105.1		94.2	

* *Value in this row represent the X and Y coordinates of the reference point.*
** *Values in this row represent values in the number sliders of the vectors forming the control points. Each number represents a distance between the control point and the reference point.*

It is noticeable from Figure 37 that the test object (dynamic freeform) has a different shape in each of the solutions; allowing for it to fit in complex non-rectilinear spots, which is impossible to model using previous modeling algorithms.

3.2.4 Modeling Dynamic Rectangles

3.2.4.1 Introduction to Dynamic Rectangles

The term "Dynamic Rectangles" refer to a set of rectangular shapes –forming rectangles– connected together at the edges forming one larger geometrical shape that has rectilinear edges with 90^o angles. The dynamic property in the "Dynamic Rectangles" reflects the ability of the forming rectangles to connect to each other in different arrangements; where the optimum arrangement is selected by the optimization model, so each run would have a different arrangement.

The algorithm of Dynamic Rectangles is developed for the main purpose of modeling site facilities that are composed of several rectangular components such as steel and wood yards. Usually as shown in Figure 39, a wood yard has a cutting yard, a joining yard, an area for storing recycled material, an area for storing new material, and an area for storing the finished units. All of these areas are almost rectangular in shape. Previous site layout optimization models used a simple rectangle to model the wood yard assuming a common arrangement as shown in Figure

39. Such modeling is not valid in the cases of changed arrangements; where, for example, the area of storing the finished units could be put right next to the recycled material storage; forming an outer boundary for the whole wood yard that is no rectangular as shown in Figure 39. From here arose the urge of finding new algorithms to model site facilities formed from small rectangular shapes. The Dynamic Rectangles algorithm fill this gap by dividing the facility into small rectangular units; where each small rectangular unit, referred to as the forming rectangle, model an element inside the facility as shown in part B of Figure 39. The forming rectangles are then programmed to have several possible arrangements such as the ones shown in Figure 40 with taking into consideration that they all have to be connected as to form one final shape without any gaps; meaning that each of the forming rectangles has to be connected to another one or more forming rectangles. The optimization model then selects the optimum arrangement of the forming rectangles depending on the shape and size of the available area in the site layout.

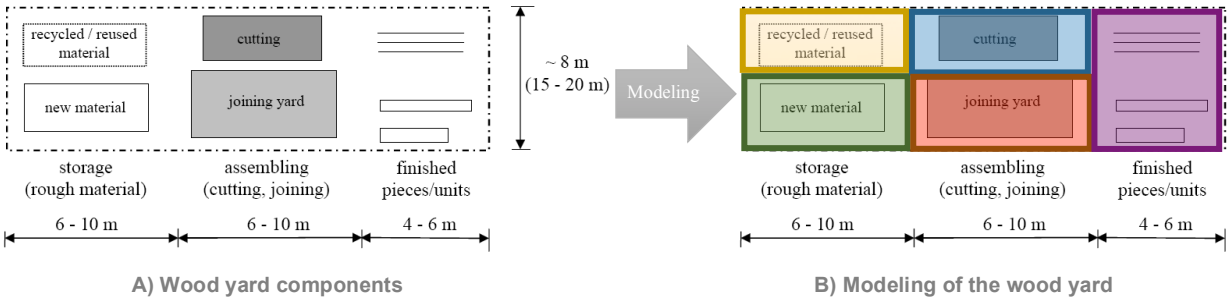

Figure 39: An example of the components of the wood yard (on the left) and its modeling as a series of connected rectangles (on the right) instead of one big rectangle

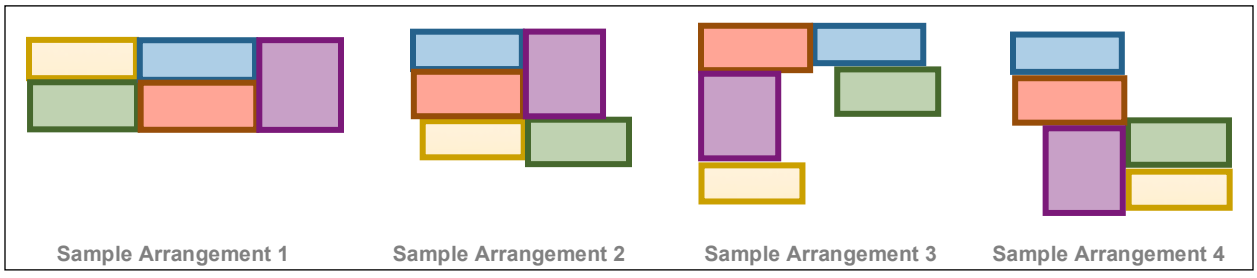

Figure 40: Different possible arrangements for the wood yard using the dynamic rectangles algorithm

The dynamic rectangles algorithm is composed of three steps: 1) Creating the forming rectangles, 2) Setting the different arrangements, and 3) forming the facility surface from the external boundaries of the overall shape.

It shall be noted that the circulation between the different rectangles should be taken into consideration by the user while designing the different arrangements. It also shall be noted that the user, not the mode, specifies the arrangements using the algorithm stated in this research. By understanding the algorithm, users can specify as much arrangements as they desire and save them to the model. The function of the model is to set the optimum arrangement and location of the DR in the overall site layout given the different conditions.

3.2.4.2 Mathematical Representation of the Dynamic Rectangles algorithm

The overall geometrical shape formed by the dynamic rectangles algorithm is the union of several rectangles ***R***, in the scope of this research, the number of rectangles is 5; ***R1, R2, R3, R4, and R5***. Each rectangle has its own dimension in the x direction denoted by ***Lx***, and dimension in the y direction denoted by ***Ly***. The reference point of each rectangle is on its bottom left vertex; where the reference points of the five rectangles are referred to as $\boldsymbol{C_{R1}}$, $\boldsymbol{C_{R2}}$, $\boldsymbol{C_{R3}}$, $\boldsymbol{C_{R4}}$, and $\boldsymbol{C_{R5}}$. The movement of any reference point with a certain translation vector results in the movement of its corresponding rectangle with the same magnitude and direction of the translation vector. The location of each of the 5 rectangles differs depending on the required arrangement according to the following equations:

$$C_{R1_{(i)}} = O_{R1} + V1_i \quad \text{.....[Eqn. 22]}$$

$$C_{R2_{(i)}} = O_{R2} + V2_i \quad \text{.....[Eqn. 23]}$$

$$C_{R3_{(i)}} = O_{R3} + V3_i \quad \text{.....[Eqn. 24]}$$

$$C_{R4_{(i)}} = O_{R4} + V4_i \quad \text{.....[Eqn. 25]}$$

$$C_{R5_{(i)}} = O_{R5} + V5_i \quad \text{.....[Eqn. 26]}$$

Where, $\boldsymbol{C_{Rj_{(i)}}}$ is the coordinates of the reference point of rectangle number *j* in the arrangement number *i*, $\boldsymbol{O_{Rj}}$ is the initial coordinates of the reference point of the rectangle number *j*, and $\boldsymbol{Vj_i}$ is the translation vector of rectangle number *j* in the arrangement number *i*.

3.2.4.3 Steps of Modeling of Dynamic Rectangles:

Step 1: Creating the forming rectangles

The number of forming rectangles in each facility (such as wood yard) is different. For

demonstration purposes the number of forming rectangles in this research is 5 rectangles with varying sizes. Rectangles 1 and 2 have a width (x-direction dimension) of 7m and length (y-direction dimension) of 5m. Rectangles 4 and 5 have a width of 8m and length of 5m. Rectangle 5 has a width of 10m and a length of 10m. Creating each rectangle in Grasshopper® is made by using the ***Rectangle*** component as shown in Figure 41.The Rectangle component is connected to a ***Panel*** parameter in its X-dimension input node and to another ***Panel*** parameter in its X-dimension input node. The numbers in the ***Panel*** parameter specify the dimensions of the rectangles. Note: The user can use any number of rectangles.

Step 2: Setting the different arrangements

A ***Number Slider*** parameter named "Arrangement" is created with a minimum of 0 and a maximum of 4. So its set of output is {0,1,2,3,4}. Meaning that in this model, there are 5 different possible arrangements. The user can specify as much arrangements as he needs.

The location of each forming rectangle is different in different arrangements. To be able to form this algorithm, the ***Stream Filter*** component is used to output the numbers of the forming rectangles'' movement in the X,Y, and Z direction depending on the inputted arrangement. The output from the ***Stream Filter*** component is the vector for the movement of the forming rectangles relevant to the origin. So, for example, in rectangle number 5, if the arrangement is 1, the location of the rectangle would be 15 meters in the X-direction and zero meters in the Y-direction measured from the origin. Also, if the arrangement is 2, the location of the rectangle would be 0 meters in the X-direction and

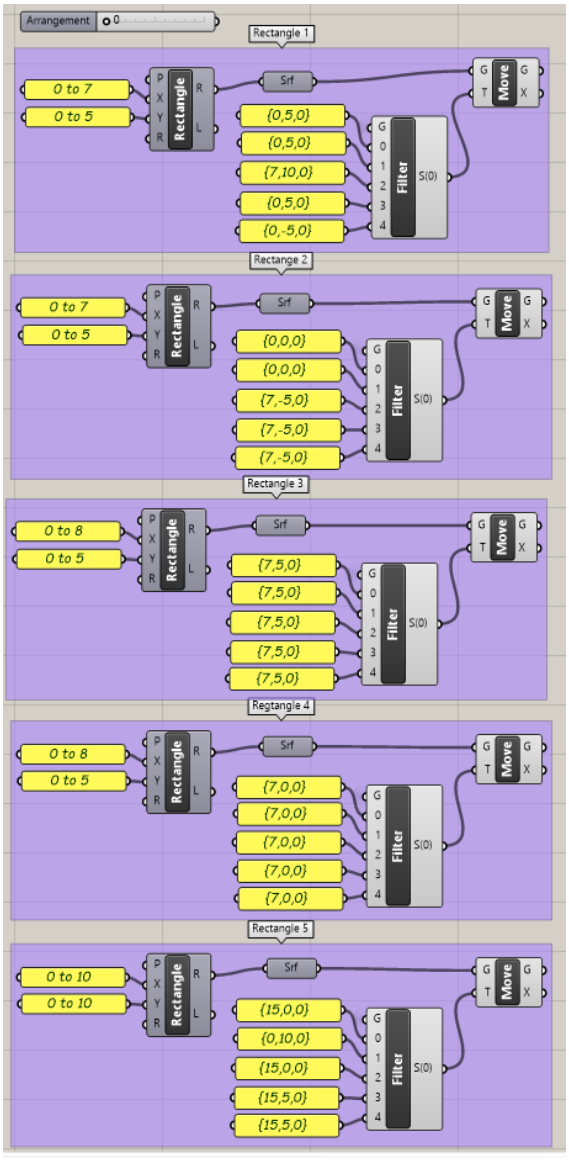

Figure 41: Grasshopper® window showing the formulation algorithm of the Dynamic Rectangles

10 meters in the Y-direction measured from the origin. To be able to do the movement, the ***Move*** component is connected to the output of the ***Stream Filter*** component for each forming rectangle. The algorithm formulation for the Dynamic Rectangles in Grasshopper® is shown in Figure 41.

Note: The output node of the "arrangement" ***Number Slider*** parameter is connected to each of the input nodes marked "G" (Gate) of all ***Stream Filter*** components of the rectangles. However, this connection is hidden in Figure 41 in order for the excessive connection lines not to cause

3.2.4.4 Flexibility of the Dynamic Rectangles

The Dynamic Rectangles algorithm is a new algorithm for modeling site facilities made of rectangular elements. The flexibility is very high, and depends on the number of possible arrangements that the user determines. For demonstration purposes, the number of preset arrangements used is 5, shown in Figure 42. The different arrangements provide different external boundaries; where, the most suitable arrangement is selected by the optimization model depending on the area and shape of the available space in the construction site.

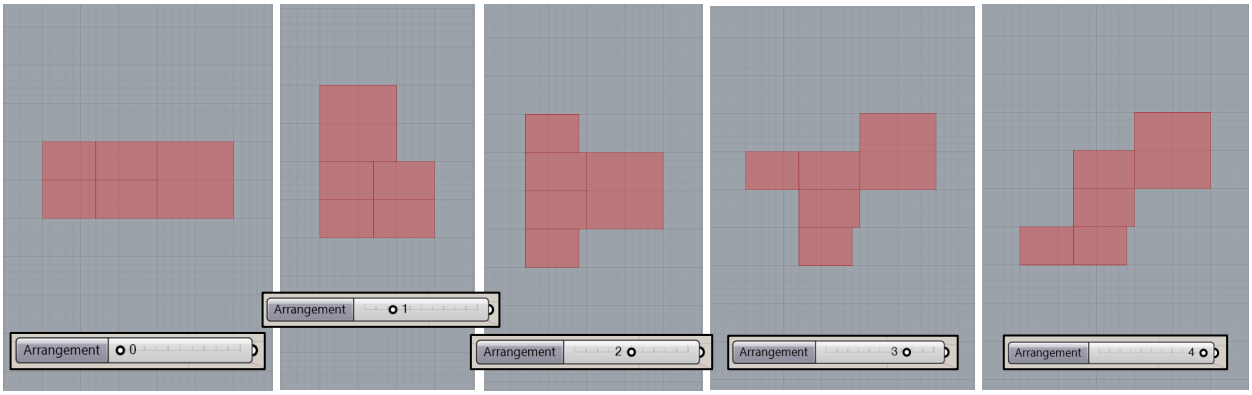

Figure 42: Different arrangements for the same facility using the Dynamic Rectangles algorithm

3.2.4.5 Variables

The formed final shape made by the dynamic forming rectangles has the ability to move anywhere in the site layout using the movement module and will also be able to rotate using the rotation module. So the variables would be: 1) arrangement of dynamic rectangles, 2) movement of full facility in X direction, 3) movement of full facility in Y direction, 4) rotation of full facility in 90^{o} increments, and 5) number of movement zone in the selective zoning module.

3.2.4.6 Verification and Validation of the DR algorithm

In order to verify the integrity of the Dynamic Rectangles algorithm and ensure the absence of any modeling flaws or bugs, a small site layout optimization model was created where the site had four obstacles A, B, C, and D – colored in rose in Figure 43. The obstacles were spaced relatively close to each other so that it is very difficult to lay the test object – colored in blue – using traditional linear modeling methods. The objective function is to minimize the distance between the centroid of the test object and the centroid of obstacle B. The variables were the arrangement number (0 to 4), the movement of the final test object in X and Y directions, and the rotation of the final test object (increments of 90^{o}). The constraint was: avoid overlapping between any of the objects on the layout (using the collision and overlapping prevention module) Figure 44 shows a snapshot of the formulation of the full verification model on Grasshopper®.

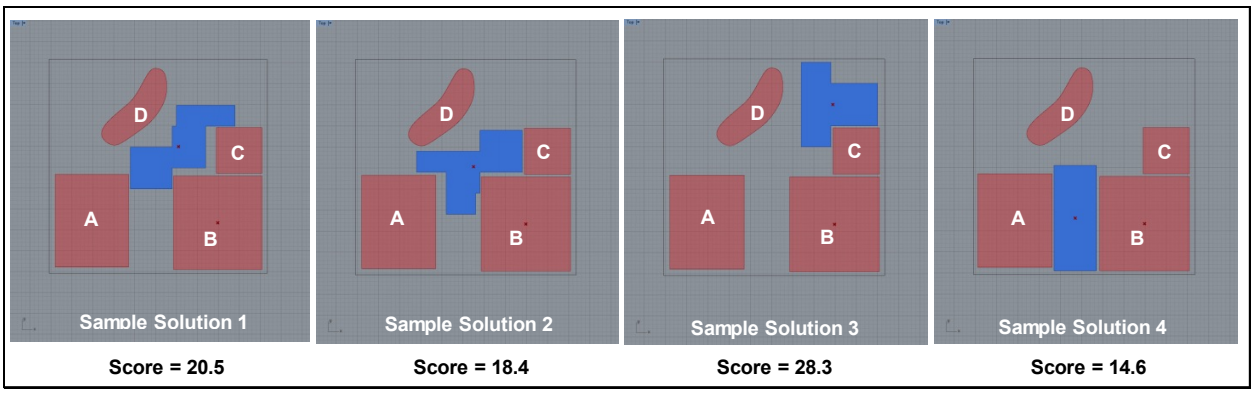

Figure 43: Rhino® window showing possible valid solutions of the verification of the Dynamic Rectangles Algorithm

The optimization engine obtained many valid results. Four of the valid results are shown in Figure 43 and Table 10. Since this simple model ran without programming errors or running bugs, the DR algorithm is verified. The solution with the lowest score that is reached by the model is logical and as anticipated; where the centroid of the test object is most close to that of obstacle B while maintaining enough distance to avoid collision; accordingly the DF algorithm is validated. Since the model is verified and validated, it is safe to state that it can work on a larger scale on a real site layout problem in parallel with other modules and object shapes.

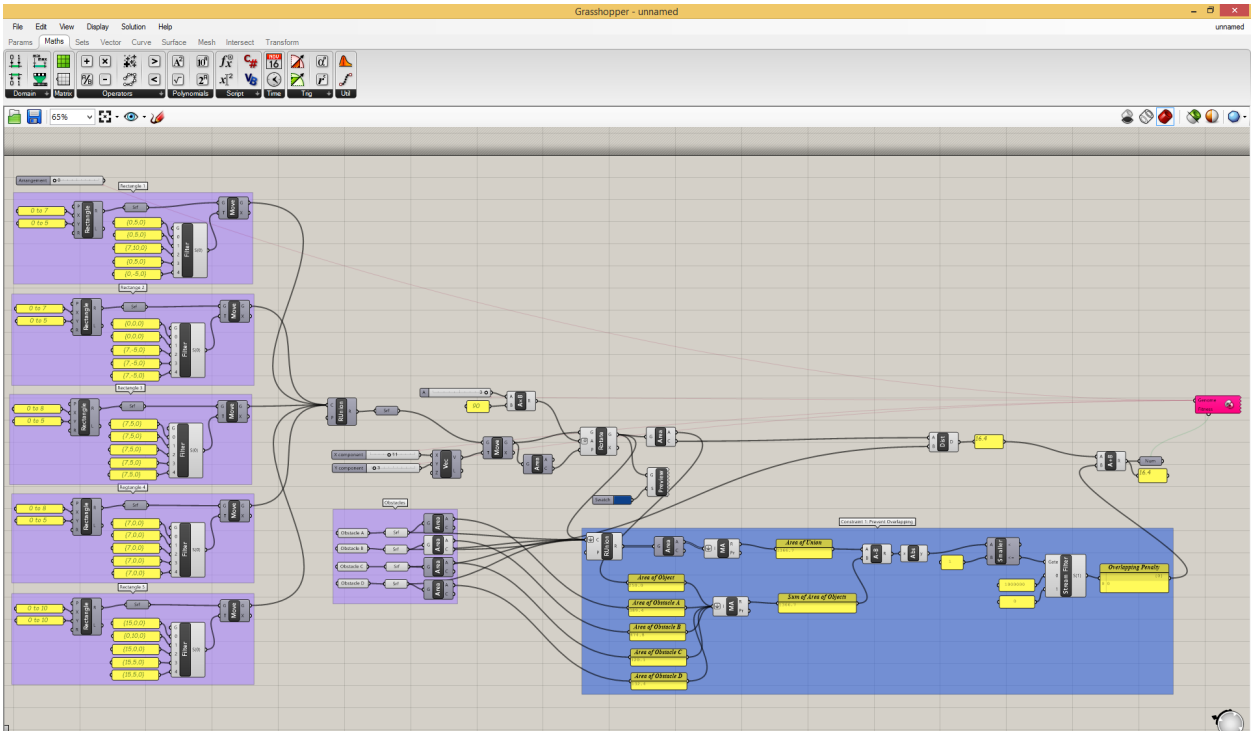

Figure 44: Grasshopper® window showing the full formulation for the verification of the Dynamic Rectangles Algorithm.

Table 10: Verification of the Dynamic Rectangles Algorithm (4 sample solutions)

Name of Number Slider	Numeric Value in the connected number slider			
	Sample Solution 1	Sample Solution 2	Sample Solution 3	Sample Solution 4
Arrangement	4	3	2	0
X-movement	17	14	25	11
Y-movement	20	14	30	3
Angle Slider	2	0	0	3
Score	20.5	18.4	28.3	**16.4**

It is noticeable from Figure 43 that the test object (blue) has a different arrangement in each of the solutions; allowing for it to fit in complex spots, which is impossible to model using previous modeling algorithms. In past site layout optimization models, the test object would have been modeled as just a rectangle, so there would be only few valid solutions in this case, because it wouldn't have fit anywhere in the site except in few places. However, by utilizing different arrangement, the number of valid solution is much higher; where the object can fit in many places on site; which is more realistic in behavior.

3.3 Selective Zoning

3.3.1 The Concept of Selective Zoning

One of the contributions of this research to the construction modeling field is the selective zoning for moving objects. Picture a construction site with irregular boundaries and internal obstacles taking a lot of the site's area resulting in the minimization of the available area of movement for the site layout objects (such as caravans and workshops). In past models, this presents a difficulty because of the many invalid optimization solutions; while in this model, by introducing the idea of selective zoning, such "difficulty" is actually a positive opportunity that leads to significant decrease in possible solutions; thus minimizing the running time of the optimization model.

In traditional site layout optimization models, the zone for which the objects are allowed to move is the area bound by the extremes of site boundaries. Such zone has many invalid solutions for the movement of the objects where the model just runs and produce invalid solutions until it reaches valid ones; which wastes a lot of valuable time. The invalid solutions are the result of the 1) presence of out-of-site areas inside the modeling zone, especially if the site is L-shaped or irregularly shaped, and 2) the presence of the permanent obstacles. Moreover, another drawback is that the number of solutions (valid and invalid) is extremely high in traditional site layout optimization models.

The idea of selective zoning is a simple procedure where, instead of setting the variables to move in one big movement zone with many invalid solutions, many small similar zones are pre-specified and the variables are set to move only in these selective zones. The selective zones are identified by the user and laid in the areas of valid solutions only. This causes a major decrease of the number of possible solutions, thus significantly reducing the running time. Not only this, but also the percentage of valid solutions relative to the total number of possible solutions is very high.

The concept of selective zones is demonstrated in Figure 45 where a sample site layout is presented. The site is L-shaped and has two permanent obstacles. For just demonstration, one object is required to be put on the site without either overlapping with any of the obstacles or

getting off of the site boundaries. In traditional optimization models, the zone of movement of the object coordinates is just one zone with the following ranges:

- X-coordinate [Min → 1; Max value → A; no. of possible solutions → A],
- Y-coordinate [Min → 1; Max value → A; no. of possible solutions → A].

This zone is square-shaped and is represented in part B of Figure 45. Thus the total number of possible solutions is A x A = A^2.

In part C of Figure 45, where the concept of selective zoning is applied, the total number of similar zones = 4. Each of the zones has the dimensions of A/4 and A/2. Thus the total number of possible solutions in this case is (A/4) x (A/2) x 4 = $A^2/2$; which is half of the number of solutions of the traditional method.

Not only does the number of possible solutions decrease significantly by applying the selective zoning concept, the efficiency of the outputted solutions is raised. In part B of Figure 45, the number of valid solutions constitutes only 11.5% of the total number of solutions due to the fact that the movement zone covers areas outside the site and the areas of the obstacles as well. While, on the other hand, in part C of Figure 45, the number of valid solutions constitutes 92% of the total number of solutions due to the fact that the zones are distributed in areas of valid solutions only with minimum overlapping with invalid solutions.

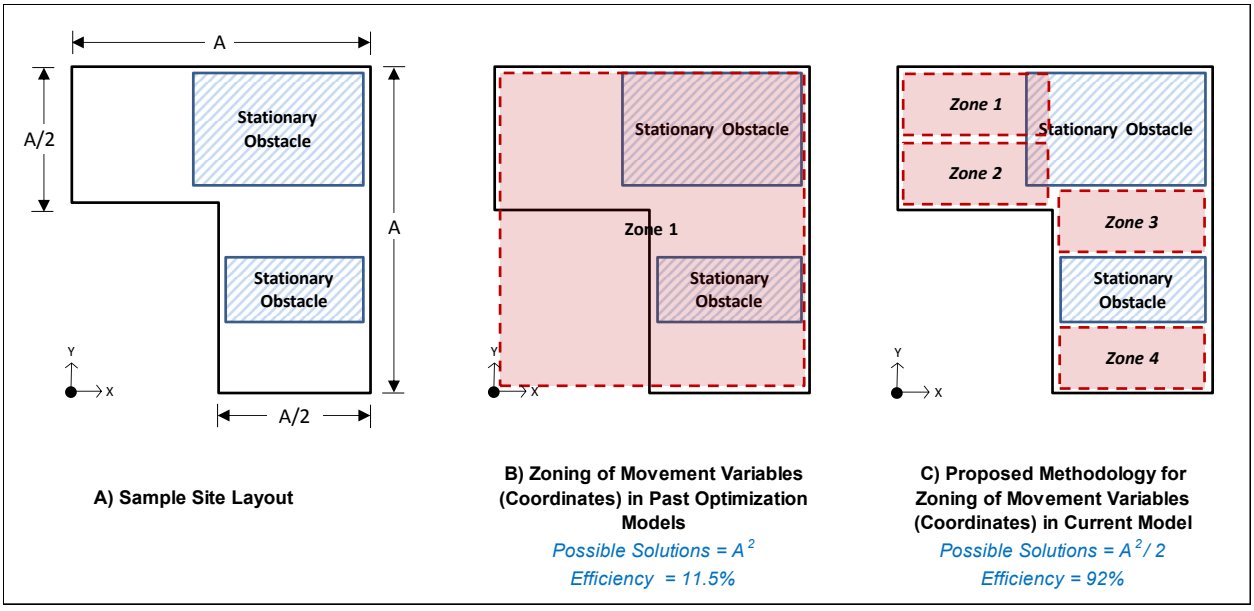

Figure 45: Demonstration of the selective zoning concept.

3.3.2 Mathematical Representation of the Selective Zoning Algorithm

In the selective zoning algorithm, an object is moved using two translation vectors. The first translation vector $t_{sz} = (a_{sz},b_{sz})$ translates each of the points in the object between the zones. The second translation vector $t_{iz} = (a_{iz},b_{iz})$ translates each of the points in the object in the zones themselves. Accordingly, the point $\boldsymbol{p_1} = (x_1,y_1)$, which is the mapping of the point $\boldsymbol{P} = (x,y)$ after translation, is obtained by simply adding the vector $\boldsymbol{t_{sz}}$ *and the vector* $\boldsymbol{t_{iz}}$ to the vector $\boldsymbol{p} = (x,y)$. Thus the vector of the image p_1 is calculated via $(x+a_{sz}+a_{sz},\ y+b_{sz}+b_{iz})$ and a translation can be described with:

$$x_1 = x + a_{sz} + a_{sz} \qquad \text{. . . . . [Eqn. 27]}$$

$$y_1 = y + b_{sz} + b_{iz} \qquad \text{. . . . . [Eqn. 28]}$$

3.3.3 Selective Zoning Algorithm Formulation on Grasshopper®

The selective zoning algorithm depends on the use of a combination of the ***Stream Filer*** and the ***Move*** components to specify the preset movement zones. The first step is to create the object that needs to be moved and located on site. For demonstration purposes, the object is a circular surface with a radius of 2.5m and formed by the Circle component as shown in Figure 46. After that there are two modules; 1) Shifting between zones, and 2) Inter-zone movement. The two modules are connected in series.

Module 1: Shifting between zones: The function of this module is to set the different movement zones that the shape is allowed to move in. A ***Number Slider*** parameter named "Zone" is created with a minimum of 0 and a maximum of 3. So its set of output is {0,1,2,3}. Meaning that in this model, there are 4 different possible movement zones. Then, the ***Stream Filter*** component is used to output the numbers of the forming the object's movement in the X,Y, and Z direction depending on the inputted zone number. The output from the ***Stream Filter*** component is the vector for the movement of the object relevant to its original location. So, for example, if the zone is 0, the coordinate of the object center would be (27.5, 2.5, 0), which is the addition of its original location (27.5, 2.5, 0) [Top left of Figure 46] and the result from the ***Stream Filter*** component (0,0,0) [bottom left of Figure 46]. And if the zone is 1, the coordinate of the object center would be (27.5, 22.5, 0), which is the addition of its original location (27.5, 2.5, 0) and the result from the ***Stream Filter*** component (0,20,0). The ***Move*** component is connected at its input

node to the output node of the ***Stream Filter*** component to make the object move to the zone corresponding to the number from the "Zone" number slider.

Module 2: Inter-zone movement: The function of this module is to set the maximum X and Y limits of the object movement inside each of the zones. A ***Vector XYZ*** component is connected at its input nodes to 2 number sliders; one for X-direction movement and one for Y-direction movement. The Vector XYZ component is connected to a ***Move*** component to set the movement of the object inside the zone. The limits of the number sliders are the limits of the boundaries of the movement zone. So, if the movement zones are required to have an area of 20m x 10m each, the values of the number sliders connected to the Vector component should be as follows: 1) X-direction: min 0 & max 20, and 2) Y-direction: min: 0 & max 10. So, after the model decides a movement zone, from module 1, it determines the location of the object inside that zone from module 2.

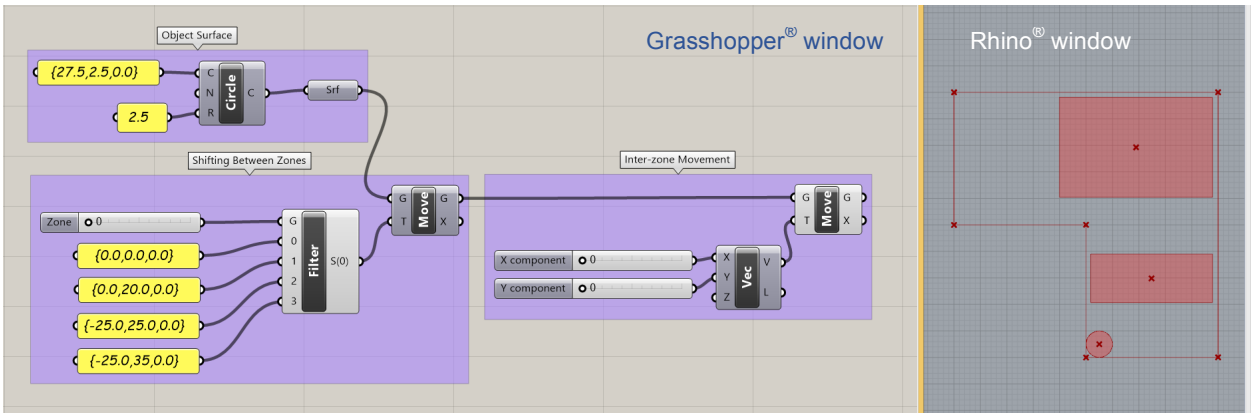

Figure 46: The formulation algorithm of the selective zoning algorithm

To demonstrate the outcome of the selective zoning algorithm, Figure 47 shows the same object in 4 different zones without changing the inter-zone movement sliders.

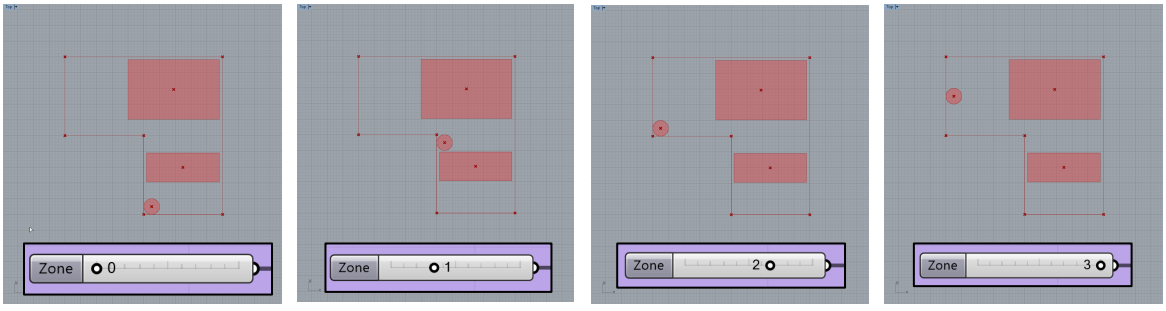

Figure 47: Demonstration of the movement zones using the selective zoning algorithm

3.3.4 Verification and Validation of the Selective Zoning Algorithm

In order to verify the integrity of the Selective Zoning algorithm and ensure the absence of any modeling flaws or bugs, a small site layout optimization model was created where the site had two rectangular obstacles. The site itself is L-shaped as shown in Figure 48. The objective function is to minimize the distance between the centroid of the circular test object and the centroid of the upper rectangular obstacle. The variables were the zone number (0 to 3), the movement of the test object in X and Y directions [Inter-zone movement]. The constraint was: avoid overlapping between any of the objects on the layout (using the collision and overlapping prevention module). Since the problem is simple, the optimum location is known without running the optimization model, but the optimization model was run in order to compare its solution with the known optimum solution for validation. The optimization model ran with no errors so the algorithm is verified. The optimization model outputted the optimum solution and resulted in a location for the object with the minimum possible score. The outputted optimum solution by the model is both logical and anticipated; where the centroid of the test object is most close to that of the upper rectangular obstacle while maintaining enough distance to avoid collision; accordingly the selective zoning algorithm is validated. Since the model is verified and validated, it is safe to state that it can work on a larger scale on a real site layout problem in parallel with other modules and object shapes.

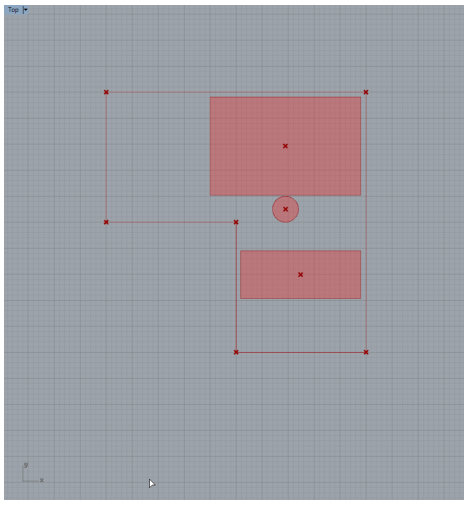

Figure 48: Optimum solution provided by the verification and validation model using selective zoning algorithm

3.4 Movement and Rotation of Objects

A primary variable in all site layout optimization problems is the movement of site facilities. The model moves the site facilities (objects) several iterations. In each iteration it calculates the optimization score. Another primary variable that is not present in some of the older site layout optimization models is the rotation. Rotating the site facilities is important because the purpose of modeling is to get as close as possible from reality; and in reality, the site layout planers lay facilities at different angles in order to ensure the site needs are maintained whilst avoiding collision and overlapping with obstacles. This section explains the formulation of movement and rotation in Grasshopper® and setting them as variables in the optimization model.

3.4.1 Movement of Facilities

3.4.1.1 Mathematical Representation

According to Pottman et al. (2007), a movement (translation) is defined by a *translation vector* **t**, which specifies the direction and magnitude of the translation. To derive a mathematical description, the position of every point p is denoted with **p** – where the position vector points from the origin 0 to the point $p(x,y)$. If a translation is defined by the vector $t = (a,b)$, which maps a point p into a point p_1. Then the vector $\boldsymbol{p_1} = (x_1,y_1)$ is obtained by simply adding the vector $\boldsymbol{t}$ to the vector $\boldsymbol{p} = (x,y)$. Thus the coordinates of the image p_1 is calculated via $(x+a, y+b)$ and a translation can be described with (Pottmann et al., 2007):

$$x_1 = x + a \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{[Eqn. 29]}$$

$$y_1 = y + b \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{[Eqn. 30]}$$

3.4.1.2 Formulation of object movement on Grasshopper®

To move any object on Grasshopper®, the ***Move*** component is used. The ***Move*** component takes the geometry (shape surface) and the vector of movement as inputs. For the vector input, the ***Vector*** component is used. The Vector component is where the movement vector is defined by X, Y, and Z-direction translation. Since the model is 2D, there is no need for translation in the Z-direction; that is why in Figure 49 the Z input for the ***Vector*** component is not connected to anything. The number slider is used for both X and Y inputs of the ***Vector*** component. The number sliders are the variables in this set-up. Maximum and minimum values of the number sliders can be set by double clicking on them and opening their properties window. The

optimizer "Galapagos" understands that these two number sliders are variables – by connecting them to it – and changes their values in each iteration to reach the optimum solution. Figure 49 shows the formulation of a movement order for a sample object. In problems with more than one object, every object should be connected to a ***Move*** component. So for each temporary facility, there are 2 variables for the movement; one for the X-direction and one for the Y-direction.

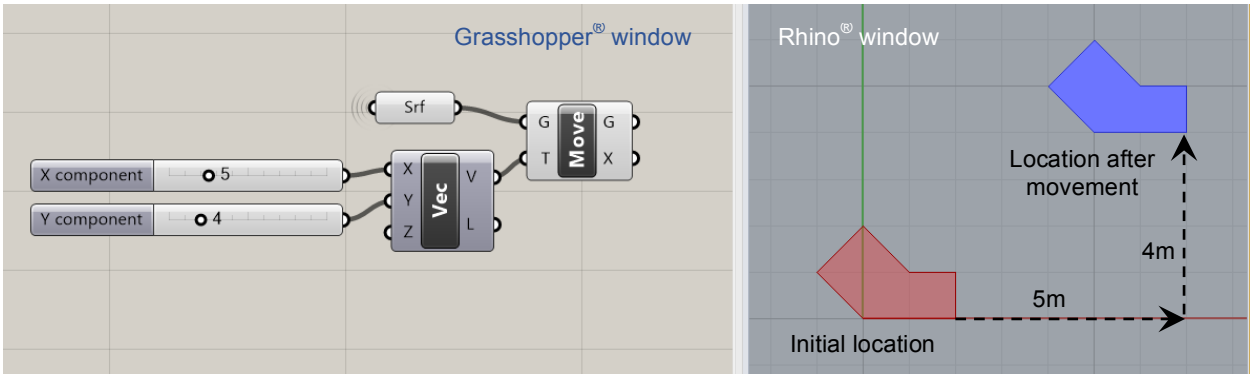

Figure 49: Formulation of object movement on Grasshopper®

3.4.2 Rotation of Facilities

3.4.2.1 Mathematical Representation

According to Pottmann et al. (2007), a rotation is defined by a fixed point *c*, the *center of rotation*, and the *rotational angle θ*. To rotate an object around the origin *o*, any point *p(x,y)* would change its coordinates to be $p_1(x_1,y_1)$; where (Pottmann et al., 2007):

$$x_1 = x \,.\, cos(\theta) - y \,.\, sin(\theta) \quad \ldots\ldots\ldots\ldots\ldots\ldots \textit{[Eqn. 31]}$$

$$y_1 = x \,.\, sin(\theta) + y \,.\, cos(\theta) \quad \ldots\ldots\ldots\ldots\ldots\ldots \textit{[Eqn. 32]}$$

In order to rotate an object around its centroid $c(x_c,y_c)$, the centroid would be considered as its origin so that the following equations would be used to determine the location of the rotated points (Pottmann et al., 2007):

$$x_1 = [(x-x_c) \,.\, cos(\theta)] - [(y-y_c) \,.\, sin(\theta)] + x_c \quad \ldots\ldots\ldots\ldots \textit{[Eqn. 33]}$$

$$y_1 = [(x-x_c) \,.\, sin(\theta)] + [(y-y_c) \,.\, cos(\theta)] + y_c \quad \ldots\ldots\ldots\ldots \textit{[Eqn. 34]}$$

3.4.2.2 Formulation of object rotation on Grasshopper®

To rotate any object on Grasshopper®, the ***Rotate*** component is used. The ***Rotate*** component takes the geometry (shape surface), the angle of rotation (in radian or degrees), and point of rotation as inputs (the point that the object will rotate around). The point of rotation of any shape is its centroid in all the cases in this research. To achieve that, the ***Surface*** component is connected to the ***Area*** component that outputs the area and centroid of the object. The centroid is then connected to the point of rotation input of the ***Rotate*** component. With regards to the rotation angle, it is preferred to rotate the objects in increments of multiples of angles such as increments of 15° or 30° or 45°, depending on the user needs. As the increment decreases, the number of possible solutions increase, which increases the running time. A convenient increment of rotation is 30°. To program the increment in the model, the ***Multiply*** component is used in between the number slider and the angle input in the ***Rotate*** component in the arrangement shown in Figure 50-A. The value of the number slider shall be ranging from 0 to [(360/increment)-1] to cover the 360° rotation of the object. In Figure 50-B, the shape is rotated 90° by setting the number in the number slider to 3 (because it is then gets multiplied by 30 to result in 90).

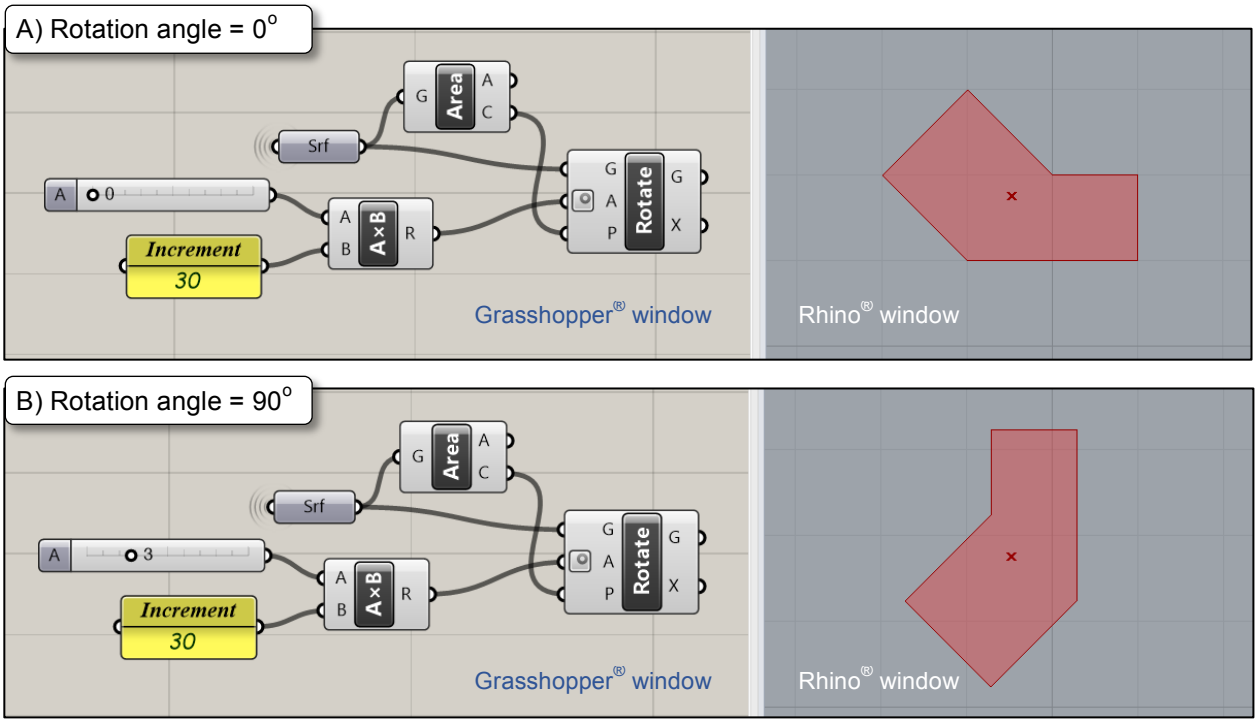

Figure 50: Formulation of object rotation on Grasshopper® (Rotation angle = 90°)

3.5 Collision and Overlapping Prevention

3.5.1 Background about Collision and Overlapping Prevention

After modeling the site layout elements using one or more of the modeling methods previously discussed in Section 3.2, and defining their variables, the constraints have to be defined. The first constraint is collision prevention. Since the main variables in the site layout components are their coordinates, there is a large probability that two or more of them to collide and overlap in the model; which is not realistic in the real world. So, a module was developed to ensure the prevention of collision and overlapping of the different elements in the model.

In Grasshopper® there is no direct way of preventing collision and overlapping, however there is a component that detects collision only and another component that detects overlapping only where both tools have to be used in parallel (Discussed in method 1). In order to simplify the model and minimize the used program components – to minimize the running time – two other modules were developed that detect both collision and overlapping at the same time (Discussed in method 2 and 3).

Since the used optimizer Galapagos® does not have a direct method of inputting optimization constraints, the constraints are inputted as penalties that are added or subtracted from the objective function (The addition or subtraction of the penalty depends on whether the user requires to maximize the overall score or minimize it).

3.5.2 Method 1: Combining the built-in collision detection and inclusion detection modules

In this method, a combination of the built-in collision detection and the inclusion detection components were used.

3.5.2.1 Collision Detection

The ***Collision Many\Many*** Component is used to test for collision between multiple objects. It takes input from the Parameter that has the surfaces (objects) that the user needs to check for collision. The ***Collision Many\Many*** Component outputs a Boolean **List** of results of the collision tests performed; True [1] for collision, and False [0] for non-collision. For example, if

there are 4 objects that need to be tested for collision and two of them are colliding with each other, the output of the ***Collision Many\Many*** Component would be {False 0, True 1, True 1, False 0} *(the arrangement of the True and False order depends on which objects in the input are colliding, the important thing is that the output shows two True values for the collision and tow False values for the non-collision)*. If the four objects were not colliding, then the output would be {False 0, False 0, False 0, False 0}.

The **List** outputted from the ***Collision Many\Many*** Component is connected as input to the ***Create Set*** Component. The ***Create Set*** Component creates and outputs the valid set from a list of items; where the valid set contains only distinct elements. In other words, the ***Create Set*** Component takes a list and removes the repeated additional items from it. For example, if the input list is {False 0, True 1, True 1, False 0}, the output set would be {False 0, True 1}.

The set outputted from the ***Create Set*** component is connected as a component to the ***Mass Addition*** Component. The output of the ***Mass Addition*** component is the summation of all the value in the set of input. So, in the case of one or more collisions, the output of the ***Mass Addition*** component would be =1. In the case of no collision, the output of the ***Mass Addition*** component would be = 0.

Without using the ***Create Set*** Component and by just connecting the output list from the ***Collision Many\Many*** component to the ***Mass Addition*** component, the resultant would be the number of collisions occurring in the model, which could be as high as the number of analyzed objects. But by using the ***Create Set*** component, the resultant of the ***Mass Addition*** component is either 0 in case of no collision, or 1 in the case of collision no matter how many objects collide. This Boolean-like output simplifies the following step, which is the step involving the ***Stream Filter*** component.

The ***Stream Filter*** component filters a collection of input streams. This component acts like a railroad switch; only one of the input streams is allowed to flow to the output parameter depending on the gate stream. It is like an “if statement”. In Figure 51-A, the inputs to the ***Stream Filter*** component are the gate, 0, and 1. The Gate is the resultant from the ***Mass Addition*** component, which is either 0 or 1. In case the gate input is 0 (meaning there is no collision), the

penalty is 0. In case the Gate input is 1 (meaning that there is collision), the penalty is 10,000,000 or as the user specifies.

Figure 51-A shows the full collision detection algorithm of method 1. The penalty is 0 since there is no collision between the objects as shown in the Rhino® window of the same figure. In Figure 51-B, the penalty is 10,000,000 since there is collision between some objects as shown in the Rhino® window in the same Figure.

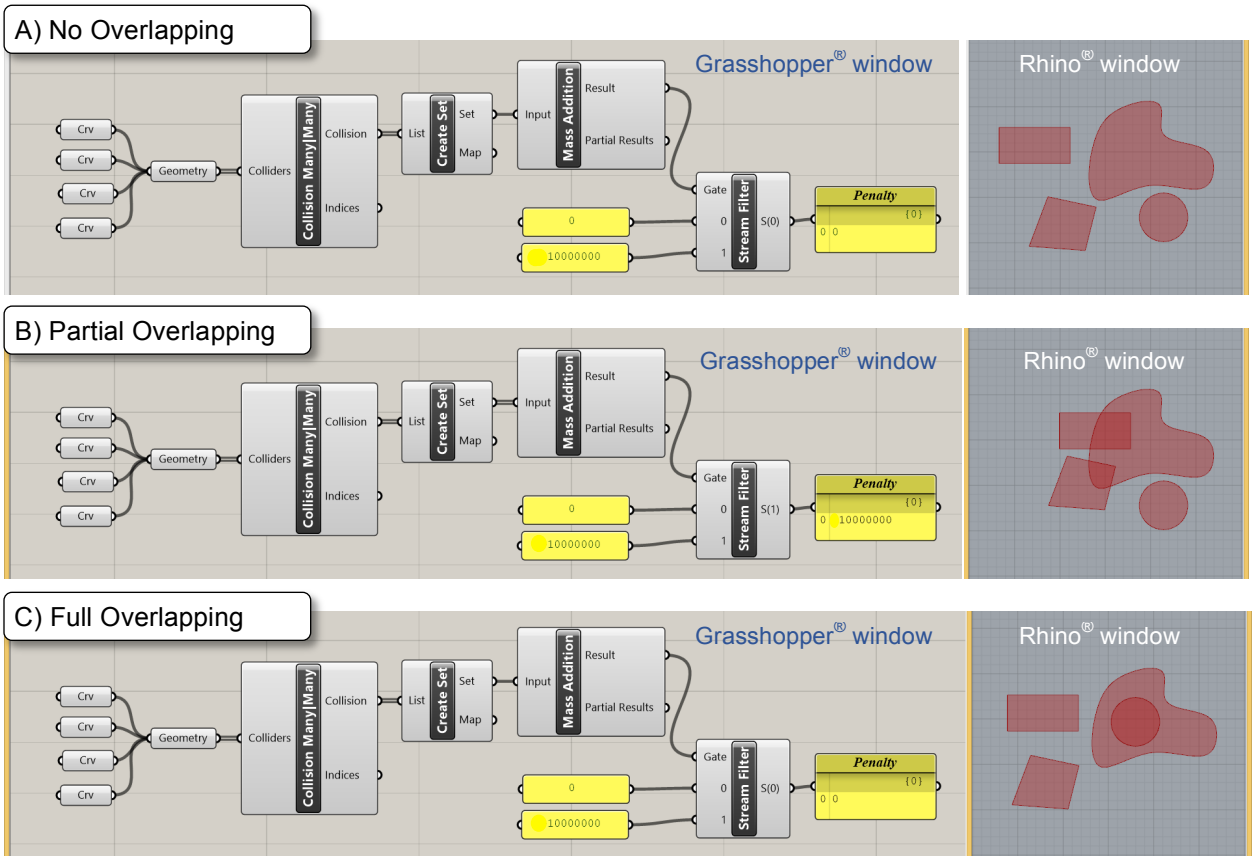

Figure 51: Method 1 of the collision detection algorithm (case of collision)

Drawback: the collision detection algorithm in this method detects only collision of the boundaries of objects. It fails to detect inclusions where an object is inside another object as shown in Figure 51-C. Thus, another algorithm is developed to test for inclusion and both algorithms are run in parallel.

3.5.2.2 Inclusion Detection:

The inclusion detection module in Grasshopper® has the main function of testing two objects for inclusion and providing an integer representing the result of whether object 2 is including (surrounding) the selected point in object 1 or not. The resulted integer is translated into penalty in case of inclusion. This module is used in parallel to the collision detection module because the later alone fails to detect full inclusions; it can only detect partial inclusions and collisions between objects. The following paragraphs describe the algorithm and formulation of the inclusion detection by itself, then a description of the utilization of both collision and inclusion detections in parallel is provided.

The ***Point in Curve*** component is used to test a point for containment in a closed curve. It takes input from the Parameter that has the surfaces (objects) that the user needs to check for inclusion. The input node "P" is where the point for region inclusion test is connected, which is usually the point representing the centroid of the shape that is tested for inclusion. The input node "C" is where the curve forming the boundary region is connected. The ***Point in Curve*** component outputs an integer representing the input/region relationship from the resulting inclusion test; [Zero 0] meaning that the tested point is outside the region boundary, [One 1] meaning that the tested point is exactly coinciding with the region boundary, and [Two 2] meaning that the tested point is inside the region boundary.

The output node of the ***Point in Curve*** component is connected to an ***Equality*** component that is programmed to output a Boolean of {True 1} in the case of full inclusion; where the output from the ***Point in Curve*** component equals to 2. The output of the ***Equality*** component is connected as an input gate at a ***Stream Filter*** component which is programmed to result in a penalty in the case of full inclusion; where the output from the ***Equality*** Component equals to 1.

Figure 52-A shows the full inclusion detection algorithm of method 1 in the Grasshopper® window. In that figure, the penalty is 0 since there is no inclusion (the circular surface is not contained by the freeform surface) as shown in the Rhino® window. In Figure 52-B, the penalty is 10,000,000 since there is a case of full inclusion; where the circular surface is fully contained by the freeform surface.

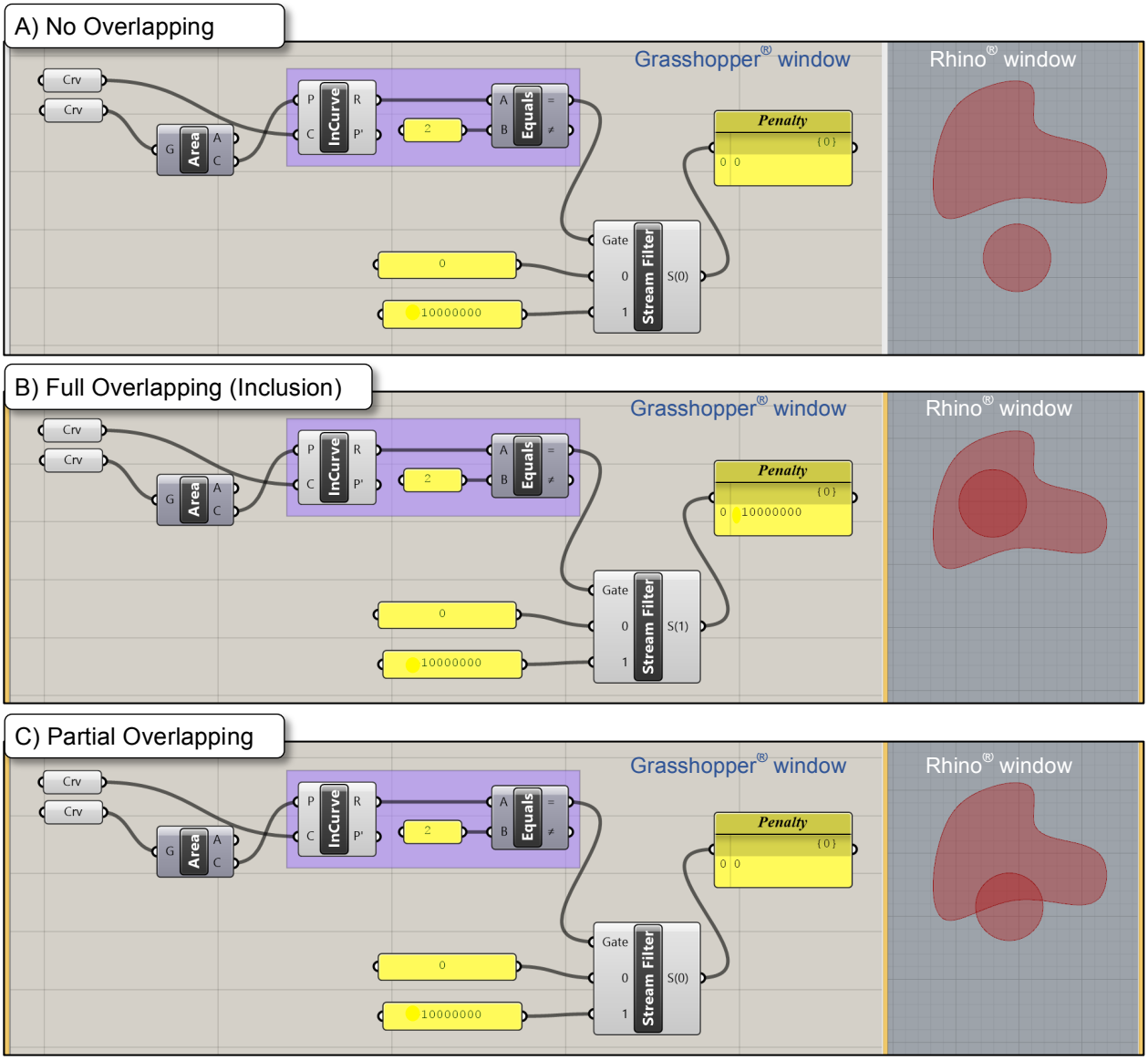

Figure 52: The inclusion detection algorithm of Method 1

Drawback: the inclusion detection algorithm fails to detect partial inclusion or collision. As shown in Figure 52-C the inclusion detection algorithm gave a penalty of 0 despite the fact that the shapes are overlapping and colliding. Such failure occurs because the point that is selected to be tested for inclusion is the centroid; which in the case of the circular surface lies outside the boundary of the enclosing surface.

3.5.2.3 Combination of collision and inclusion detection modules:

The collision detection algorithm of method 1 detects collision (partial inclusion) and fails to detect full inclusion. And the inclusion detection algorithm of method 1 detects full inclusion and

fails to detect collision (partial inclusion). Accordingly, both algorithms have to be combined and used in parallel in order to provide credible results in all cases of partial and full inclusion. Table 11 shows the different outputs of the different components in both the collision and inclusion detection modules used in method 1.

Table 11: Different results from the method 1 collision and inclusion modules

Method 1 of collision and overlapping detection		**In the case of collision (partial inclusion)**	**In the case of full inclusion**
Collision detection module	Output from the ***Collision Many\Many*** Component	1	0
	Penalty provided by algorithm	Yes	No
Inclusion detection module	Output from the ***Equality*** Component	0	1
	Penalty provided by algorithm	No	Yes
Combination of both	Output from the ***Mass Addition*** Component	***1***	***1***
	Penalty provided by algorithm	***Yes***	***Yes***

The purpose of using a combination of both modules is to produce an algorithm that has a unified output for both partial and full inclusion. To perform so, the output nodes of both the ***Equality*** component and the ***Collision Many\Many*** component are connected as inputs to a ***Merge*** component as shown in Figure 53-A. The ***Merge*** component merges a group of data streams. Its output is connected to the input node of a ***Create Set*** Component that creates and outputs the valid set from a list of items; where the valid set contains only distinct elements. A Mass Addition component is connected to add all the numbers in the set. The addition would always be equal to 1 in the case of either partial or full inclusion. The addition would be equal to 0 in the case of no inclusion or collision.

3.5.2.4 Verification and Validation of Method 1 (Using the combination of modules):

In order for Method 1 to be verified, it was tested in the following three cases: no overlapping, partial overlapping (collision), and full overlapping (full inclusion). Figure 53-A demonstrates the Grasshopper® formulation of Method 1 showing no penalty due to the fact that there is no overlapping of objects shown in the Rhino® window of the same figure. In the case of partial overlapping shown in Figure 53-B the Grasshopper® formulation outputs a penalty. In the case of full overlapping - inclusion - shown in Figure 53-C, the Grasshopper® formulation outputs a penalty. Thus, Method 1 is verified and validated.

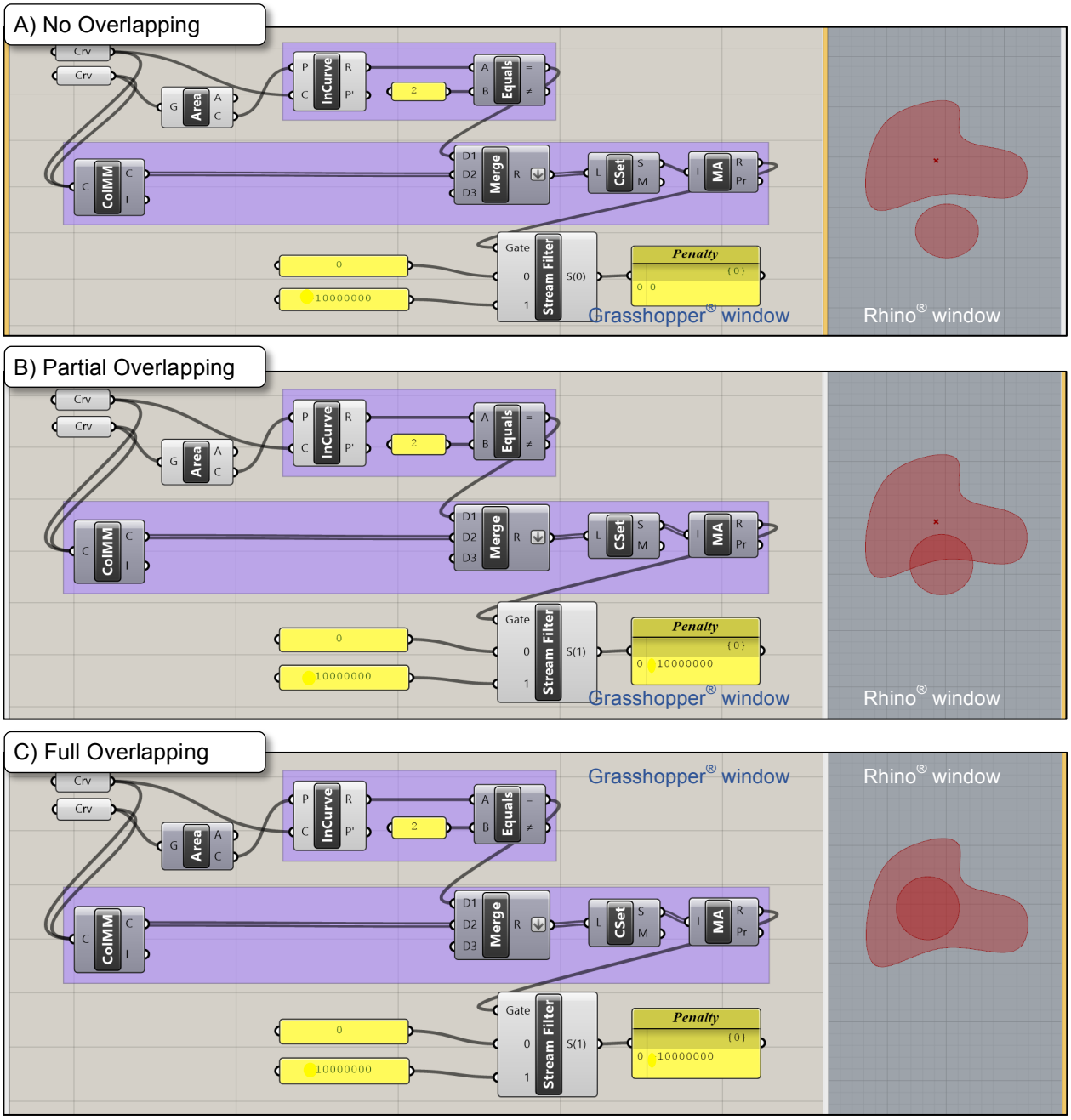

Figure 53: Combination of Modules of Method 1 (full overlapping)

3.5.2.5 Drawbacks of method 1

Using a combination of the built-in collision and inclusion detection modules of Grasshopper® provides a valid algorithm to penalize the cases of partial and full overlapping in a penalty function that would be understood by the optimization model as a constraint. However, it poses two drawbacks. The first drawback is that the formulation of this algorithm on Grasshopper® is relatively complex; which contradicts one of primary purposes of using Grasshopper® as a

modeling tool; especially that there are other methods that were found to be much less complex (See method 2 and method 3). The second drawback, which is the primary reason for not selecting this method, is that it cannot test for the inclusion of more than one object in another object; so it is unrealistic to use this since it is crucial to test for overlapping and inclusion in all of the objects.

3.5.3 Method 2: Using the Area Union Approach

3.5.3.1 The algorithm of Method 2 in collision and overlapping prevention

In this method, a whole new mathematical approach is followed for overlapping detection. This approach is made applicable by the ***Region Union*** Component of Grasshopper®. The algorithm behind this method is as follows:

1. Specify the area of each object [A_i].
2. Get the summation of the areas of the objects [ΣA_i].
3. Union the surfaces of the objects into one surface.
4. Specify the area of the resultant of the union of surfaces [A_u].
5. There are only two possible outcomes:
    a. $\Sigma A_i = A_u$ → means that there is NO overlapping → No penalty.
    b. $\Sigma A_i > A_u$ → means that there is overlapping → Penalty.

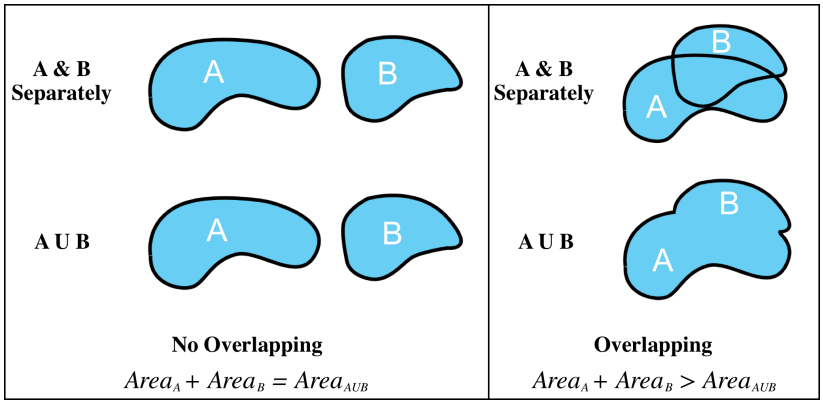

Figure 54: Visual explanation of the idea behind overlapping detection using the surface union

Figure 54 provides a visual explanation of the utilization of the surface union in the overlapping detection. In the left side of Figure 54, objects A and B are not overlapping, the union of their surfaces is actually their same surfaces with no change in the boundaries or areas. So the addition

of their areas as separate surfaces is equal to the area of the union of their surfaces. On the other hand, in the right side of Figure 54, objects A and B are overlapping, so the union of their shapes deletes the repetition of the overlapped area. Accordingly the union of their shapes is one shape with an area that is less than the summation of the their areas separated.

3.5.3.2 Formulation of Method 2 on Grasshopper®:

As shown in Figure 55, after the surfaces are specified, the process is branched into an upper branch and lower branch then it is combined back again into a single branch comparing both to each other.

The upper branch yields the summation of the areas of the shapes separately. The ***Area*** component is connected at its input node to all of the objects that are tested for overlapping. The output is a list of the areas of all the connected objects. Then all of the numbers (areas) in the list are added using the ***Mass Addition*** component resulting in $[\Sigma A_i]$ discussed in Section 3.5.3.1.

The lower branch yields the area of the union of objects. The ***Region Union*** component is connected at its input node to all of the objects that are tested for overlapping. The output is a single surface that is defined by the union of all of the connected surfaces. The area of the outputted surface is obtained and represented as $[A_u]$.

The $[\Sigma A_i]$ and the $[A_u]$ are then connected to the ***Inequality*** components that tests for inequality; where $[\Sigma A_i]$ is connected to Input Node A and $[A_u]$ is connected to Input Node B. The Output Node marked [=] provides a Boolean output; where "1" means that A = B and "0" means the $A \neq B$. The Output Node marked [=] is conneted to the ***Stream Filter*** component that imposes a certain penalty in the case overlapping. A More detailed description of the ***Stream Filter*** component is provided in section 3.2.2.

3.5.3.3 Verification and Validation of Method 2:

In order for Method 2 to be verified, it was tested in the following three cases: no overlapping, partial overlapping, and full overlapping. Figure 55-A demonstrates the Grasshopper® formulation of Method 2 showing no penalty due to the fact that there is no overlapping of objects. In the case of partial overlapping shown in Figure 55-B, the Grasshopper® formulation

outputs a penalty. In the case of full overlapping - inclusion - shown in Figure 55-C, the Grasshopper® formulation outputs a penalty. Thus, Method 2 is verified and validated.

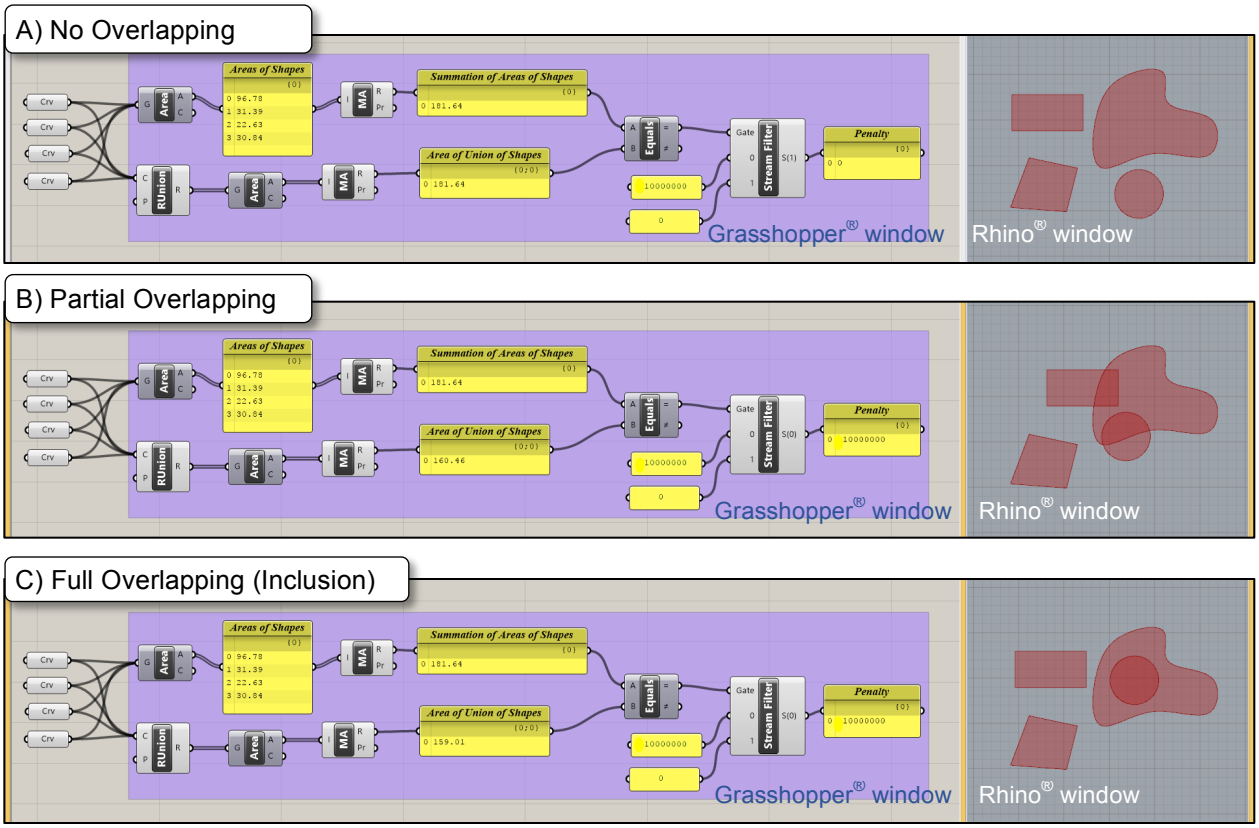

Figure 55: Formulation of Method 2 in overlapping detection algorithm

3.5.4 Method 3: Using the Area Union Approach – Visual Basic Replacing the *Stream Gate* Component

3.5.4.1 The algorithm of Method 3 in collision and overlapping prevention

The algorithm in this method is exactly the same as in Method 2. The only difference is in the means.

3.5.4.2 Formulation of Method 3 on Grasshopper®:

This method uses the same formulation in Grasshopper® as in Method 2 with only one difference. In this method, the ***VB.NET Scripting*** component is used to decide the penalty instead of the ***Stream Filter*** component used in Method 2.

The ***VB.NET Scripting*** component compiles and runs user specified Visual Basic code. By default there are two input parameters {x,y} and one output parameter {A}, all of which are of

type System.Object. Input and output parameter can be added, deleted and renamed through the component menu.

In this method, the ***VB.NET Scripting*** component is connected at its input node to the equality output of the ***Inequality*** component. The Visual Basic code is written in the component as an if-then-statement as shown in Figure 56.

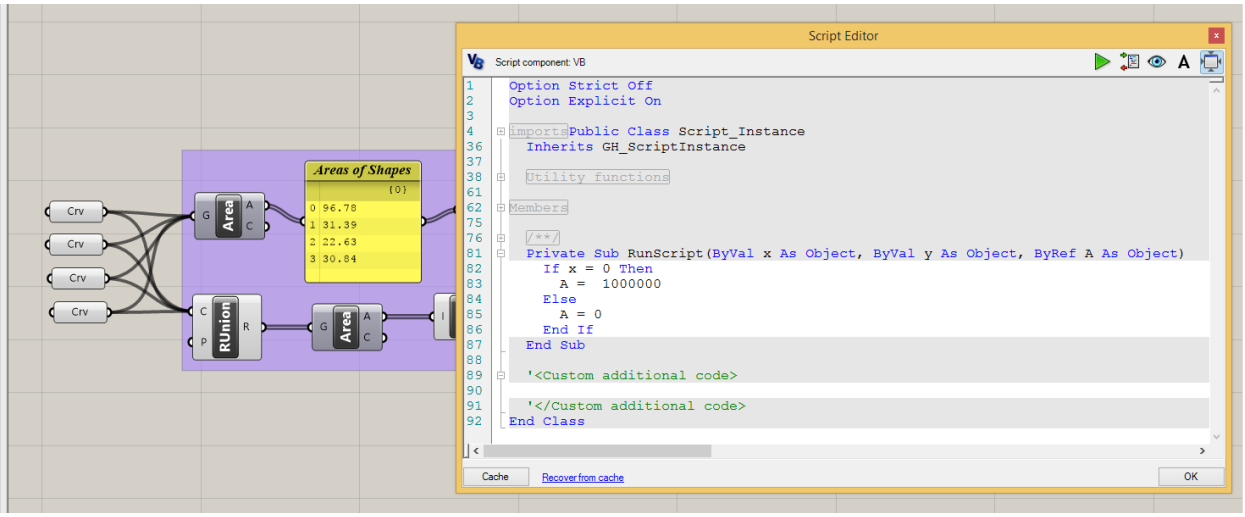

Figure 56: The Visual Basic code used in Method 3

3.5.4.3 Verification and Validation of Method 3:

In order for Method 3 to be verified, it was tested in the following three cases: no overlapping, partial overlapping, and full overlapping.

Figure 57-A demonstrates the Grasshopper® formulation of Method 3 showing no penalty due to the fact that there is no overlapping of the objects. In the case of partial overlapping shown in Figure 57-B, the Grasshopper® formulation outputs a penalty. In the case of full overlapping - inclusion - shown in Figure 57-C, the Grasshopper® formulation outputs a penalty. Thus, Method 3 is verified and validated.

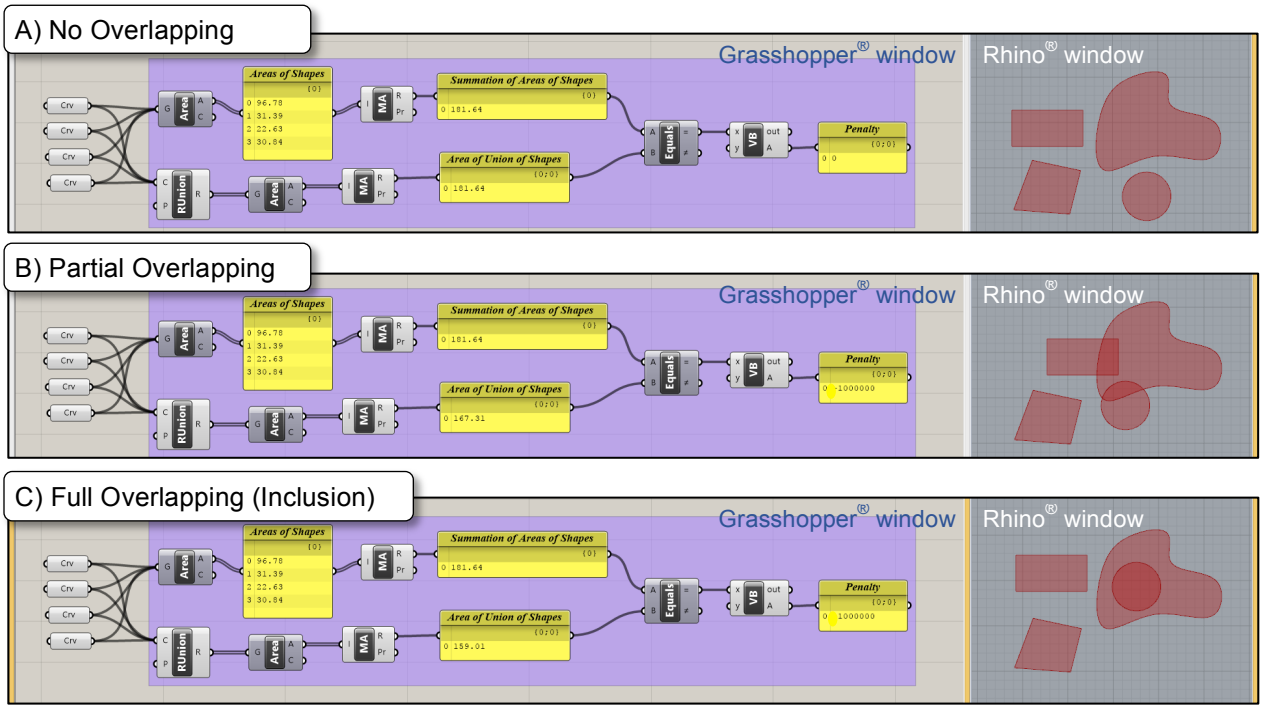

Figure 57: Formulation of Method 3 in overlapping detection algorithm

3.5.4.4 Drawback of method 3

The only drawback in this method is its relatively long processing time. Grasshopper® takes longer time (relative to Method 2) to run a model containing Visual Basic script because in each run it has to open the Visual Basic component and reads the code and compiles it to a format that matches the rest of the Grasshopper® code.

3.5.5 Comparison of methods in collision and overlapping detection

Method 1 utilizes the Grasshopper® built-in collision detection and inclusion detection modules. However, its coding is relatively complex and it gets more complicated as the number of analyzed objects increases. Method 2 uses a different approach; which is the area union approach. The advantage of this method is that it is not complicated and does not get complicated by increasing the number of objects. The algorithm and formulation is the same regardless what the number of objects is. Method 3 uses the same approach of Method 2 but with a minor change in the Grasshopper formulation by using Visual Basic to provide the penalty in case of overlapping. This method has the same advances of simplicity as Method 2. However it has a drawback that it takes relatively longer time to run.

Accordingly, the selected method for overlapping detection and prevention is **Method 2**.

3.6 In-site Constraint

3.6.1 Algorithm of the In-site Constraint

The objective of the In-site constraint is to ensure that facilities are inside the site boundaries. If a facility (object) is partially or fully outside the site boundaries in any of the model runs, the In-site constraint would result in a penalty that is added to the objective function. The resulting penalty is named the Out-of-Site Penalty. The In-site constraint uses the same methodology –the area union approach- used in the collision and overlapping prevention constraint; method 2 (Refer to section 4.4.3 for further demonstrations of the concept of the area union approach). The algorithm for the In-site constraint is as follows:

1. Form a surface from the site boundaries, in order for the site to be considered as an object.
2. Specify the area of the site [A_s].
3. Union the surfaces of the objects and the site into one surface.
4. Specify the area of the resultant of the union of surfaces [A_u].
5. There are only two possible outcomes:
    a. $A_s = A_u$ → means the objects are inside the site boundaries → No penalty.
    b. $A_s > A_u$ → means the objects are inside the site boundaries → Out-of-site penalty.

3.6.2 Formulation of the In-site Constraint on Grasshopper®

As shown in Figure 58, after the surfaces of the objects and the site are specified on Grasshopper®, the process is branched into an upper branch and lower branch then it is combined back again into a single branch comparing both to each other. The upper branch fetches the area of the site [A_s] using the ***Area*** component connected to the input node of the ***Surface*** component representing the site. The lower branch obtains the area of the union of objects. The ***Region Union*** component is connected at its input node to all of the ***Surface*** components representing the site facilities and the site area. The output is a single surface that is defined by the union of all of the connected surfaces. The area of the outputted surface is obtained and represented as [A_u]. The [A_s] and the [A_u] are then connected to the ***Inequality*** component that tests for inequality; where [A_s] is connected to Input Node A and [A_u] is connected to Input Node B of the ***Inequality*** component. The Output Node marked [=] provides a Boolean output; where "1"

means that A = B and "0" means the A ≠ B. The Output Node marked [=] is connected to the ***Stream Filter*** component that imposes a certain penalty in the case any of the facilities are out of the site boundaries. A More detailed description of the ***Stream Filter*** component is provided in Section 3.2.2.

3.6.3 Verification of the In-site Constraint Algorithm

In order for the algorithm to be verified, it was tested in the following two cases: case 1) all objects inside the site boundaries, and case 2) some objects are inside the site boundaries and others are partially outside the site boundaries. Figure 58-A demonstrates the Grasshopper® formulation of the In-site Constraint algorithm where all facilities are inside the site boundaries; thus no penalty was applied. In the case 2, as shown in Figure 58-B the Grasshopper® formulation outputs a penalty due to the fact that some facilities are outside of the site boundaries. Thus, the algorithm is verified and validated.

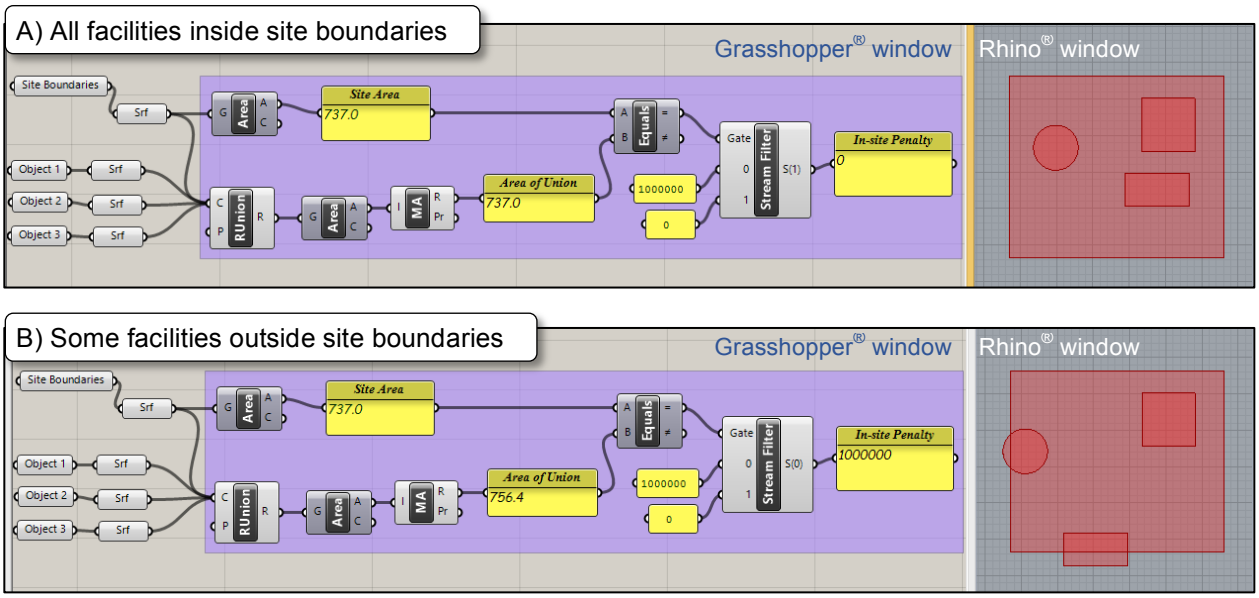

Figure 58: Formulation of In-site Constraint Algorithm

The use of In-site Constraint algorithm is essential in every site layout optimization module to be formulated using Grasshopper®. Since the in-site constraint using the area union algorithm is verified and validated, it can be used with any number of site facilities and any shape of site boundaries.

3.7 Area Constraint For Dynamic Geometrical Shapes

3.7.1 Algorithm of the Area Constraint

Dynamic objects (Sections 4.2.2, 4.2.3, and 4.2.4) change their form in every run in the optimization model. By changing the form, the area changes as well. Sometimes the site planner is bound to specify upper and lower limits for the shapes areas depending on other planning criteria. For example, a sand pile is modeled as a Dynamic Freeform (Section 4.2.3) where it shapes itself depending on the site conditions. However, the area of the pile of sand should have a range depending on the site. So, if the site planner needs $80m^2$ of land to be used a sand pile, he would set an upper limit of $90m^2$ and lower limit of $70m^2$. The model runs and provides a suitable shape for the sand pile with an area within the range set by the planner. The algorithm behind the Area Constraint is as follows:

1. Obtain the area [A] of the dynamic shape.
2. Set minimum [A_{min}] and maximum [A_{max}] limits of the area.
3. If $A > A_{min}$ AND $A \leq A_{max}$, then no penalty, Else, then apply penalty.

3.7.2 Formulation of the Area constraint on Grasshopper®

In Grasshopper®, the area of the dynamic object is obtained using the ***Area*** component connected to the output node of the ***Surface*** component representing the dynamic object. The ***Larger*** component and the ***Smaller*** components combined with the ***And*** component in the arrangement shown in Figure 59 to model the "If-then function" described in step 3 of the algorithm. The resultant from the ***And*** component is a Boolean (0 for not satisfying the area conditions and 1 for satisfying the area condition). The output node of the ***And*** component is connected to the input node of the Stream Filter component for the application of penalty in the case of not satisfying the area conditions.

3.7.3 Verification and Validation of the Area Constraint Algorithm

In order for the algorithm to be verified, it was tested in the following two cases: case 1) dynamic object area between the lower and upper limits (satisfying the area condition), and case 2) dynamic object area lower than the lower limit (not satisfying the area condition). Figure 59-A demonstrates the Grasshopper® formulation of the Area Constraint algorithm in case 1; thus no

penalty was applied. In the case 2, as shown in Figure 59=B the Grasshopper® formulation outputs a penalty due to the non-satisfaction of the area conditions. Since the formulation of the algorithm on Grasshopper® behaved as expected and ran without errors, the algorithm is verified and validated.

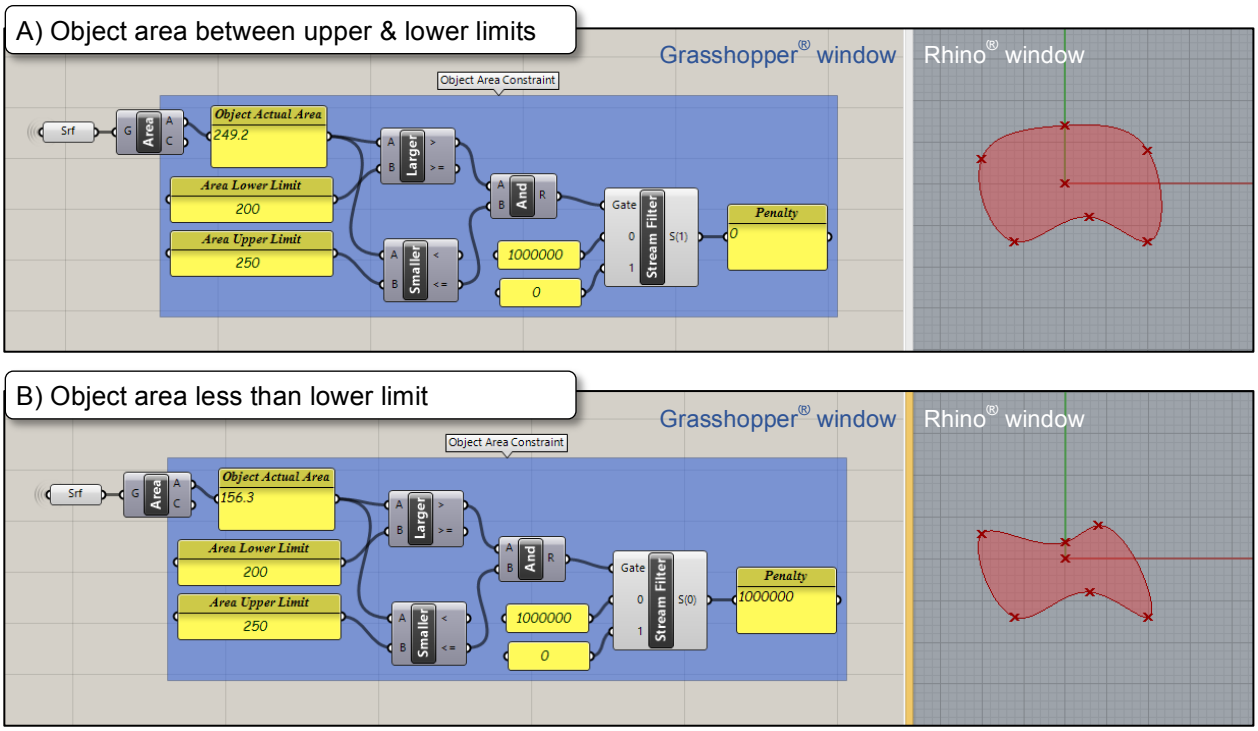

Figure 59: Formulation of Area Constraint Algorithm (Case 2: object area less than lower limit)

3.8 Proximity Relationships - Identification of Control Points of Objects for Distance Measurement

3.8.1 Background about the used proximity relationships

One important aspect in any site layout optimization model is the measurement of distance between facilities; because such distances are multiplied by the proximity weights to obtain the overall site layout score. In this research, 3 different proximity relationships between facilities are used: 1) Center-to-Center [CC], 2) Point-to-Point [PP], 3) Center-to-Point [CP], and 4) Side-to-Side [SS].

3.8.2 Center-to-Center [C-C] Proximity Relationship

The most simple proximity measure to use on Grasshopper® is the Center-to-Center; where the distance is measured between the centroids of the selected facilities. Figure 60 provides a demonstration of a simple Center-to-Center proximity relationship. An ***Area*** component is connected to the ***Surface*** component corresponding to each of the two objects. One of the outputs of the ***Area*** component is the coordinates of the centroid, marked as (C) in the output end of the component. After obtaining the coordinates of the centroids, measuring the distance between them can be made by one of two distance measurement techniques that are described in details in Section 3.9. In the demonstration shown in Figure 60, the distance measurement technique is the Direct (Cartesian) one for simplicity.

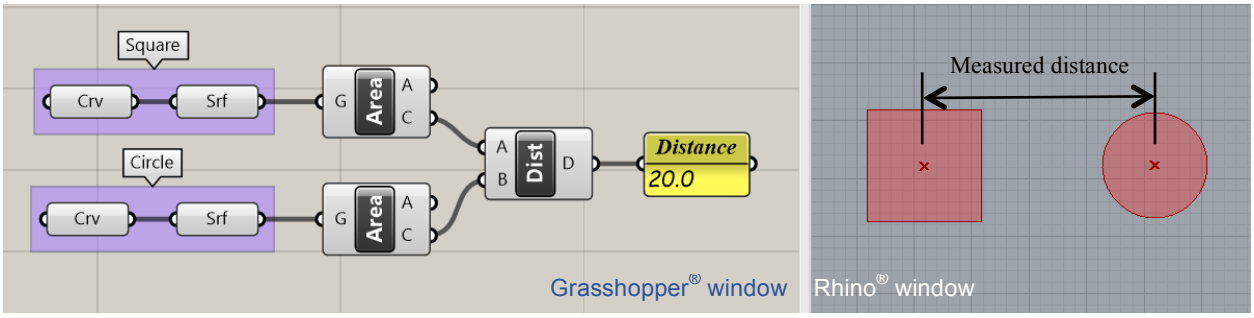

Figure 60: Formulation of the Center-to-Center proximity relationship

Figure 61 shows the alignment of the two objects previously shown in Figure 60 if they were connected to an optimization engine with the objective function to minimize the distance between their centroids without collision.

Center-to-Center proximity relationship is a fair abstraction of the proximity between site facilities, and can be used in models with no concerns in many of the cases; however, in some cases, measuring the distance between the centroids of facilities does not fully capture the real need of the site planner, in the sense that sometimes the significance of distance is abstracted between points that are not the centroids. As an example, the site planner might need the door (not the centroid) of a certain facility to be close to the door of another. So,

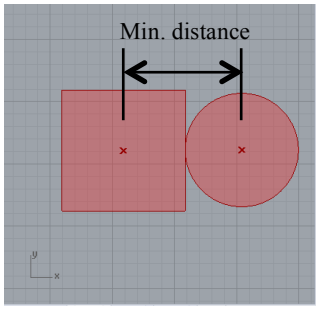

Figure 61: Min. [CC] distance between two objects

the centroidal distance is not the important measure here but the rather the distance between the doors; or in other words the Point-to-Point distance; because a door of a facility is modeled as a point on its outer perimeter.

3.8.3 Point-to-Point [P-P] Proximity Relationship

As mentioned in Section 3.8.2, sometimes the centroidal distances are not the proper representations for the actual modeling required by the site planner. The first step in the Point-to-Point proximity algorithm is to select the points on the objects' external perimeter, which is referred to in Grasshopper® as point on curve. In order to achieve that, a ***Point on Curve*** parameter is connected to the output nodes of each of the ***Surface*** components representing the objects. The output of the ***Point on Curve*** parameter is simply a point on the external parameter of the connected shape. The ***Point on Curve*** parameter is a slider with values ranging from 0 to 1, representing the whole perimeter of the object. So the user slides that slider until a point of satisfaction is reached, After obtaining the points that the distance is required to be measured from, the second step is to use either of the measuring distance techniques described in Section 4.9 to measure the distance between the selected points. In the demonstration shown in Figure 62, the distance measurement technique is the Direct (Cartesian) one for simplicity.

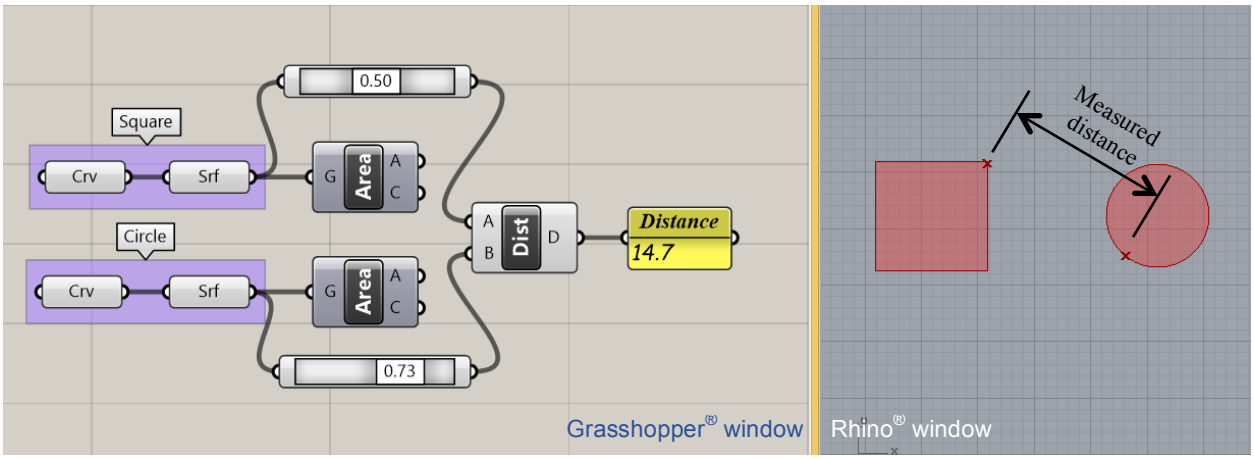

Figure 62: Formulation of the Point-to-Point proximity relationship

Figure 63 shows the alignment of the two objects previously shown in Figure 62 if they were connected to an optimization engine with the objective function to minimize the distance between their selected points [PP] without collision.

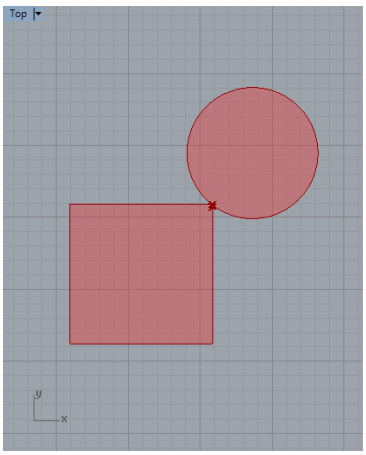

Figure 63: Min. [PP] distance between two objects

3.8.4 Point-to-Center [PC] and Center-to-Point [CP] Proximity Relationship

The PC and CP proximity relationship is where the distance is measured between the centroid of an object and the point on curve of the second object. The formulation does not differ from what is presented in Sections 3.8.2 and 3.2.3. The user uses the ***Area*** component to obtain the centroid coordinates of the first object and uses the ***Point on Curve*** parameter to obtain the point on the second object's perimeter. This is just a combination of the formulations used in Sections 3.8.2 and 3.2.3.

3.8.5 Side-to-Side Proximity Relationship

Sometimes, the planner does not only want to ensure a certain facility or object to be next to another object; he also might want to ensure that a certain side from object A (such as the entrance side of workshop) is close to a certain side from object B (such as an entrance side of a storage facility). For example in Figure 64, the centroids of objects A and B are close to each other. So if normal centroidal distance [CC proximity] was used, the model would probably not make any changes and consider this arrangement good enough. However, the site planner might want to specify that he/she wants the left side of object A to be close to the right side of object B; hence a model that understands this would make suitable changes in the alignment to satisfy the planner's criteria as shown in Figure 65. In this type of proximity relationship, the used algorithm for formulation is the same as used in the Point-to-Point proximity relationship algorithm; except that here, the user specifies 3 points in each object. So, the ***Surface*** component representing object A has 3 ***Point on Curve*** parameters connected to it to generate 3 points on

the selected side of the object (Points P_{A1}, P_{A2}, and P_{A3}). Same goes for object B, the three formed pints are P_{B1}, P_{B2}, and P_{B3}. Three distances are then measured: 1) Distance between first point of object A and first point of object B [D_1], 2) Distance between second point of object A and second point of object B [D_2], and 3) Distance between third point of object A and third point of object B [D_3]. The average of the 3 distances [$A_{D1,2,3} = (D_1+D_2+D_3)/3$] is obtained using the ***Average*** component as shown in Figure 64. The number outputted from the average component is the number that is entered in the inter-facility-distance matrix (D) (See section 4.10).

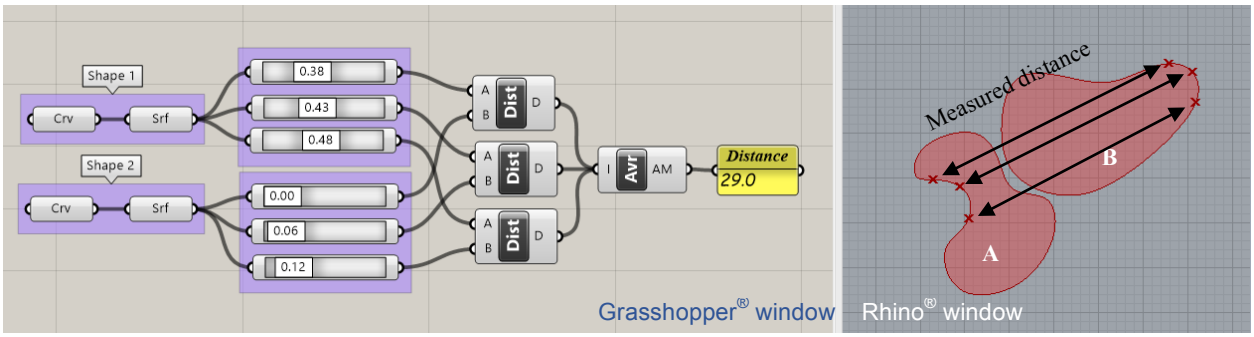

Figure 64: Formulation of the Side-to-Side proximity relationship

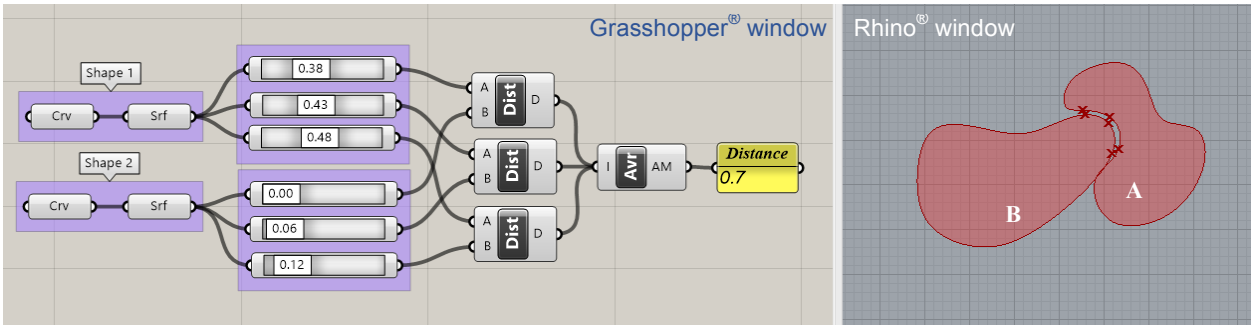

Figure 65: Min. [SS] distance between two objects

Figure 65 shows the alignment of the two objects previously shown in Figure 64 if they were connected to an optimization engine with the objective function to minimize the distance between their selected sides [SS] without collision.

3.9 Distance Measurement Techniques

3.9.1 Introduction to the Used Distance Measurement Techniques

After the user specifies the "snipping" points, meaning the points that the distances are measured in between, it is essential that distance measurement technique to be specified. This research provides two possible distance measurement techniques: 1) Direct (Cartesian) distance , and 2) Shortest Walk distance. The Cartesian distance between two points is the shortest straight-line distance between any two points. The Shortest Walk distance is the distance between two points while taking into consideration a certain path that avoids colliding with obstacles.

3.9.2 Direct (Cartesian) Distance

The Cartesian distance is the mostly used distance measurement technique by many of the previous researchers. Its common spread comes from its simplicity. The Cartesian distance between any two points with known coordinates is calculated by this simple formula:

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2} \quad \text{..... [Eqn. 35]}$$

Where,

d: distance between the two points
Δx: difference between the x-coordinates of the two points
Δy: difference between the y-coordinates of the two points
Δz: difference between the z-coordinates of the two points

On Grasshopper®, calculating the Cartesian distance between any two points is very simple by defining the points and using the ***Distance*** component connected to the two points as shown in Figure 66. So, after the user defines the points using either of the mentioned proximity relationships stated in Section 3.8, he connects between the defined points using the ***Distance*** component.

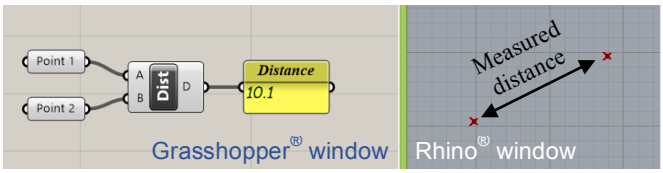

Figure 66: Cartesian Distance measurement on Grasshopper®

3.9.3 Shortest Walk Distance

The Shortest Walk distance in this research is defined as the minimum distance between two points; where it follows a certain path (walk) that does not collide with obstacles. This distance measure is more realistic since it mimics the behavior of man, material, and equipment movement on site. Providing an algorithm for measuring the Shortest Walk distance is not a simple task as the Cartesian distance. A 7-seps algorithm is provided in this research for measuring the Shortest Walk distance on Grasshopper®. The main 7 steps of the algorithm formulation are shown in Figure 67 (upper part of the figure shows the full formulation and components connections on Grasshopper® while lower part of the figure shows a visual demonstration of the 7 steps).

Measurement of the Shortest Walk distance on Grasshopper® is made possible by the component plug-in named "Shortest Walk" developed and released by McNeelEurope in 2011 (www.food4rhino.com). The "Shortest Walk" component measures the Shortest Walk distance between any two points by analyzing the available paths (walks) and selecting the one that has the shortest distance of walking. The plug-in only understands lines as paths (walks), and the walks (network of lines) have to be pre-specified. So the most convenient way of creating the walks is by using network grids. In a grid, there are many possibilities of connecting between any two points; the plug-in component counts all the possible paths (walks) and calculates their distances and specifies the walk with the shortest distance. To enable to the plug-in to avoid obstacles, which are all the objects in the site layout, the objects have to be off the grid; meaning that there are no possible walks that pass inside them. The developed algorithm for incorporating the "Shortest Walk" component plug-in to site layout problems is made of 7 steps that are necessary for the automation of the model and for making a generic one (meaning that it can work on any shapes of the objects and the site, and it can work while the solver is running by re-setting the grids and performing all steps to recalculate the Shortest Walk distance in every run. The following paragraphs provide detailed explanation of the algorithm steps and closer screenshots from the Grasshopper® window showing the corresponding formulation of each step.

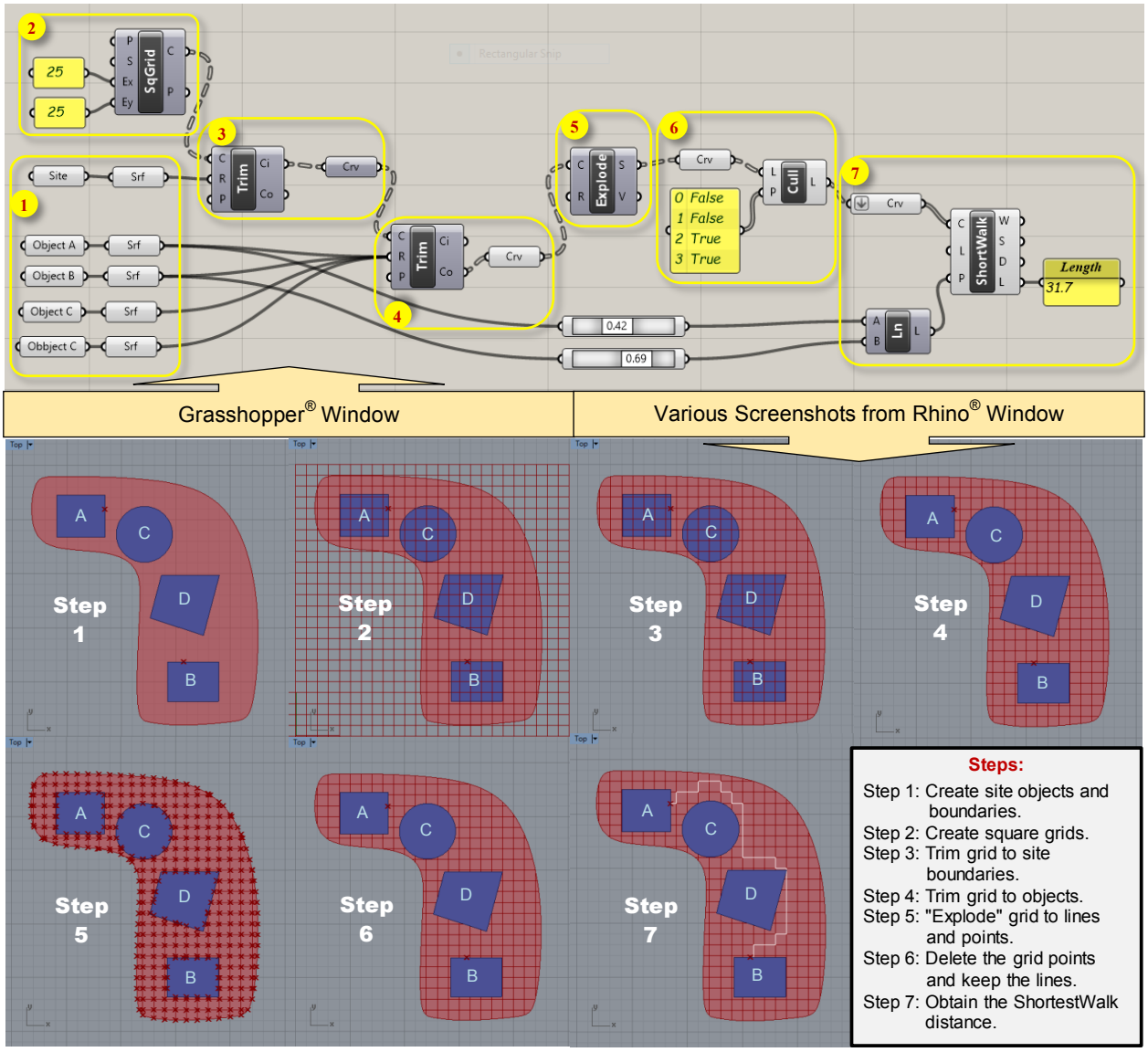

Figure 67: Steps of formulated the Shortest Walk distance measurement algorithm on Grasshopper®

Step 1 – Creating site objects and boundaries: This step is an integration of many steps that result in the creation of the shapes and sizes of both the site boundaries and the facilities (objects) discussed in Section 3.2. Since the this section is for the purpose of demonstrating the Shortest Walk algorithm and not the forming of the site objects, both the construction site boundaries and the shapes of the objects are formed on Rhino® and imported to Grasshopper® in a summarized form as shown in Figure 67. The site is treated as object in the sense that is has its own surface as all of the rest of the site objects.

Step 2 – Creating square grids: The square grids form the possible walks that will be used by the Shortest Walk component plug-in in its analysis. The Shortest Walk plug-in does not consider any paths outside of the grid lines since it only consider walks formed by connected lines or curves. There are several other grid types such as rectangular, circular, and pentagonal grids. The grids used in this research are square grids. But, the model is very generic and it is very simple to change the type of used grid. The square grids are formed using the ***Square Grid*** component on Grasshopper®. In this component, the user has the flexibility of setting the plane of the grid (preset as the XY plane), the size of the grid cell (set t 1x1m), number of grids in the x-direction (set as 25 in the figure), and the number of grids in the y-direction (set as 25 in the figure). The output node marked as "C" in the ***Square Grid*** component provides the grid cell outlines (gridlines) as one whole curve definition. When forming the grids, the grids should embrace the site boundaries; in other words, the borders of the gridlines should surround the site borders from all directions.

Step 3 – Trimming the grids to the site boundaries: All grids outside the site boundaries are not needed, so they are trimmed using the ***Trim With Regions*** component that trims a curve definition (lines or curves) with one or more regions (surface definitions). The input node marked as "C" in the ***Trim With Regions*** component is where the curve-to-trim is connected; so it is connected to "C" output node of the ***Square Grid*** component. The input node marked as "R" in the ***Trim With Regions*** component is where the regions-to-trim-with are connected; so it's connected to the output node of the ***Surface*** component representing the site. The output node "Ci" is the split curves inside the regions and the "Co" is the split curves outside the regions. Since the external part of the grid is the one that needs to be removed, the "Ci" node is connected to a ***Curve*** parameter, and the preview original grid along with the preview of the trimming tool are turned off, so only the grids inside the site boundaries are shown.

Step 4 – Trimming the grids to the objects: After ensuring that the grids are inside the site boundaries (step 3), step 4 is concerned about ensuring that the grids are outside of the site facilities (objects). The resulting grid from step 3 is trimmed using the ***Trim With Regions*** component. This time the input node marked as "C" in the ***Trim With Regions*** component is connected to the ***Curve*** parameter representing the resultant gridlines of step 3, and the input node marked as "R" in the ***Trim With Regions*** component is connected to the output nodes of

the ***Surface*** components representing the site facilities (objects). This connection is shown in the upper part of Figure 67. This time the output node "Co" is the one that is connected to a ***Curve*** parameter and all of the other previous components representing grids are hidden (preview turned off) to preview only the grids between the objects and the site boundaries.

Step 5 – "Exploding" the grids into separate lines and points: So far, the grids are treated as one whole curve definition. The Shortest Walk component plug-in takes separate connected lines as input, so the grid has to be "exploded". The ***Explode*** component explodes the curve definition of the grid into smaller segments; the resultant is a series of lines and points.

Step 6 – Delete the grid points and keep the lines: The lines and points resulting from the ***Explode*** component (step 5) can not be used as inputs to the Shortest Walk plug-in yet; the points have to be filtered out so that only the lines forming the grids are outputted. The ***Cull Pattern*** component is used for this purpose. The function of a Cull Pattern component is to cull (remove) elements in a list using a repeating bit mast that is defined as a list of Boolean vales. The bit mask is repeated until all elements in the data list are evaluated. For example, if an input list = {A, B, C, D, E, F, G, H, I} and the bit mask pattern = {True, True, False, False}, then the output list = {A, B, E, F, I}. Since the list outputted by the Explode component is arranged in the following order {line, line, point, point, line, line, point, point …}, the bit mask to remove the points is {True, True, False, False}. The culled list containing the grid "lines" is stored in the ***Curve*** parameter connected to the output node of the ***Cull Pattern*** component.

Step 7 – Obtain the Shortest Walk distance: The ***Shortest Walk*** distance plug-in component takes 3 inputs; only 2 are used in the scope of this research. The inputs are "C" the curves group and "P" the line from the start to the end of the path. The curves group is the group of lines forming the grids; which is represented in the ***Curve*** component of step 6. For the "P" input, a line has to be formed between the two points that the distance is measured between. The line is formed using the ***Line (Ln)*** component. The formed line can be hidden so it does not disrupt the user's view. The ***Shortest Walk*** component draws the path representing the Shortest Walk distance.

The 7-stepped algorithm ensures both a generic and flexible stand-alone module that can be added to any site layout optimization problem on Grasshopper®. The algorithm adjust the grids

automatically in each run because it repeats all of the steps in each run. However, using this algorithm for distance measurement adds to the overall computing time if used in an extensive site layout optimization model. So it is recommended to use Cartesian distances unless it was required to use the Shortest Walk distances.

3.9.3.1 Effect of grid size on the Shortest Walk distance

As the grid size decreases, the better the accuracy of the obtained distance. Figure 68 shows the Shortest Walk between two points in three cases of grid sizes while maintaining all other variables constant. Larger grid size results in unnecessary distances in the walks. So as the number of size of grids decrease, the Shortest Walk distance decreases (becomes more efficient and accurate), and actually the path itself changes to a shorter path as shown in the figure.

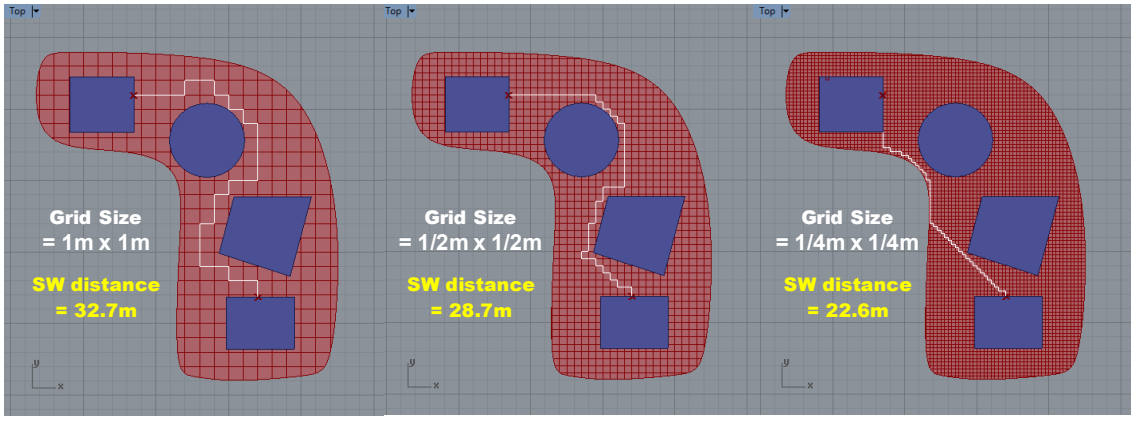

Figure 68: The effect of grid size on the Shortest Walk distance

3.9.4 Comparison between Cartesian and Shortest Walk distances

Figure 69 provides a comparison between the Cartesian distance and the Shortest Walk distance in various situations. As expected, the Cartesian distance is smaller than the Shortest Walk distance in all cases. It can also be noticed that the Shortest Walk distance algorithm can work in cases where the measurement points are inside the objects (such as centroids) and not on the grid. In such cases, the Shortest Walk path starts from the point on the grid that is nearest to the point of the measurement point such as in cases 3 and 4. In Figure 69, the white line resembles the Shortest Walk distance while the green line resembles the Cartesian distance.

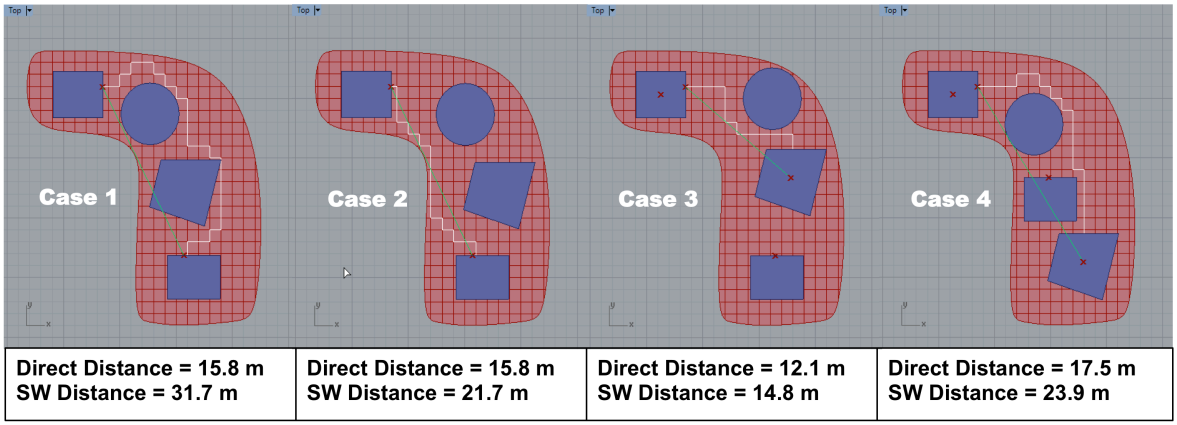

Figure 69: Comparing the Cartesian distance to the SW distance

There are no certain rules for preferring to use the Shortest Walk distance or the Cartesian distance in the model; both options are available in the developed model and the user is left to choose any of them. The model allows the user to choose a distance measurement technique for each single distance between the different facilities; which provides a great flexibility. Also, the user might prefer to generalize a distance measurement technique on all distances; the model allows for that too.

3.10 Buffer Zones

Some site facilities are allowed to be laid next to each other without a buffer space and others require some space around them. If a square-shaped facility with dimensions of 4m x 4m requires a buffer zone around it of 1m, and the required buffer zone is a must; meaning that it absolutely not allowed for any reason to be put next to it with spacing less than this 1m, then that facility shall be modeled as a 5m x 5m square and the collision prevention module shall ensure there is no collision. However, if the buffer zone is not a must; meaning that it is only preferred to leave a spacing of 1m (or any other dimension) unoccupied around the facility with putting a small penalty for any other facility that collides with this buffer zone, then the "Buffer Zones Module" is recommended to be used.

The Buffer Zones module is an algorithm developed on Grasshopper® in this research to model buffer zones around facilities with user-defined thickness, while adding a soft constraint for colliding with these buffer zones. The soft constraint is part of the algorithm where a penalty is calculated base on the area of collision with the buffer zone. So, the optimization model might

allow for facilities overlapping with buffer zones of other facilities if it is for the better of good of the whole objective function.

The Buffer Zones module is made of two sub-modules: 1) Creation of buffer zones, and 2) Setting the soft constraint penalty. The creation of buffer zones on Grasshopper® is based on 3 steps demonstrated in Figure 70. The first step is to create the shape representing the facility using any of the algorithms stated in Section 3.2. The second step is forming an offset curve definition on the external end of the original shape using the ***Offset*** component. The user can determine the offset distance by writing it in a ***Panel*** parameter and connecting it to the "D" input node of the ***Offset*** component. The third step is forming a surface between the original shape boundaries and the offset curve using the ***Ruled Surface*** component that creates a surface between any two curves.

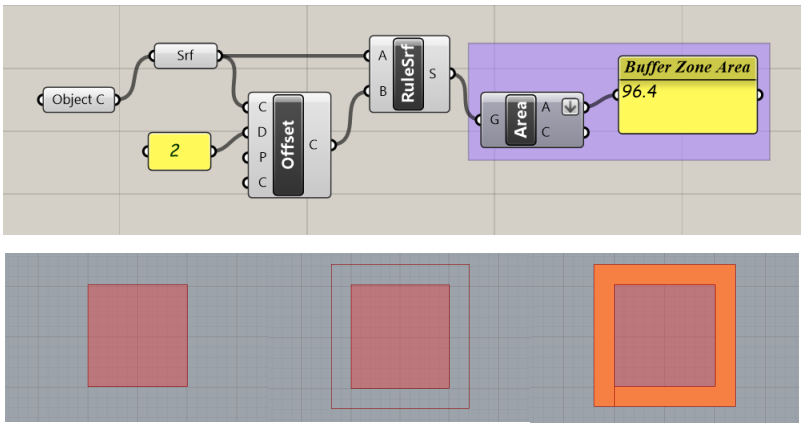

Figure 70: Steps of creating buffer zones on Grasshopper®(Up) and the corresponding demonstration of the three steps on Rhino® window (Down)

Setting the buffer zone soft constraint penalty on Grasshopper® is similar to setting the collision and overlapping constraint (method 2) described in Section 3.5.3 with two differences: 1) The unioned surfaces are the buffer zones, not the facilities, and 2) the penalty is not a static penalty, but rather a dynamic one depending on the area of overlapping using the following function:

$$Buffer\ Zone\ Penalty\ =\ Overlapped\ area \times Unit\ Penalty \quad \ldots\ldots [Eqn.\ 36]$$

where the unit penalty is a user-defined penalty per $1m^2$ of overlapping with buffer zones. Figure 71 demonstrates the Buffer Zone penalty algorithm on Grasshopper® using a unit penalty of $500/m^2$. The user can modify the penalty depending on his preferences.

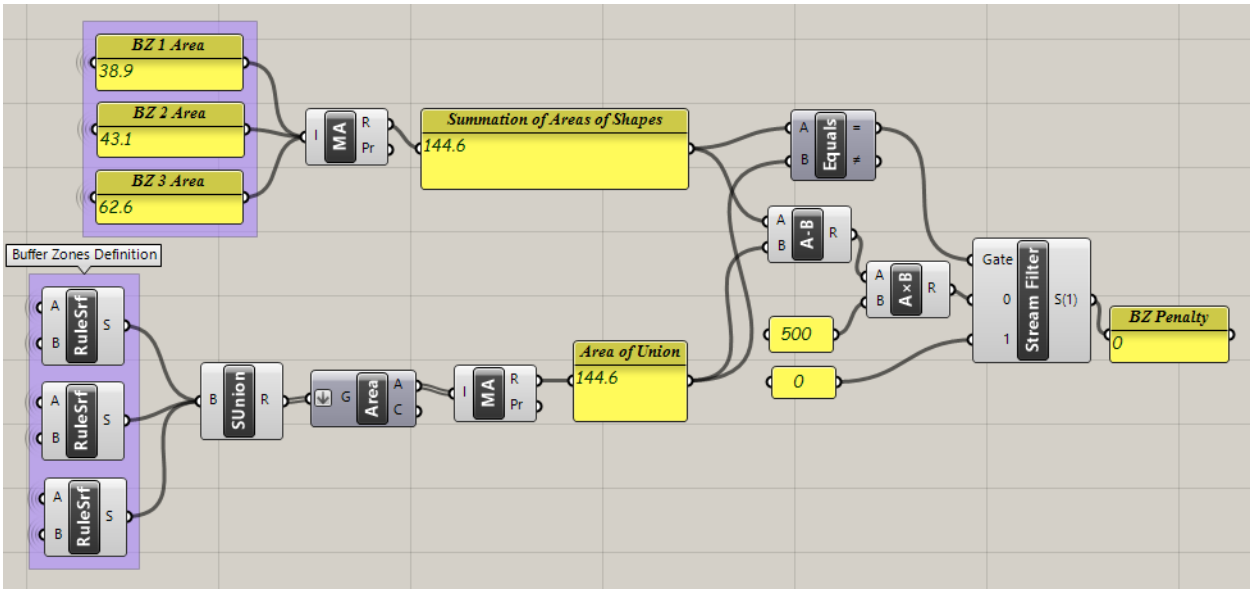

Figure 71: Formulation of the Buffer Zones Penalty algorithm on Grasshopper®

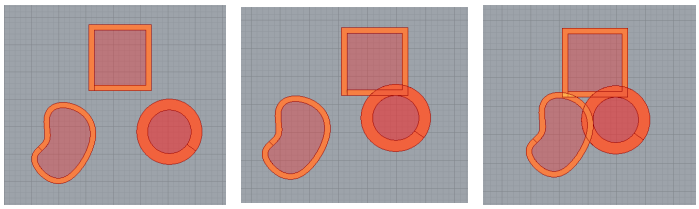

Figure 72: Demonstrating the Buffer Zones soft constraint

To demonstrate the dynamic nature of the buffer zone soft constraint, a sample layout of 3 facilities with different shapes were modeled and shown in Figure 72. In part A of the figure the resulting penalty is 0 since there was not overlap between any of the buffer zones. In part B, the penalty was 3,400 added to the overall model score due to some overlapping. In part C, the overlapping was more severe, accordingly the penalty increased to 10,700. The penalty and the overall score do not have a specific unit.

Based on tests, it shall be noted that using the Buffer Zones module almost doubles the processing time of the model. Accordingly it is not recommended to be used at this current stage, unless it is strongly required by the user, as it requires further development to minimize its processing time. It also shall be noted that this module works on any of the geometries stated in this research (Section 3.2).

3.11 Formulating the Objective Function

3.11.1 Proximity Weights

Effective placement of facilities within the site facilitates the movement of resources, or basically the interactions, among the facilities. Such interactions are referred to as the proximity relationships among the facilities (Hegazy and Elbeltagi, 1999). The proximity relationships represent the preference of the site planner in locating the facilities near or far from one another. There are quantitative and qualitative methods for specifying these proximities. The quantitative methods consider the actual transportation time and cost of personnel and materials between the facilities. The qualitative method considers subjective numerical proximity weight to represent the desired inter-facility proximity (close or far). The scope of this research is not focused on investigating the most suitable method. The developed model suits both methods as it is generic. The user has the flexibility to input the proximity weights that he desires. For demonstration purposes in this research, the qualitative method is used. The method used in this research is the qualitative method. If the two facilities are required to be close to each other, the proximity weight between them would have a high value. As stated in the literature review, several scales have been adopted to represent the proximity weights representing the qualitative importance of closeness of objects in numerical values. The proximity relationship weights used in this research are provided in Table 12. It is to be noted that there is no theoretical background for the exact numerical values given to each closeness level except that the used values are almost the same as the ones used in various previous research efforts. Since the objective function is to minimize the overall score (which is the multiplication of the proximity weight and the inter-facility distances), as the proximity weight increases, the model targets to decrease the distance to minimize the score. Accordingly, in this research, any two facilities with undesirable relationship would have a proximity weight of a negative value, so as the distance between those two facilities increases the smaller the score. Accordingly, the model would focus on both minimizing the distances between facilities with strong relationships and maximizing the distances between the facilities with undesirable relationships.

Table 12: Proximity Relationship Weights

Desired relationships between facilities	Proximity Weight (W)
Absolutely necessary	81
Especially important	37
Important	9
Ordinary closeness	3
Unimportant	1
Undesirable	-9

3.11.2 Objective Function

The inter-facility distance is measured (explained in Sections 4.8 and 4.9) and organized in a matrix named the inter-facility distance matrix (D). Matrix D is then multiplied by Matrix W in a scalar multiplication as shown in Figure 73, then all of the numbers in the resulting matrix are added to obtain the proximity score.

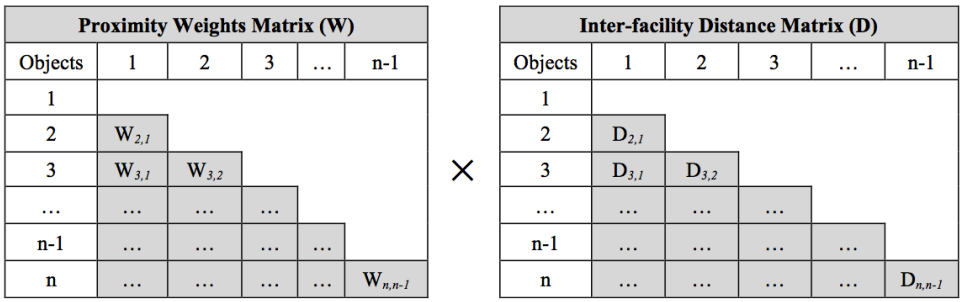

Figure 73: Multiplication of W and D matrices

The mathematical formulation of the proximity score is as follows:

$$Proximity\ Score = Total\ Travel\ Distance = \sum_{i=2}^{n} \sum_{j=i-1}^{n-1} D_{ij} W_{ij} \quad \ldots [Eqn.\ 37]$$

Where,

D_{ij} = *distance between facilities i and j.*
W_{ij} = *desired proximity weight value between facilities i and j.*

For static site layouts, the objective function is to minimize the summation of the proximity score and all the penalties corresponding to the constraints. Since the penalties are very large positive values and the model objective is to minimize the objective function, the model will produce

many runs to reach a near-optimum solution where all the penalties are zeros and at the same time the proximity score is minimized. A small proximity score means that the site facilities are laid in accordance to the preferences of the user (facilities that are required to be close to each other are actually close to each other and vice versa). In tight construction sites, the number of valid solutions (where all penalties are zeros) is relatively small, so the efforts of the optimization would be to find valid solutions rather than optimal solutions. In a way, valid solutions in a tight site are some kind of near-optimum solutions. However, if it is possible to start the solver with a valid solution, the solver would concentrate its efforts to find better solutions so there are better chances of reaching more optimum solutions. The objective function used for static site layouts is shown in Equation 38.

$$Objective\ Function = minimize \sum_{i=2}^{n} \sum_{j=i-1}^{n-1} D_{ij} W_{ij} + P_c + P_s + P_a + P_{bz} \quad \ldots\ldots\ldots [Eqn.\ 38]$$

Where,

D_{ij} = distance between facilities i and j.
W_{ij} = desired proximity weight value between facilities i and j.
P_c: collision penalty
P_s: out-of-site Penalty
P_z: area penalty
P_{bz}: buffer zones penalty

For dynamic site layouts, each phase is treated as a separate static optimization problem while taking into consideration the relocation cost in the objective function. The used approach for the dynamic layout optimization in the model is the Critical Phase approach; which is based on the prevalence of a phase that is identified a critical phase for its higher transportation costs or for any other logical reasons that might entail to the site planner. Dynamic optimization proceeds in backward chronological order for all phases preceding the critical phase and in forward chronological order for all phases succeeding the critical phase (Othman et al., 2003). In the critical phase, the objective function is to be calculated using Equation 38, while in all other phases the objective function is to be calculated using Equation 39, where the relocation is included and the layout of each preceding phase is considered. It shall be stated that other dynamic optimization approaches could be easily incorporated in the model.

$$Objective\ Function\ of\ Phase\ a = minimize \sum_{i=2}^{n} \sum_{j=i-1}^{n-1} D_{ij} W_{ij} + P_c + P_s + P_a + P_{bz} + \sum_{i=1}^{n} RC_i$$

Where,

D_{ij} = distance between facilities i and j.
W_{ij} = desired proximity weight value between facilities i and j.
P_c: collision penalty
P_s: out-of-site Penalty
P_z: area penalty
P_{bz}: buffer zones penalty
RC_i: relocation cost of facility i from phase a to phase a-1

[Eqn. 39]

In dynamic site layouts, after calculating the score of each phase separately (using the objective functions in Equations 38 and 39), the weighted average score is calculated using Equation 40; where the average score is the representative score of the overall layout efficiency because it takes the phases' duration into consideration.

$$Weighted\ Average\ Score = \frac{\sum_1^2 S_i D_i}{\sum_1^2 D_i}$$

Where,

S_i = site layout score of phase i.
D_i = duration of phase i.

[Eqn. 40]

3.11.3 Formulation of the Objective Function on Grasshopper®

Modeling the objective function on Grasshopper requires 5 sequential steps (modules):

1. Module 1: Setting the Proximity Relationship and Distance Measurement Techniques
2. Module 2: Proximity Weight Matrix (W)
3. Module 3: Inter-facility Distance Matrix (D)
4. Module 4: Proximity Score Calculation ∑W×D
5. Module 5: Objective Function Calculation

Each module of them depends on one or more of the preceding modules. Such dependence is through a connection between output nodes and input nodes of different components and parameters. Figure 74 provides an overall look of the 5 modules and how they're connected in an example of a site with just 4 facilities for demonstration only. Details of forming each module are provided in the following paragraphs. In a full-scale site layout optimization problem with large number of temporary facilities, the 5 modules are formulated in the same arrangement and logic as described in this chapter.

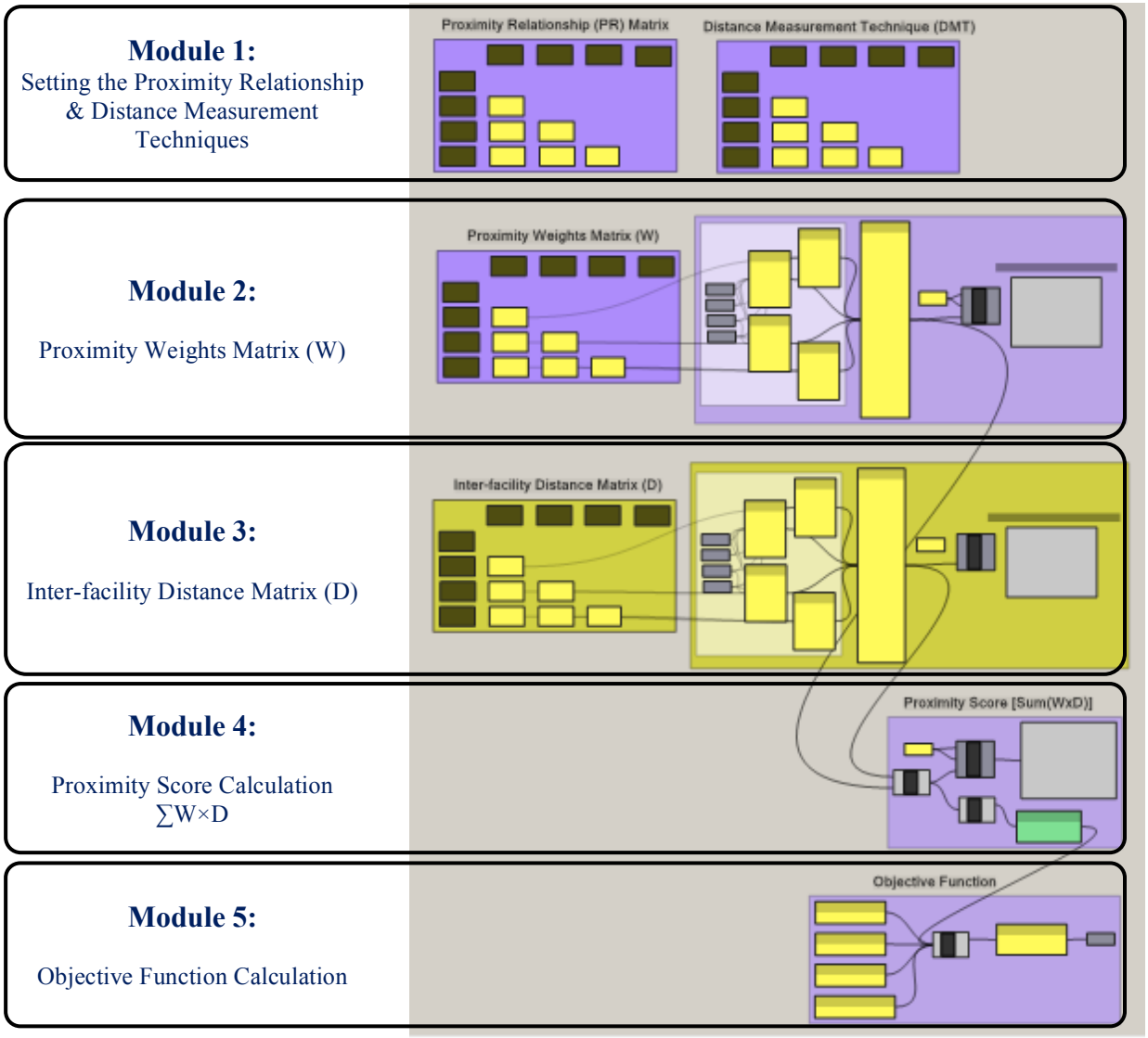

Figure 74: Formulation of the five modules of the objective function on Grasshopper®

3.11.3.1 Module 1: Setting the Proximity Relationship & Distance Measurement Techniques

The function of this module is just for the user to identify the proximity relationships (Center-to-Center, Point-to-Point, Center-to-Point, or Side-to-Side) and the distance measurement techniques (Cartesian or Shortest Walk) between the different facilities. There are no calculations made in this module. It just acts as an information module, because it will affect the way the distance is measured between the facilities. Formulating this module on Grasshopper® is nothing more but using a series of ***Panel*** parameters organized in the shape of a matrix. Each cell is a separate ***Panel*** parameter with the values written in it as shown in Figure 75 (note: the figure provides a demonstrative example for a site layout with only 4 facilities).

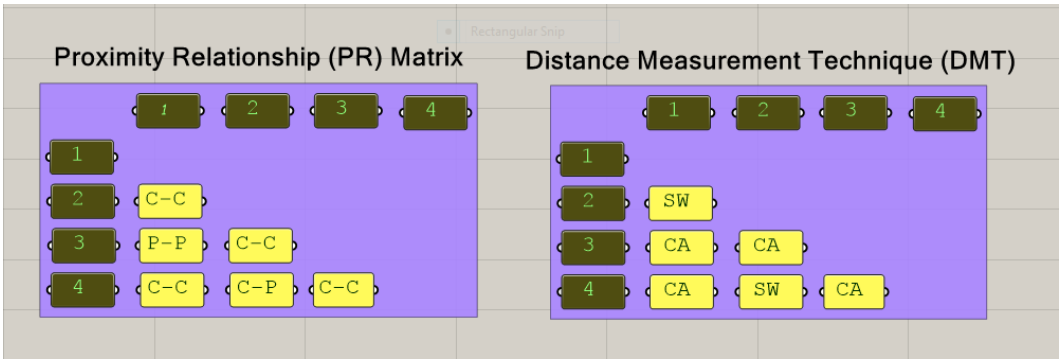

Figure 75: Formulation of Module 1 of the Objective Function on Grasshopper®

3.11.3.2 Module 2: Proximity Weights Matrix (W)

The function of this module is for the user to input the values for the desired proximity weights between facilities and for the model to convert these values from the matrix view that is user-friendly to a form that can be used in model operations. Because there is no direct matrix input tool in Grasshopper®, the user-friendly part of the proximity weights is formed using a set of ***Panel*** parameters; a separate ***Panel*** parameter for each cell in the matrix. The left side of Figure 76 shows the matrix form of the proximity weight; which is the part that the user inputs the desired proximity weights. In order to form this into a matrix form that Grasshopper® understands, the numbers have to be put in a list (named All Cells (W)) and then converted into a matrix from this list. The arrangement of the numbers in the list is by row. So a separate list of each row is formed then they are combined to form the list representing the whole matrix. The list for each row is formed using the ***Panel*** parameter through a series of connections to the numbers in the user-friendly matrix on the left as shown in Figure 76. The connections between the rows list and the user-friendly matrix are dimmed out in the figure in order not to cause confusion; however the rest of the connections in the figure are not dimmed out because they are simple and do not cause confusion to readers. After forming the list with all the values of the proximity weights matrix organized by row, a matrix is formed using the ***Construct Matrix*** component. The ***Construct Matrix*** component requires 3 inputs: 1) R: number of rows, 2) C: number of columns, and 3) V: the value of the matrix. In the demonstration shown in Figure 76, the number or rows and columns is 4; and the value of the matrix is all the numbers in the list. To summarize Figure 76, on the left is the user-friendly matrix where the user inputs values for the inter-facility proximity weights. On the right side is the matrix that the software understands and has the ability to perform mathematical operations on it. It has the same values as the user-

friendly matrix. In the middle of the figure is the algorithm that converts the user-friendly matrix into a software-understandable matrix.

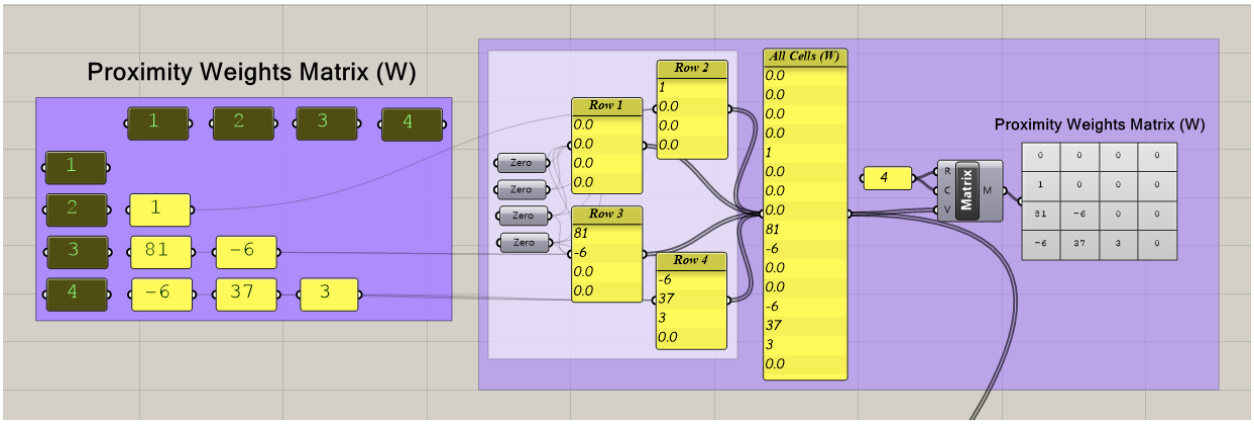

Figure 76: Formulation of Module 2 of the Objective Function on Grasshopper®

3.11.3.3 Module 3: Inter-facility Distance Matrix (D)

The function of this module is to output the matrix showing the distances between the facilities that are used in the objective function calculation as explained in Section 3.10.2. The module starts by fetching the inter-facility distances from the distance measurement algorithm between each and every facility and presenting them in ***Panels*** forming the shape of a matrix as shown on the left side of Figure 77.

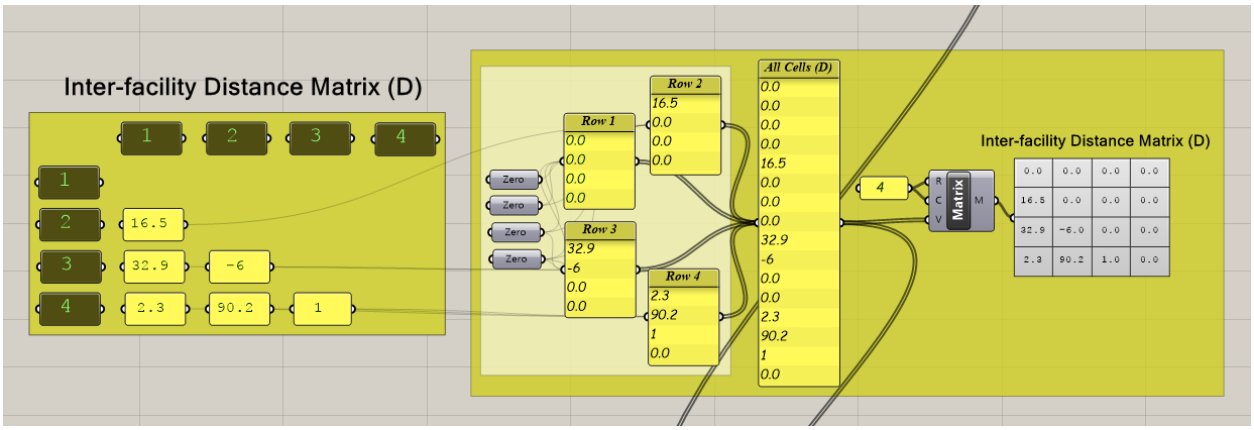

Figure 77: Formulation of Module 3 of the Objective Function on Grasshopper®

The values in the panels are no inputted by the user, but rather calculated by the model in each run. So in each run, the values in the ***Panels*** will be different due to the movement of the site facilities. Module 3 then translates these values into a matrix form that is understood by the

software by using the same steps followed in module 2. The connections between the rows list and the user-friendly matrix are dimmed out in the figure in order not to cause confusion; however the rest of the connections in the figure are not dimmed out because they are simple and do not cause confusion to readers.

3.11.3.4 Module 4: Proximity Score Calculation $\Sigma W \times D$

The function of this module is to calculate the proximity score by multiplying the proximity weights matrix and the inter-facility distance matrix. To perform this mathematical operation, the ***Multiply*** component is used to multiply between two inputs. The first input is the list containing the values of the proximity weights matrix (The list named All Cells (W) in Figure 76). The second input is the list containing the values of the inter-facility distances matrix (The list named All Cells (D) in Figure 77). Since two lists are multiplied, the output of the ***Multiply*** component is also a list of the product of each two numbers from the list with the same order (first number from the first list multiplied by the first number of the second list and the second number of the first list multiplied by the second number of the second list and so on). The list showing the product is not needed to be shown. The numbers in that list need to be added together. Such operation is made using the ***Mass Addition*** component connected to the output node of the ***Multiply*** component. The output node of the mass addition is connected to the input node of a ***Panel*** parameter to view the proximity score. Formulation of module 4 is shown in Figure 78.

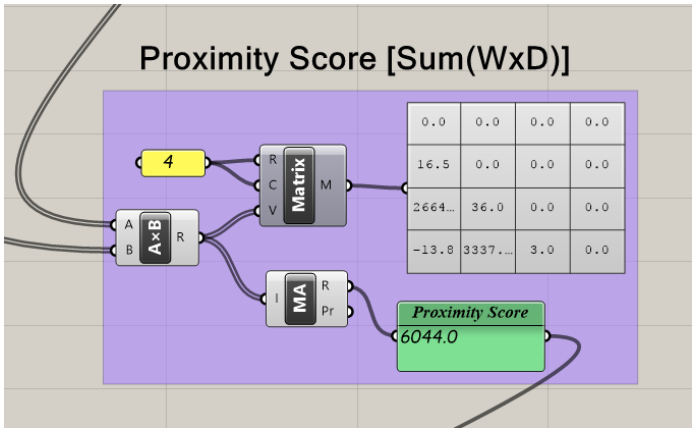

Figure 78: Formulation of Module 4 of the Objective Function on Grasshopper®

3.11.3.5 Module 5: Objective Function Calculation

The function of this function is to calculate the objective function; which is the proximity score in addition to the collision penalty (obtained from the collision prevention constraint algorithm discussed in Section 3.5), the out-of-site penalty (obtained from the In-site constraint algorithm discussed in Section 3.6), the area penalty (obtained from the area constraint algorithm discussed in Section 3.7), and the buffer zones penalty (Section 3.10) using the ***Mass Addition*** component. Figure 79 shows the formulation of module 5. After the objective function is calculated, it has to be connected to a ***Number*** parameter since Galapagos (the evolutionary optimization solver) only takes a ***Number*** parameter as an input for the objective function.

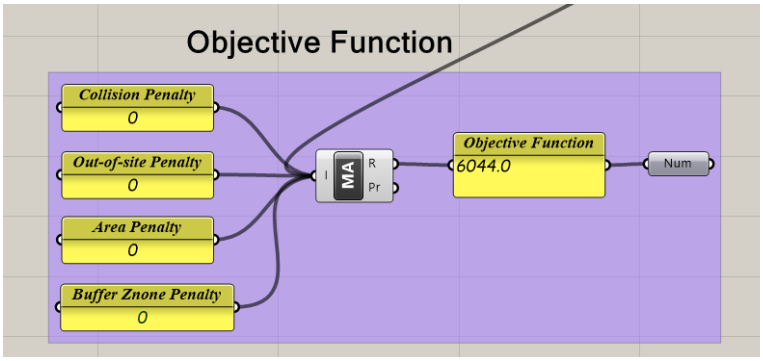

Figure 79: Formulation of Module 5 of the Objective Function on Grasshopper®

All of the 5 modules of the objective function formulation are for the purpose of reach the ***Number*** parameter at the end of module 5. This ***Number*** parameter is the element that is going to be connected to Galapagos as the objective function for the optimization process.

3.12 Optimization Using Galapagos:

After building the model (setting the site boundaries, facilities fata and variables, constraints, and objective function), Galapagos is used to run GA to find the near-optimum solution for the problem. Galapagos is used in Grasshopper® as a component or a parameter, with two input nodes: 1) Variables node, and 2) Objective Function node. Unlike other GA tools such as Evolver in Microsoft Excel®, there are no hard constraints in Galapagos. To overcome this limitation, the objective function includes very high values for penalties that would be modeled as hard constraints in other GA tools.

The variables node (also named as the Genomes node) is connected to all the Number Slider parameters representing the different variables in the model (facility movements, rotation, dynamic shapes, and selective zones). It shall be noted that the variables input node does not take inputs from any parameters or components except for ***Number Slider*** parameters. That is why it very important to formulate the variables in ***Number Slider*** parameters (with upper and lower limits in each variable). The objective function node (also named as the Fitness node) is connected to the ***Number*** parameter representing the objective function. The objective function node does not take inputs from any components or parameters except from ***Number*** parameters. It shall also be noted that Galapagos is a single-objective optimizer. So, its objective function node is connected only to one ***Number*** parameter representing only one objective function.

If the Galapagos element is clicked twice, a window is opened showing three tabs: Options, Solvers, and Record. The "options" tab provides the user with the controls for the evolutionary solver such as population number, mutation and crossover rates, and stopping criteria. The "solvers" tab provides some kind of a visual representation of the optimization process and the convergence of results. It saves organizes the best valid results from the fittest to the less fit so the user can reinstate any of them. This is a very good feature, because sometimes the user might decide to choose a "less fit" solution rather than taking the "most fit" solution for reasons beyond the modeling criteria. Figure 80 shows a screenshot of the "options" tab and the "solvers" tab of Galapagos.

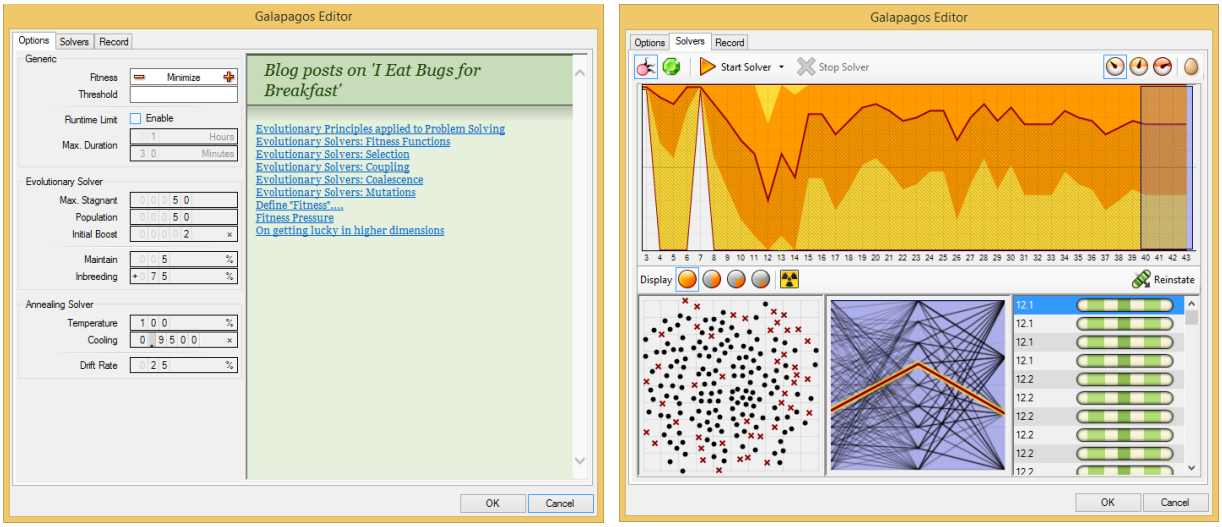

Figure 80: Galapagos control window

Chapter 4

Model Validation and Application

4 Chapter 4: Model Validation and Application

In order to validate the developed algorithms and their ability to be integrated into one model depending on the problem, a case study was made where data was gathered for a construction project and a model was developed to model and optimize the allocation of site facilities using the different algorithms developed in this research. This chapter describes the project data, the followed steps in modeling and optimization, and the results.

4.1 Project Information

The selected project for the case study is a mixed-use project located in New Cairo, Egypt; where the land boundaries are not linear and the shapes of the buildings are irregular. The project consists of 2 main buildings (A and B) constituting around 60% of the land area, and green landscape constituting the majority of the rest of the land area. Building A is for administrative purposes and building B is for a showcase and a food court. The project has 3 entrance gates and the garage is in the basement floors of the buildings. The general layout of the project is shown in Figure 81.

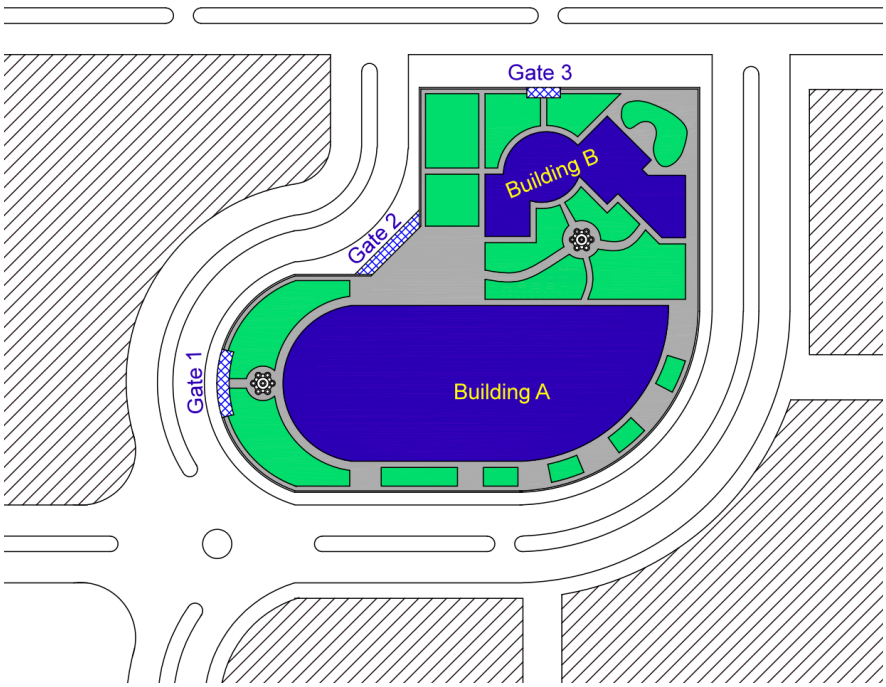

Figure 81: Case study project general layout

According to the owner, the area to the south and east of building A (marked in Xs in Figure 82) not accessible for the contractor during the construction stage of the civil works. Such area is going to be used as a temporary parking spot and as an information center for the project. Works in Building A will mark the beginning of the project, and the works in Building B will be initiated after 8 months from the start date of the construction. Accordingly, the project will assume a dynamic site layout in two phases:

Phase 1 - From month 0 to month 8: In this phase, the obstacles will be Building A, the west access road, and the area between Building A and the south and east site boundaries. The rest of the site is accessible; where the temporary facilities can be laid anywhere except in the obstacles area. The site conditions in Phase 1 are shown in part A of Figure 82.

Phase 2 - from month 9 to month 20: In this phase, Building B is in the construction process and the central and north-western access roads are in place and considered obstacles in the way of temporary facilities. The site conditions in Phase 2 are shown in part B of Figure 82. In phase 2, the site layout is very strict and the available areas for laying temporary facilities is much complex in shape.

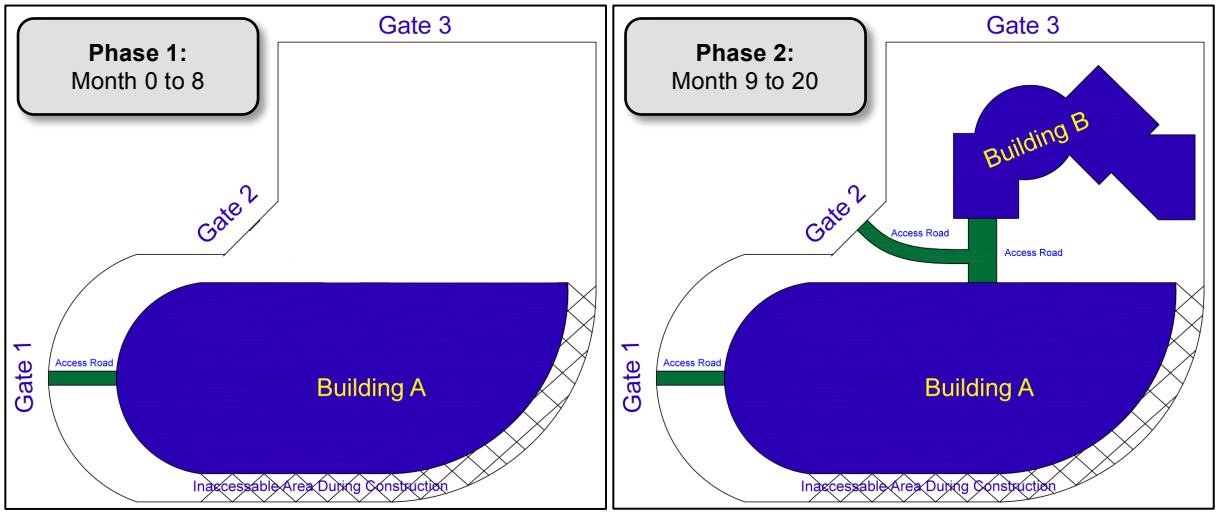

Figure 82: Site conditions during construction

4.2 Identifying and Modeling Site Facilities

After identifying the project needs, the following are the main 9 temporary facilities that were used in the model:

Table 13: Site facilities of the case study

Object No.	Temporary Facilities (Objects)	Modeling Technique
1	Offices (1) (for Owner, Project Manager, & Supervision Consultant)	Offsetted Planar Curves
2	Parking	Offsetted Planar Curves
3	Materials & excavation piles	Dynamic Freeforms
4	Wood yard	Dynamic Rectangles
5	Steel yard	Dynamic Rectangles
6	Electrical control	Static Rectangle
7	Testing labs	Static Rectangle
8	Offices (2) (for contractor & sub-contractor)	Offsetted Planar Curves
9	Labor restrooms	Static Rectangle

Assumptions: With regards to the offices (objects 1 and 8), these offices are made of caravans of standard sizes laid next to each other that include meeting rooms, restrooms, printing stations, and personnel offices. With regards to the pile (object 3), at the initial stages of the project, this pile is to be used for temporarily storing excavated soil before disposal, and storing the backfilling sand before backfilling. At later stages it is to be used for storing other materials such as finishing materials. The wood yard (object 4) includes cutting and joining yards and storage of raw and finished pieces. The steel yard (object 5) includes aligning and bending yards and storage of raw and finished pieces. The electrical control (object 6) includes the generator, the solar tank, and the control board. The testing labs (object 7) include restrooms for lab personnel. The labor restrooms (object 9) include their own sewage disposal unit. The used concrete for civil works is ready-mix, so there is no need for a batching plant or gravel and cement storage inside the site since all the concrete is coming ready from an external source.

When it comes to formulating this site layout optimization problem on Grasshopper®, it is important to model the site conditions and obstacles before modeling the site facilities. Accordingly, the following steps shall be followed:

1. Model the site boundaries
2. Model the site obstacles (in this case study, the obstacles are the two buildings that are planned to be constructed and the access roads, and the temporary site facilities)
3. Model the site facilities

4.2.1 Modeling the site boundaries and obstacles:

There are two main methods for modeling the site boundaries and obstacles. The first method is obtaining the geometric parameters of the 2D shapes forming the boundaries and obstacles an using the obtained parameters to form corresponding points and curves on Grasshopper®. The second method is by importing the lines from AutoCAD directly to Rhino® and setting the imported shapes as the boundaries and obstacles on Grasshopper® afterwards. The following paragraphs provide a brief description of the to methods. It shall be noted that the 2 buildings are not inputted in the model as is, but rather their boundaries are offsetted outside for a distance of 2 meters, and the offsetted geometry is the one that is used in the model. Such offset took into consideration the excavation method and excavation depth for the foundations.

Method 1 – Modeling by using parameters: The advantage of this method is that enables the users to benefit from the parametric capabilities of Grasshopper® by dividing the geometries into points, lines, an curves with known parameters and drawing them on Grasshopper® using these parameters, so when any of the parameters changes, the geometry is easily modified without revising the whole model. The disadvantage of this method is that it takes a lot of time initially since the user has to divide the geometry into points, lines, and curves, then input each of them on Grasshopper® using the different components and parameters.

Figure 83 demonstrates the modeling of site boundaries and obstacles on Grasshopper using method 1. The first step is to obtain the following from the AutoCAD drawing of the site:

- For lines: coordinates of the starting and ending points.
- For arcs: coordinates of the starting and ending points, and any point in the middle.
- For freeforms: coordinates of three or more points depending on the degree and the number of control points.

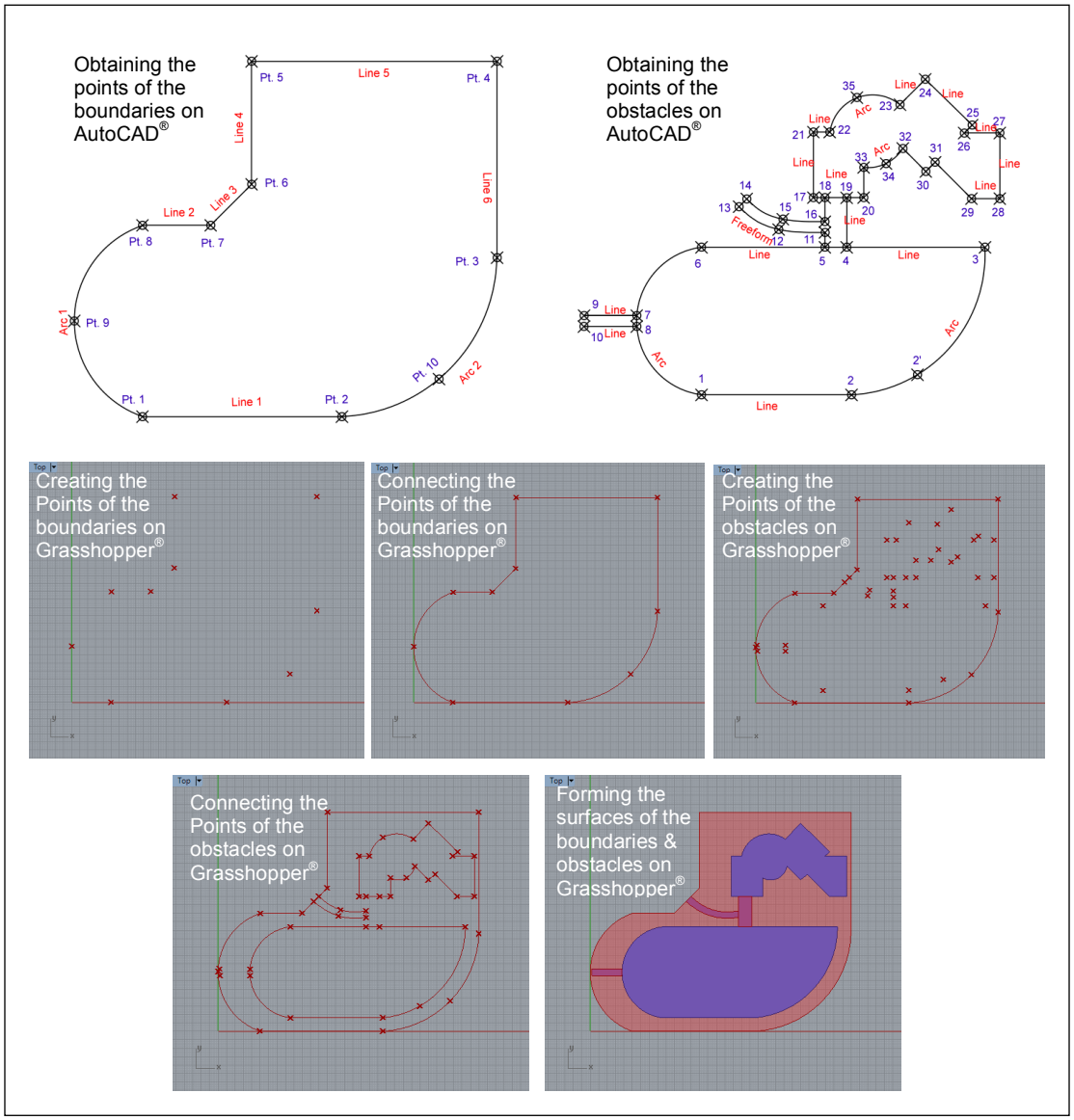

Figure 83: modeling of site boundaries and obstacles on Grasshopper using method 1

The obtained parameters of the site boundaries and obstacles of the case study from the AutoCAD drawing are shown in Table 14 and Table 15. Afterwards, in Grasshopper®, the ***Construct Point*** component is used to create the points in the model. Then a combination of ***Line, Arc, Interpolate***, and other components are used to connect the points to form lines and curves. Then, the ***Boundary Surfaces*** component is used to create the surface definitions that are enclosed by the lines and curves materializing the boundary edge curves, thus forming the site boundaries and obstacles.

Table 14: Parameters of site boundaries of the case study obtained from AutoCAD (units in m)

Points	Coordinates
Point 1	(25,0,0)
Point 2	(98,0,0)
Point 3	(155,58,0)
Point 4	(155,30,0)
Point 5	(65,130,0)
Point 6	(65,85,0)
Point 7	(50,70,0)
Point 8	(25,70,0)
Point 9	(0,35.5,0)
Point 10	(138,18,0)

Point connections	First Point	Second Point	Third Point
Line 1	Point 1	Point 2	--
Line 2	Point 8	Point 7	--
Line 3	Point 7	Point 6	--
Line 4	Point 6	Point 5	--
Line 5	Point 5	Point 4	--
Line 6	Point 4	Point 3	--
Arc 2	Point 2	Point 10	Point 3
Arc 2	Point 1	Point 9	Point 8

Table 15: Parameters of site obstacles of the case study obtained from AutoCAD (units in m)

Points	Coordinates
Point 1	(43,8,0)
Point 2	(98,8,0)
Point 3	(147,62,0)
Point 4	(96,62,0)
Point 5	(88,62,0)
Point 6	(43,62,0)
Point 7	(19,37,0)
Point 8	(19,37,0)
Point 9	(19,33,0)
Point 10	(1,33,0)
Point 11	(67,67.5,0)
Point 12	(71.5,68.5,0)

Points	Coordinates
Point 13	(57,77,0)
Point 14	(60,80,0)
Point 15	(73,72,0)
Point 16	(67,71.5,0)
Point 17	(84,80,0)
Point 18	(88,80,0)
Point 19	(96.5,80,0)
Point 20	(102.5,80,0)
Point 21	(84,104,0)
Point 22	(90,104,0)
Point 23	(116,114,0)
Point 24	(125,123.5)

Points	Coordinates
Point 25	(142.5, 106.5, 0)
Point 26	(139.5,104,0)
Point 27	(152.4,104,0)
Point 28	(142.4,80,0)
Point 29	(142,80,0)
Point 30	(125,90,0)
Point 31	(129,93,0)
Point 32	(117,98,0)
Point 33	(102.5,91,0)
Point 34	(98,115,0)
Point 35	(112,95,0)

Method 2 – Directly importing from AutoCAD to Rhino®: The advantage of this method is that it is very time-saving and it does not involve a lot of parameter and components such as the ones used in method 1 where there has to be a component for each and every point and curve. The user imports the geometries from AutoCAD to Rhino® then selects the lines and curves and sets them as Curve definitions on Grasshopper® using the ***Curve*** component – It does not require that each single line and curve to have its own "curve" definition, but rather the user may select multiple lines and curves and put them under one "curve" definition on Grasshopper®. Finally, the ***Surface*** parameter is used to create a "surface" definition of the boundaries and obstacles. Figure 84 shows a demonstration of the importing and modeling process on Rhino®.

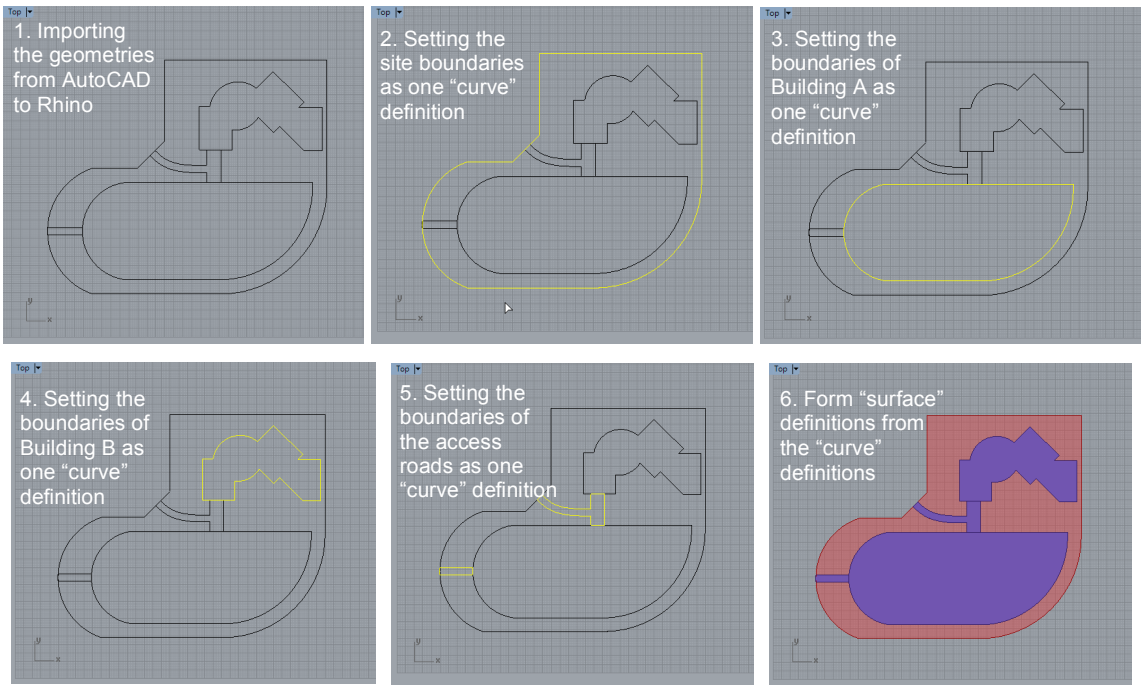

Figure 84: The importing and modeling process of site boundaries and obstacles of the case study on Rhino®

4.3 Selective Zoning

In order to minimize the number of possible solutions and increase the model efficiency to reduce the computing time, the selective zoning module was utilized as discussed in Section 3.3. The accessible area of the site is divided into 12 similar zones in Phase 1 that are turned into 11 zones in Phase 2. Each zone is 20 meters in width and 25 meters in length. The zones shown in Figure 85 represent the allowed movement of the centroids of the site facilities. Accordingly, the actual borders of the facilities may reach outside of the zones, which is acceptable, and which also is the reason for placing some zones not strictly right next to each other, but rather with keeping some distance between them. The reference point of each zone is the bottom left vertex. Coordinates of reference points of the different zones are shown in Table 16.

Table 16: Reference point coordinates for selective zoning

Zones	Reference point coordinates	
	Phase 1	Phase 2
Zone 1	(20, -3,0)	(20,-3,0)
Zone 2	(0,7,0)	(0,7,0)
Zone 3	(0,38,0)	(0,38,0)
Zone 4	(64, 75.5,0)	(64, 75.5.0)
Zone 5	(69,68,0)	(64,75.5,0)
Zone 6	(69,99,0)	(65,103,0)

Zones	Reference point coordinates	
	Phase 1	Phase 2
Zone 7	(89,99,0)	(86,109,0)
Zone 8	(109,99,0)	(107,112,0)
Zone 9	(129,99,0)	(120,104,0)
Zone 10	(129,68,0)	(124,64,0)
Zone 11	(109,68,0)	(100,64,0)
Zone 12	(89,68,0)	N/A

The site zones in both phases for the selective zoning algorithm are shown in Figure 85.

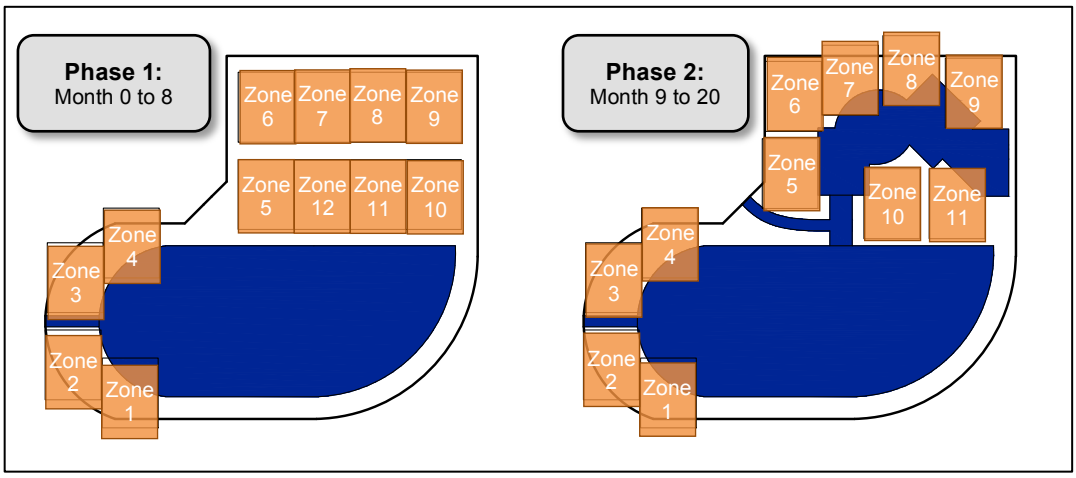

Figure 85: Division of site area into zones

4.4 Modeling the site facilities

The needed site facilities and their modeling techniques are provided in Table 13. Steps of the algorithms for the different modeling techniques of the static and dynamic site facilities are described in Section 3.2. The following conditions are inputted according to the assumptions and needs of the project:

- Width of the offsetted planar curve modeling the offices (objects 1 and 8) is 5 m.
- Width of the offsetted planar curve modeling the parking (Object 2) is 12 m.
- For the offsetted planar curve modeling the offices (a) (object 1), the minimum area is 150 m^2 and the maximum allowed area is 200 m^2.
- For the offsetted planar curve modeling the parking (object 2), the minimum area is 300 m^2 and the maximum allowed area is 400 m^2.
- For the dynamic freeforms modeling the materials and excavation piles (object 3), the minimum area is 350 m^2 and the maximum allowed area is 400 m^2.
- Dimensions of the rectangle modeling electrical control (object 6) are 6 m and 10 m.
- Dimensions of the rectangle modeling testing lab (object 7) are 10 m and 18 m.
- For the offsetted planar curve modeling the offices (b) (object 8), the minimum area is 130 m^2 and the maximum allowed area is 180 m^2.
- Dimensions of the rectangle modeling the labor restrooms (object 9) are 6 m and 10 m.

- In the collision detection constraint, the following are considered as obstacles: 1) the areas of the two buildings (A and B) under construction, 2) the three access roads, and 3) the site facilities (9 objects). Any collision or overlapping between any of the mentioned objects results in a penalty that is added to the objective function.

4.5 Proximity and Distance Measurement Inputs

In the distance measurement module, only the buildings and the facilities are considered obstacles. The access roads are not considered obstacles in distance measurement techniques because labor, equipment, and material can move across them. Regarding the Shortest Walk distances, the used grid size is 1.5 meters. The used inter-facility distance measurement techniques in the case study are shown in Table 17. Symbols in Table 18 show the proximity relationship matrix between the temporary facilities. The used proximity weights between the temporary facilities are shown in Table 20. Data in these tables were compiled in accordance to the preference of the site planner of the case study. For example, the site planner requires that the noise-producing facilities to be far from the offices. Accordingly, a negative value for the proximity weight was put between the wood and steel yards and the offices. Also there is a negative proximity weight between the electrical control facility and the rest of the site facilities since it is a noise-producing facility and for safety reasons, it needs to be put far from the rest of the facilities. One way of validating the model is by analyzing the logic of the resulting near-optimum solution with reference to the logic behind the proximity weights. The relocation costs for the temporary facilities are shown in Table 19. The relocation costs are qualitative numbers that proportional to the time, cost, and manpower required for facility relocation. For example, the offices (objects 1 and 8) have the highest relocation costs because, in addition to the required equipment for their relocation, these offices have organized paper work that would make the relocation very difficult for the fact that many important documents get lost from the offices while relocating. The used relocation costs in the case study are close to those used by Osman et al. (2003) due to the similarities in the qualitative nature of proximity weights and in the used objective function.

Table 17: Distance measurement technique matrix for case study

Distance Measurement Technique										
Objects	Description	1	2	3	4	5	6	7	8	9
1	Offices (1)									
2	Parking	CA								
3	Material piles	SW	CA							
4	Wood yard	CA	CA	CA						
5	Steel yard	CA	CA	CA	CA					
6	Electrical control	CA	CA	CA	CA	CA				
7	Testing labs	SW	CA	SW	CA	CA	CA			
8	Offices (2)	CA	CA	SW	CA	CA	CA	SW		
9	Labor restrooms	CA	CA	CA	SW	SW	CA	CA	CA	

Table 18: Proximity relationship matrix for the case study

Proximity Relationship										
Objects	Description	1	2	3	4	5	6	7	8	9
1	Offices (1)									
2	Parking	C-P								
3	Material piles	C-P	C-C							
4	Wood yard	C-P	C-C	C-C						
5	Steel yard	C-P	C-C	C-C	C-C					
6	Electrical control	C-P	C-C	C-C	C-C	C-C				
7	Testing labs	C-P	C-C	C-C	C-C	C-C	C-C			
8	Offices (2)	P-P	C-P	C-P	C-P	C-P	C-P	C-P		
9	Labor restrooms	C-P	C-C	C-C	C-C	C-C	C-C	C-C	C-C	

Table 20: Proximity weights matrix for case study

Proximity Weights (X)										
Objects	Description	1	2	3	4	5	6	7	8	9
1	Offices (1)									
2	Parking	81								
3	Material piles	-9	-9							
4	Wood yard	-9	0	3						
5	Steel yard	-9	0	3	81					
6	Electrical control	-9	0	-9	0	0				
7	Testing labs	3	3	37	-9	-9	-9			
8	Offices (2)	37	37	-9	-9	-9	-9	3		
9	Labor restrooms	0	0	0	37	37	3	-9	0	

Table 19: Relocation cost of temporary facilities

	Temporary Facilities (Objects)	Relocation cost
1	Offices (1)	5000
2	Parking	300
3	Materials & excavation piles	2500
4	Wood yard	1500
5	Steel yard	1000
6	Electrical control	300
7	Testing labs	4000
8	Offices (2)	5000
9	Labor restrooms	300

4.6 Buffer Zones

Buffer zones were created around the wood yard and steel yard with a space of 2 meters all around because of the heavy material circulation that requires some allowance in the space for maneuvering. Moreover, a buffer zone of 1 meter all around the labor restrooms is created to enhance the safety of the labor since due to the heavy circulation around it. A buffer zone was also created around the material pile to allow for labor and equipment to maneuver the materials while reducing the probabilities of collision incidents with other facilities.

4.7 Final Model

Figure 86 shows a broad view of the final formulation of the Grasshopper® model prior to running and searching for the optimum solution. It includes a mix of components and parameters forming algorithms that are described in Section 3 relevant to the case study. The model is fully parametric; meaning that it was not imported directly from AutoCAD®, but rather the geometries and the full optimization model were drawn and programmed using the graphical algorithm editor Grasshopper®.

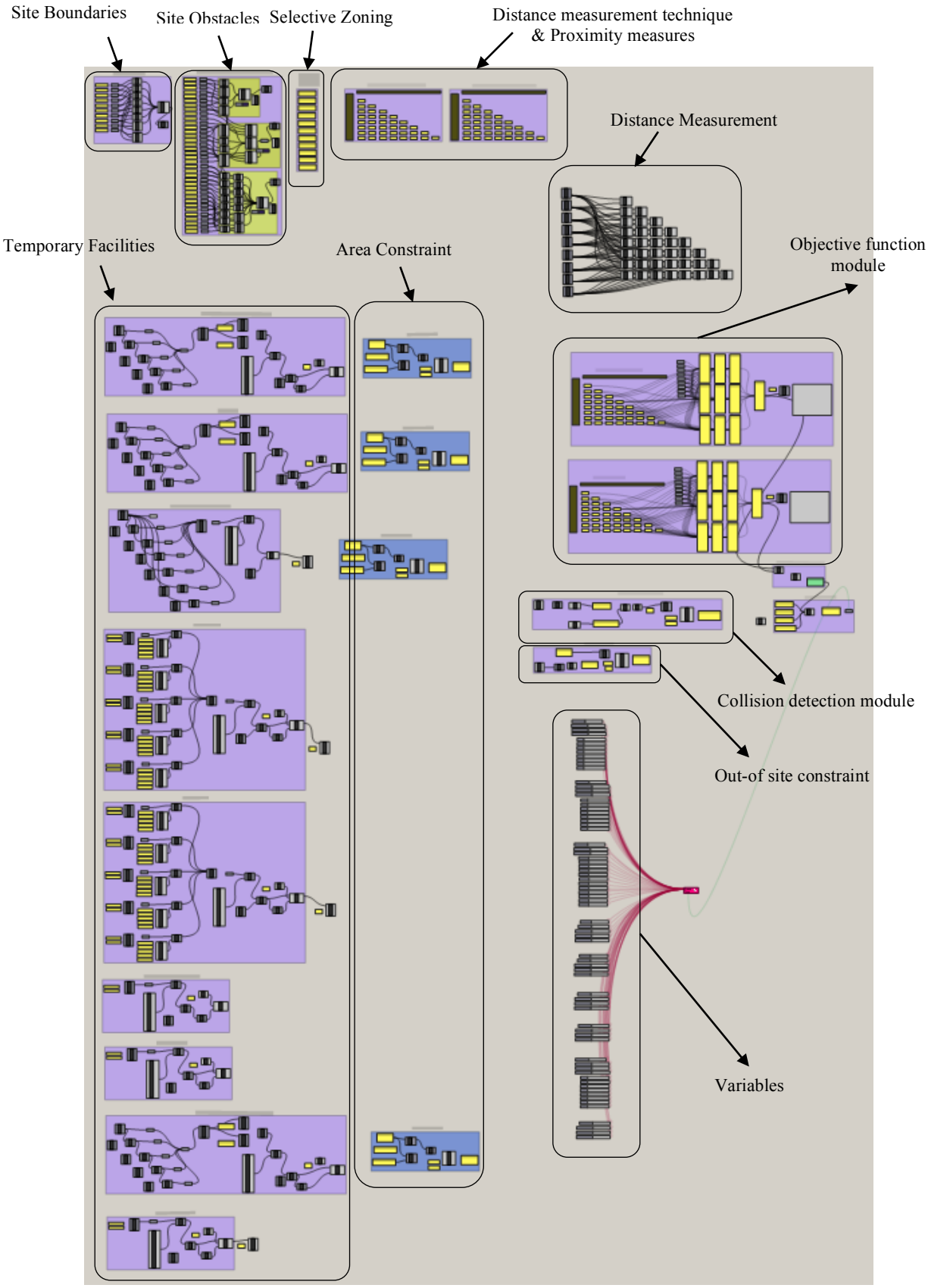

Figure 86: Grasshopper® model of the case study

4.8 Optimization Results and Model Validation

The site layout in the case study is the dynamic type, so a run was made for each phase. The used approach for the dynamic layout optimization in this case study is the Critical Phase approach; which is based on the prevalence of a phase that is identified a critical phase for its higher transportation costs or for any other logical reasons that might entail to the site planner. Dynamic optimization proceeds in backward chronological order for all phases preceding the critical phase and in forward chronological order for all phases succeeding the critical phase (Othman et al., 2003). The critical phase in the case study is Phase 2 because in this phase, the site is very strict, thus not giving much room for maneuvering or proposing many possible layouts. On the other hand, in Phase 1 the site is close to an open site with abundant area for maneuvering and flexible possibilities for temporary facilities. Accordingly, it is more logical to make the layout of Phase 1 dependent on the layout of Phase 2. It shall be stated that other dynamic optimization approaches could be easily incorporated in the model. RUN 1 was made for Phase 2; where the objective function was calculated using Equation 38. In RUN 1 (Phase 2), the obstacles were Building A, the western access road, and the area between Building A and the eastern and southern site boundaries. RUN 1 was fully formulated using the visual programming capabilities of Grasshopper® and was run using the evolutionary solver add-in, Galapagos. The mutation rate was set to 45% with population size of 50 offsprings. The model was programmed to stop when it reaches 500 stagnant successive generations without progress in the score. The model produced a near-optimum solution with a score of **3,511.8.**

For Phase 1, RUN 2 was made taking into consideration the relocation cost as shown in Equation 39; thus solving the whole problem as a dynamic layout problem. In RUN 2 (Phase 1), the obstacles were Building A, Building B, the western access road, the central and northwestern access roads, and the area between Building A and the eastern and southern site boundaries. RUN 2 was fully formulated using the visual programming capabilities of Grasshopper® and were run using the evolutionary solver add-in, Galapagos. The mutation rate was set to 45% with population size of 50 offsprings. The model was programmed to stop when it reaches 500 stagnant successive generations without progress in the score. The model produced a near-optimum solution with a score of **2,539.9.**

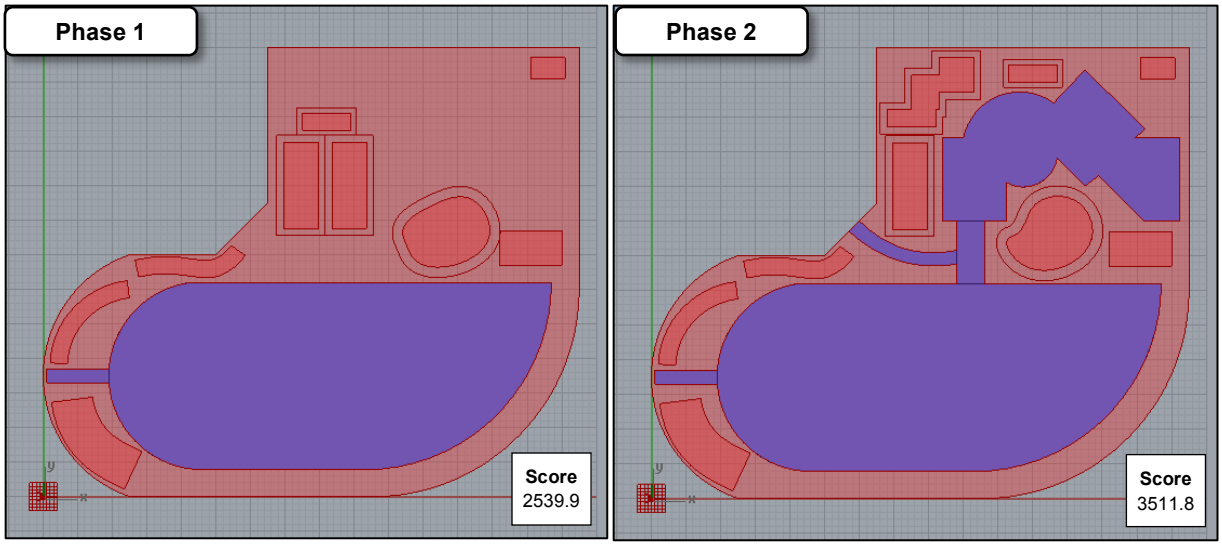

Figure 87: Near-optimum solution obtained by the developed model

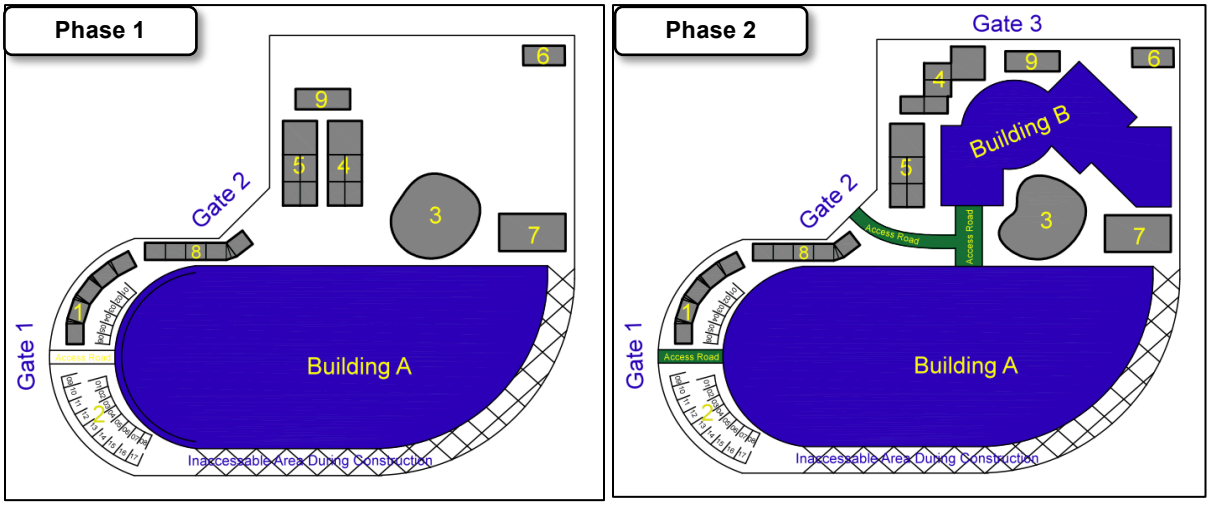

Figure 88: Corresponding AutoCAD® drawing of the selected near-optimum site layout plan

The near-optimum solutions for Phase 1 and Phase 2 are shown in Figure 87. According to the figure, the produced site layout plan is very logical; in phase 2, which is the critical phase in the dynamic optimization, the caravans are laid next to each other and the parking is laid next to them; which was highlighted in the proximity relationships in the proximity relationship matrix set by the project manager. Moreover, the excavation and material pile is put close to both of the two buildings; which facilitates the material movement between the buildings and the excavation and material pile. The electrical control facility is laid very far from the other facilities to minimize the noise in accordance to the proximity relationship matrix. Furthermore, the wood yard and steel yards are laid next to each other due to the strong proximity relationship that they

have. They both were laid in different arrangements according to the available area on site; which made use of the Dynamic Rectangles algorithm. The buffer zones around the wood yard and steel yard were considered by the model and there were no overlapping in any of the buffer zones accordingly. Although the forming curves of the caravans (objects 1 and 8) were modeled using third degree polynomial interpolation, the real-life interpretation is just the normal rectangular caravans laid next to each other trying as much as possible to mimic the modeled shapes as shown in Figure 88, which shows the AutoCAD drawing of the selected site layout plan during construction. Table 21 shows the corresponding values of the different variables for the temporary facilities. In RUN 2, for phase 1, the model moved only 2 facilities; the wood yard and the labor restrooms, and left the rest of the temporary facilities in their locations set from RUN 1. Although the site in RUN 2 is an open site with very large area for maneuvering temporary facilities and finding better near-optimum solutions, the model chose to keep most of the temporary facilities in their original location set from RUN 1 because the added relocation costs would have added to the objective function more than the better proximity measures would have deducted. Some would consider this as a limitation to the Critical Phase approach in the dynamic layout optimization, however, the overall resulting dynamic layout plans from the Critical Phase approach are logic and do not have significant shortcomings. Other methods in dynamic optimization can be used in the developed model, but investigating the performance of the model in the various dynamic optimization approaches is not the main scope of this research.

As discussed in this section, the resulting site layout plans from RUN 1 and RUN 2 are logical and conformant with the inputted proximity weights matrix. Such logical results provide validation of the model and its capability of outputting good solutions.

Another validation is made by solving the same case study problem with past optimization models and comparing their results to the results of the developed model. Such comparison and validation is provided in Section 4.9.

Table 21: Values of variables of the temporary facilities corresponding to the best-reached near-optimum solution

Variables		Numerical Values								
		Object 1	Object 2	Object 3	Object 4	Object 5	Object 6	Object 7	Object 8	Object 9
Movement zone	Phase 1	3	2	11	5	5	9	10	4	6
	Phase 2	3	2	11	6	5	5	10	4	7
Shape Movement (X)	Phase 1	-3	1	3	7	-7	17	12	7	13
	Phase 2	-3	1	9	4	-1	11	18	7	22
Shape Movement (Y)	Phase 1	10	7	7	17	17	25	4	16	7
	Phase 2	10	7	11	9	9	14	8	16	11
Shape Rotation (x45)	Phase 1	1	3	-	2	2	2	4	0	0
	Phase 2	1	3	-	0	2	2	4	0	0
Arrangement	Phase 1	-	-	-	0	0	-	-	-	-
	Phase 2	-	-	-	4	0	-	-	-	-
Pt 1 (X) Vector	Phase 1	-	-	0	-	-	-	-	-	-
	Phase 2	-	-	0	-	-	-	-	-	-
Pt 1 (Y) Vector	Phase 1	-	-	9	-	-	-	-	-	-
	Phase 2	-	-	9	-	-	-	-	-	-
Pt 2 (X) Vector	Phase 1	7	3	12	-	-	-	-	9	-
	Phase 2	7	3	12	-	-	-	-	9	-
Pt 2 (Y) Vector	Phase 1	4	1	11	-	-	-	-	1	-
	Phase 2	4	1	11	-	-	-	-	1	-
Pt 3 (X) Vector	Phase 1	10	9	17	-	-	-	-	9	-
	Phase 2	10	9	17	-	-	-	-	9	-
Pt 3 (Y) Vector	Phase 1	2	2	0	-	-	-	-	-1	-
	Phase 2	2	2	0	-	-	-	-	-1	-
Pt 4 (X) Vector	Phase 1	9	10	-5	-	-	-	-	4	-
	Phase 2	9	10	-5	-	-	-	-	4	-
Pt 4 (Y) Vector	Phase 1	-2	-2	-6	-	-	-	-	0	-
	Phase 2	-2	-2	-6	-	-	-	-	0	-
Pt 5 (X) Vector	Phase 1	5	5	-7	-	-	-	-	7	-
	Phase 2	5	5	-7	-	-	-	-	7	-
Pt 5 (Y) Vector	Phase 1	-3	-3	0	-	-	-	-	5	-
	Phase 2	-3	-3	0	-	-	-	-	5	-
Pt 6 (X) Vector	Phase 1	-	-	-5	-	-	-	-	-	-
	Phase 2	-	-	-5	-	-	-	-	-	-
Pt 6 (Y) Vector	Phase 1	-	-	1	-	-	-	-	-	-
	Phase 2	-	-	1	-	-	-	-	-	-

4.9 Validation by comparison with other past optimization models

The same case study was attempted once using with the circular modeling method - used by Andayesh & Sadeghpour (2013), and once using the orthogonal square grids modeling method – used by Hegazy & Elbeltagi (1999) and Osman et al. (2003).

In the first attempt, by using the circular modeling method, the site boundaries, buildings, and obstacles are modeled using a series of connected straight lines; which is a good abstraction for them that does not result in significant inefficiencies. The temporary facilities, on the other hand, are represented by their minimum bounding circles as shown in Figure 89, which poses large area inefficiencies. Although this method is very fast in running due to the small number of variables, the inefficiency in modeling the facilities is very significant. In Figure 89 the left and middle parts of the figure show the representation of the site boundaries, buildings and obstacles in Phase 1 and Phase 2; and the right side of the figure shows the representation of the temporary facilities that need to be organized in the available area of the site. Since the site is strict and complex in shape, modeling the temporary facilities using enclosing circles is highly inefficient.

Site layout optimization was attempted in both Phase 1 and Phase 2 but, as expected, the optimization model using the circular method failed to generate valid solutions in both phases. Accordingly, by comparing the developed model to the circular modeling method, it is clear that just providing a valid solution for the problem provides a good validation for the developed model's capabilities in performing in strict sites.

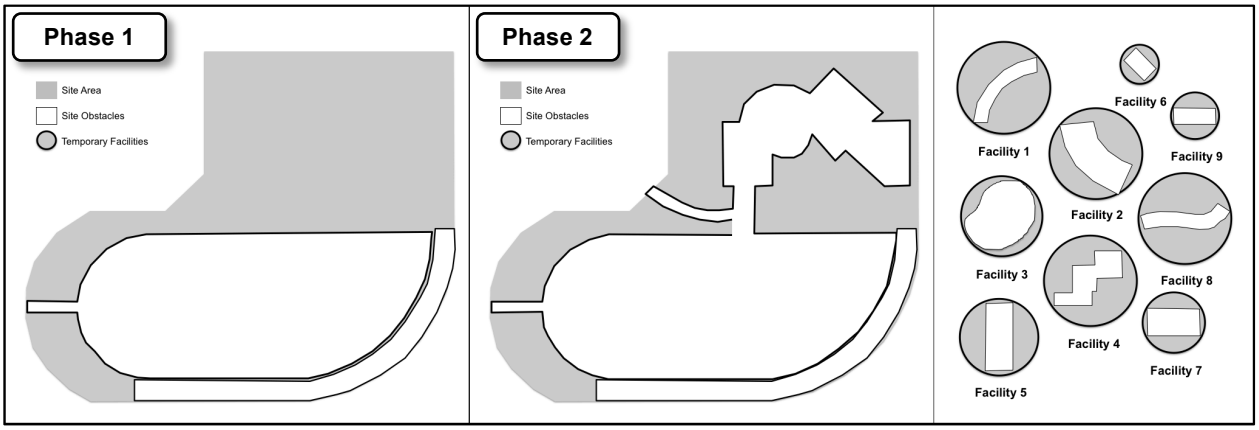

Figure 89: Modeling the case study problem using the circular modeling method. *Facilities: 1) Offices (a), 2) parking, 3) material pile, 4) wood yard, 5) steel yard, 6) electrical control, 7) testing labs, 8) offices (b), 9) labor restrooms*

In the second attempt, by using the orthogonal square grids modeling method, the site boundaries, buildings, obstacles, and temporary facilities were modeled using two-dimensional square grids that add up the corresponding areas. The model ran using a combination of the algorithms provided by Hegazy & Elbeltagi (1999) and Osman et al. (2003). Phase 2 was taken as the critical phase and its run was made first (RUN 1). RUN 1 obtained a near-optimum score of **8,584.8**; which is double the score reached by the developed model. The model could not reach any better scores due to the inefficiencies in the geometrical modeling using the orthogonal grids and using static geometrical representations in general; especially that the construction site in Phase 2 is very strict and does not give the luxury to models with static geometrical representations of finding many valid solutions. In RUN 2 (phase 1), the orthogonal grid model produced a near-optimum solution with a score of **2,117.7** by moving 4 facilities; parking, wood yard, steel yard, and labor restrooms. Since the score from RUN 1 was very high, the model chose to move many temporary facilities from their original location set from RUN 1 because the corresponding deductions in the objective function from the better proximity measures are more than the added relocation costs. The near-optimum solutions of the orthogonal square grids modeling method for Phase 1 and Phase 2 are shown in Figure 90.

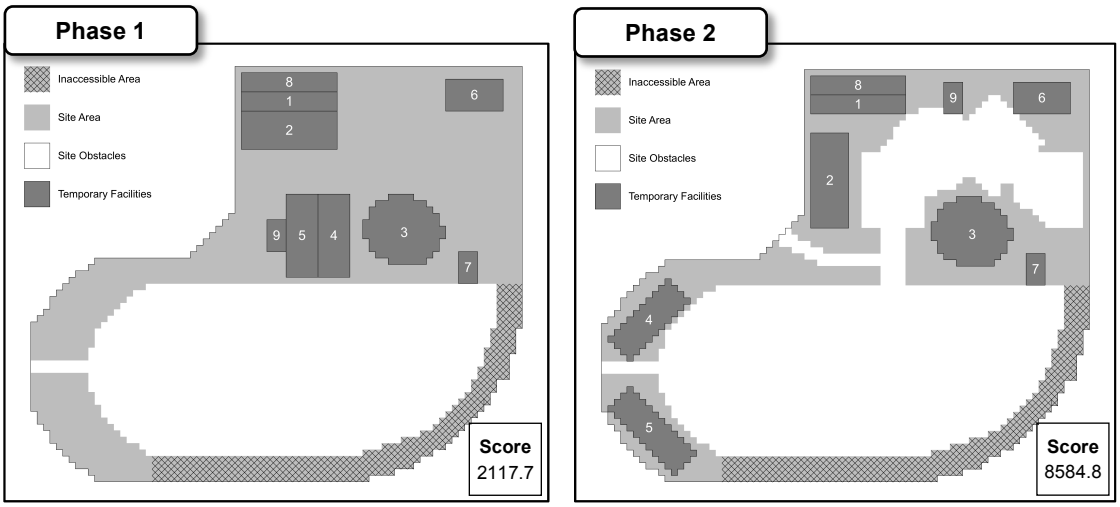

Figure 90: Modeling the case study problem using the orthogonal square grids method. *Facilities: 1) Offices (a), 2) parking, 3) material pile, 4) wood yard, 5) steel yard, 6) electrical control, 7) testing labs, 8) offices (b), 9) labor restrooms*

A quantitative comparison between the developed model and the other used models is provided in Table 22. The overall layout score is the addition of the score of Phase 1 and Phase 2.

Table 22: Comparison of scores corresponding from different dynamic site layout optimization models

	Developed Model	Orthogonal Grids	Circular Representations
Phase 1 Score	2,539.9	2,117.7	N/A
Phase 2 Score	3,511.8	8,584.8	N/A
Overall Score	**6,051.7**	**10,702.5**	**N/A**
Weighted Average Score	**3,123.0**	**5,998.0**	**N/A**

In Phase 2, where the site in the case was strict, the developed model obtained a near-optimum solution that has a score less than half of the score obtained by the orthogonal grid modeling method, which provides a measure of the efficiency and flexibility of the developed model. Comparing between the models in each phase separately does not grasp the dynamic efficiency of the modeling; a better comparison is achieved by comparing between the overall scores because they take all phases into consideration. An even better comparison is achieved by comparing between the weighted average scores because they take into consideration the phases and their durations. The developed model utilizing dynamic geometrical representations achieved a lower overall score and weighted average score of 43% and 48% respectively from those achieved by the orthogonal grids model; which is a significant enhancement. It could be

deduced that no matter how accurately a model can geometrically represent site layout elements, a static geometrical representation will always pose restrictions and inefficiencies leading to unnecessary high proximity scores, especially in strict construction sites. A dynamic geometrical representation of temporary site facilities eliminates many geometrical restrictions and minimizes the inefficiencies by continually shaping the temporary facilities to fit them into the near-optimum locations that would not have been found fitting in models with static geometrical representation; especially in strict sites.

Chapter 5

Conclusion and Recommendations

5 Chapter 5: Conclusion and Recommendations

5.1 Summary and Conclusion

An accurate representation of the construction site is important for site layout modeling, as it enables the development of more realistic and efficient layouts. Previous research efforts used grids, circles, and rectangles to model site facilities; which simplifies the search procedure but jeopardizes the modeling efficiency resulting in loss of modeled site area leading to difficulties in finding valid solutions in strict sites. In reality, the construction site and facilities acquire complex and flexible shapes. After investigating the current progress in the site layout modeling research, concluding the current gaps, and clearly defining the problem statement, the following is a summary of what was performed in this research:

- **New algorithms for modeling regularly and irregularly shaped static and dynamic geometrical representations of site facilities:** This research presented a new and more realistic approach of modeling site facilities. It presented the algorithms of modeling different "static" geometries such as rectangles, triangles, circles, irregular polygons, ellipses, and freeforms, and "dynamic" geometries such as offsetted planar curves, dynamic freeforms, and dynamic rectangles. Dynamic geometries are where a facility is assumed to acquire a certain shape and in every iteration, it changes its shape given certain constraints until it reaches a shape that fits in the unoccupied land on site.

- **Proximity relationships and distance measurement techniques:** The developed model provides different proximity measures and distance measurement techniques rather than the normal centroidal Cartesian distances used in most models. The new proximity measures take into consideration actual movement between the facilities including any passageways or access roads on the site. Four proximity relationships are utilized; center-to-center, point-to-point, center-to-point, and site-to-side. Two distance measurement techniques are also used; Cartesian distance, which is the direct straight-line distance between any two fatalities, and shortest walk distance, which is the distance of the path between any two facilities that does not interfere with obstacles. Soft constraints for buffer zones around the site facilities are developed as well.

- **Selective zoning concept:** The research also introduces the concept of selective zoning; where instead of setting the variables to move in one big movement zone with many invalid solutions, many small similar zones are pre-specified and the variables are set to move only in these selective zones. The concept of selective zoning causes a major decrease of the number of possible solutions, thus significantly reducing the running time. Not only this, but also the percentage of valid solutions relative to the total number of possible solutions is high.

- **Formulation of a full site layout optimization model:** All of the developed algorithms were integrated into one site layout optimization model and formulated on a commercial parametric modeling tool, Grasshopper®, as the platform got modeling for its capabilities in graphical algorithm programming. The optimization was performed through genetic algorithms. After each of the algorithms has been verified and validated, a case study of a dynamic site layout planning problem was made to validate the comprehensive model combining all of the modules together. Different proximity measures and distance measurement techniques were considered, along with different static and dynamic geometrical shapes for the temporary facilities. The dynamic technique in dynamic optimization for the case study was the critical phase approach. The model produced valid near-optimum solutions that are in conformance with the given inputted conditions, then the same case study problem was attempted using two previous models: once with the circular modeling method and once with the orthogonal square grids modeling method. The square modeling method failed to produce any valid solutions for the case study due to the large geometrical inefficiencies in the modeling. The orthogonal square grids modeling method produced a near-optimum solution that has a score double of the score produced by the developed model. Accordingly, the developed model has proven its modeling capabilities and advantages in strict site layouts. It shall also be stated that the developed model can suit dynamic optimization approaches other than the critical phase approach by simple modifications in the variables and constraints.

It was concluded that no matter how accurately a model can geometrically represent site layout elements, a static geometrical representation will always pose restrictions and inefficiencies leading to unnecessary high proximity scores, especially in strict construction sites. A dynamic

geometrical representation of temporary site facilities eliminates many geometrical restrictions and minimizes the inefficiencies by continually shaping the temporary facilities to fit them into the near-optimum locations that would not have been found fitting in models with static geometrical representation.

5.2 Research Outcomes and Contributions

The following points summarize the contribution of this research to the ongoing research of site layout modeling and optimization:

- Exploring new automated modeling capabilities through the use of the *visual programming* capabilities of the parametric modeling software (Grasshopper®).
- Enhancing the ability to model complex geometrical shapes to increase efficiency and decrease area waste due to poor geometrical representation of the site facilities.
- Introducing 3 different algorithms for dynamic shapes for modeling site facilities, which is a more realistic and flexible representation that fits in strict sites:
    - Offsetted planar curves
    - Dynamic freeforms
    - Dynamic rectangles
- Introducing the modeling concept of selective zoning and providing an algorithm for it. The concept significantly enhances the model efficiency by minimizing the number of solutions through selection of pre-determined movement zones inside the site.
- Utilizing different proximity measures that are more flexible than the ones used in previous research efforts.
- Providing an algorithm for the shortest walk distance measurement technique; which finds the shortest uninterrupted path between facilities.
- Simplifying the application of buffer zones around facilities acting as soft constraints for better space utilization.

5.3 Limitations and Recommendations

A limitation of the developed model is the relatively slow processing time due to the immense number of variables resulting in a very large pool of probable solutions. The processing time is even slower in the case of utilization of the buffer zones. Moreover, in the selective zoning

algorithm, the predetermined zones have to be equal in size, which might pose some modeling inefficiencies. Furthermore, to be able to use the algorithms introduced in this research, one has to have prior knowledge of Grasshopper®, a software that is rarely used by construction engineers. The different dynamic optimization approaches were not heavily investigated in this research because it was more oriented towards developing new and better algorithmic geometrical representations for the site layout facilities. It shall be noted that the site safety was not heavily studied in the developed model in this research, as its main focus was concentrated on finding better ways of modeling the site facilities. Another limitation of this study is that it did not investigate the effect of other parameters, such as the probability of crossover and the probability of mutation on the convergence.

With regards to the recommendations, the model is recommended to undergo further developments to have the ability to incorporate more dynamic site layout optimization algorithms. Also, it is recommended to find new ways to model site facilities that reduce the variables and lessen the computing time; especially in the buffer-zones algorithm. A suggestion would be to combine heuristic methods with the mathematical modeling methods to provide much faster results with better convergence rates. Another recommendation also is to develop the model into a more user-friendly interface by summarizing each of the modules into one component of Grasshopper® without getting the user involved in the details of the visual programming and the internal calculations. It is also recommended to add further developments to the evolutionary optimizer Galapagos by adding a module for hard constraints. Researchers are also encouraged to explore the model's performance and capabilities in various dynamic and hybrid optimization approaches. Moreover, it is also recommended in future research to investigate the site safety and find ways to incorporate different safety factors in the objective function calculation.

6 References

[1] Andayesh, M., Sadeghpour , F. (2013). “Dynamic site layout planning through minimization of total potential energy”, Automation in Construction, 31, 92-102

[2] Andayesh, M., Sadeghpour , F. (2014). “The time dimension in site layout planning”, Automation in Construction, 44, 129-139

[3] Aweida, C. (2011). “Evolutionary Form Finding with Grasshopper + Galapagos”, accessed January 3rd , 2015. *<https://yazdanistudioresearch.wordpress.com/2011/08/04/evolutionary-form-finding-with-grasshopper-gal apagoes/>*

[4] Cheng, M.Y. and O’Connor, J.T. (1996). “ArcSite: Enhanced GIS for construction site layout”, Journal of Construction Engineering & Management, ASCE, 122(4), 329-336

[5] Cheung, S.-O., Thomas Kin-Lun Tong, Tam, C.-M. (2002). “Site pre-cast yard layout arrangement through genetic algorithms” Automation in Construction , 11(1), 35-46.

[6] Elbeltagi, E., “Construction Site Layout Planning”, Lecture Notes (PDF File), Department of Structural Engineering, Faculty of Engineering, Mansoura University, Egypt. Accessed November 2014.

[7] Elbeltagi, E. and Hegazy, T. (2001) “A hybrid AI-Based system for site layout planning in construction”, Computer-Aided Civil and Infrastructure Engineering, Blackwell Publishers ,16(2), 79-93

[8] El-Rayes, K., Said, H. (2009). “Dynamic Site Layout Planning Using Approximate Dynamic Programming”, Journal of Computing in Civil Engineering, 23 (2), 119-127

[9] Fonseca, C., Fleming, P. (1995). Multiobjective optimization and multiple constraint handling with evolutionary algorithms II: Application example, University of Sheffield, Department of Automatic Control and Systems Engineering.

[10] Hamamoto, S., Yih, Y., Salvendy, G. (1999). “Development and validation of genetic algorithm-based facility layout—A case study in the pharmaceutical industry”, International Journal of Production Research 37(4) 749–768

[11] Harniani, A., Popescu. G. (1988). “CONSI1E: A knowledge-based expert system for site layout Computing in civil engineering”, Microcomputers to supercomputers. K. M. Will. ed., ASCE, New York, N.Y., 248-256.

[12] Hegazy, T. and Elbeltagi, E. (1999). “EvoSite: Evolution based model for site layout

planning", Journal of Computing in Civil Engineering, ASCE, 13(3), 198-206

[13] Heng Li and Love, P.E. (1998). "Site level facilities using genetic algorithms", Journal of Computing in Civil Engineering, ASCE, 12(4), 227-231

[14] Huang, C., Wong, C.K. (2015). "Optimisation of site layout planning for multiple construction stages with safety considerations and requirements", Automation in Construction, 53, 58-68.

[15] Khalfallah, A., El-Rayes, K. (2002). "Trade-off Between Safety and Cost in Planning Construction Site Layouts", Journal of Constrcution Engineering and Management, 20, 313-320

[16] Khalfallah, A., El-Rayes, K. (2011). "Automated multi-objective optimization system for airport site layouts", Automation in Construction, 20, 313-320

[17] Lien, L., Cheng, M. (2012). "A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization", Automation in Construction, 39, 9642-9650.

[18] Michiel, Hazewinkel, ed. (2001), "Bézier curve", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

[19] Miller, N. (2009). "NBBJ: Parametric Strategies in the Design of Hangzhou Stadium (Part 1)", The providing Ground Blog. Accessed December, 2014. *<http://www.theprovinggground.org/2009/12/parametric-strategies-in-design-of.html>*

[20] Mitchell, M. (1996). An introduction to genetic algorithms. Cambridge, Mass.: MIT Press.

[21] Nassar, N., "Construction Site Design and Planning", Lecture Notes, 2014. CENG 423: Methods and Equipment for Construction. Department of Construction and Architectural Engineering, School of Sciences and Engineering, the American University in Cairo, Egypt. Accessed November 2014.

[22] Nedzémlyi, L., Vattai, Z., "Site Layout Design", Lecture Notes (PDF File), Department of Construction Technology and Management, Budapest University of Technology and Economics. Class: Managing Civil Engineering Projects, accessed November 2014.

[23] Nguyen, X., Lam, K., Lam, M. (2011). "A decision-making system for construction site layout planning", Automation in Construction, 20, 459-473

[24] Ning, H., Duy. (2014). "An Automated Approach to Dynamic Site Layout Planning", Master's Thesis. Department of Civil and Construction Engineering, Western Michigan University.

[25] Ning, X., Lam, K. C., & Lam, M. C. (2011). “A decision-making system for construction site layout planning”, Automation in Construction, 32, 459-473.

[26] Ning, X., & Lam, K. C. (2013). “Cost–safety trade-off in unequal-area construction site layout planning”, Automation in Construction, 32, 96-103.

[27] Osman, H., Georgy, M., Ibrahim, M. (2003). “A hybrid CAD-based construction site layout planning system using genetic algorithms”, Automation in Construction, 12, 749-764

[28] Pottmann, H., Asperl, A., Hofer, M., Kilian, A. (2007). *Architectural Geometry*: Exton, Pennsylvania USA. Bentley Institute Press. ISBN 978-1-934493-04-5

[29] Rutten, D. (2010). “Evolutionary Principles applied to Problem Solving”. Grasshopper3D online blog. Accessed December, 2015. *<http://www.grasshopper3d.com/profiles/blogs/evolutionary-principles>*

[30] Sadeghpour, F., Moselhi, O., Alkass, S. T. (2006). “Computer-aided site layout planning”, Journal of construction engineering and management, 132(2), 143-151.

[31] Said, H. (2010). “Optimizing Site Layout and Material Logistics Planning During the Construction of Critical Infrastructure Projects”. PhD. University of Illinois at Urbana-Champaign.

[32] Said, H., El-Rayes, K. (2013). “Performance of global optimization models for dynamic site layout planning of construction projects”, Automation in Construction, 36, 71-78.

[33] Tam, C. M., Tong, T. K., Leung, A. W., & Chiu, G. W. (2002). “Site layout planning using nonstructural fuzzy decision support system”, Journal of construction engineering and management, 128(3), 220-231.

[34] Tanaka, H., Yoshimoto, K. (1993). “Genetic algorithm applied to the facility layout problem”, Dept. of Industrial Engineering and Management, Waseda Univ., Tokyo.

[35] Tate, D., Smith, A. (1993). “Genetic algorithm optimization applied to variations of the unequal area facilities layout problem”, Proceedings of the 2nd Industrial Engineering Research Conference, 335–339

[36] Tate, D., Smith, A. (1995). “A genetic approach to the quadratic assignment problem”, Computers and Operations Research Journal, 22(1) 73–83.

[37] Tommelien, I.D., Levit, R.E., and Hayes-Roth, B. (1992). “SitePlan model for site layout”, Journal of Construction Engineering & Management, ASCE, 118(4), 749-766

[38] Xu, J., Li, Z. (2012). “Multi-Objective Dynamic Construction Site Layout Planning in Fuzzy Random Environment”, Automation in Construction, 27, 155-169

[39] Yahya, M., Saka, M. (2014). “Construction site layout planning using multi-objective artificial bee colony algorithm with Levy flights”, Automation in Construction, 38, 14-29

[40] Yeh, I. (1995). “Construction-site layout using annealed neural network”, Journal of Computing in Civil Engineering, ASCE, 9(3), 201-208

[41] Zadeh, L. A. (1965). “Fuzzy sets”, Information and Control, 8(3) 338-353

[42] Zhou, F., AbouRizk, S.M. and Al-Battaineh, H. (2009)." Optimisation of construction site layout using a hybrid simulation-based system", Simulation Modelling Practice and Theory, ASCE, 16(2), 348-363

[43] Zouein, P., Harmanani, H. and Hajar, A. (2002)."A genetic algorithm for solving the site layout problem with unequal size and constrained facilities", Journal of Computing in Civil Engineering, ASCE, 16(2), 143-151

[44] Zouein, P., Tommelein, I. D. (1999). “Dynamic layout planning using a hybrid incremental solution method”, Journal of construction engineering and management, 125(6), 400-408.