Automation of Steel Shear Connection Design using Generative Design

by

Eric Duong

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

STRUCTURAL ENGINEERING

Department of Civil and Environmental Engineering
University of Alberta

ABSTRACT

Steel fabricators and connection designers in the building construction industry often rely on traditional connection design methods, which involve using connection design software or spreadsheets to simulate and design connections with the goal of minimizing costs associated with fabrication and erection. However, accurately assessing the efficiency and cost implications of these designs can be a significant challenge. This is due to the complexity of the process, which involves considering a wide range of parameters such as material tonnage, machinery availability, connection type, labour cost, fabrication and erection constraints, and available optimization tools.

The connection design process can be time-consuming, especially when changes occur throughout the lifecycle of a project. These changes are common in the construction industry, and they can result in designers having to redo or recalculate a significant number of their designs, leading to a loss of efficiency and productivity.

To address this challenge, this research aims to develop an optimization tool that streamlines the connection design workflow and allows designers to leverage emerging technologies such as parametric and generative design to generate multiple options in the early stages of design. This will enable designers to receive instant feedback on how changes affect cost and constructability, allowing for more efficient and effective connection design. Additionally, this tool will promote better integration of the workflow across different disciplines, further increasing efficiency and productivity for steel fabricators and connection designers.

ACKNOWLEDGEMENTS

This research was funded by the Natural Sciences and Engineering Research Council of Canada, the Canadian Institute of Steel Construction, and the CISC Centre for Steel Structures Education and Research. Additional financial support was provided by the University of Alberta.

I extend my sincere thanks to Collins Steel for their invaluable contribution in sharing their existing projects and their collaborative support towards this research project.

Table of Contents

List of Tables

List of Figures

1 INTRODUCTION

1.1 Statement of Research Problem

The design of steel buildings typically involves a multi-disciplinary team, with structural engineers being responsible for all structural aspects of the design. However, steel fabricators and connection designers are typically brought in only after the principal structural systems have been designed, limiting their influence on member selection and steel framing. As a result, inefficiencies can arise, such as the need for additional reinforcement, which can lead to increased costs for the client. Connection designers, therefore, focus on designs that can minimize costs associated with fabrication and erection, but accurately assessing the efficiency and cost implications of these designs can be a significant challenge.

One major difficulty is the laborious and intricate process involved in designing connections for buildings. The connection designer utilizes finished drawings to design connections at every joint, generally followed by sketching connection details that the detailer interprets and manually enters into a 3D model. Although designing certain connections, such as shear connections, may be relatively straightforward, the entire process needs to be repeated for every connection on a building project, which can mean thousands of connections. While many steel fabricators have implemented their own methods to streamline the process, determining if the final design of the building represents the most cost-effective solution remains challenging and time-consuming.

These challenges are particularly pertinent to the steel construction industry, which plays a critical role in contributing to the economies of countries like Canada and the United States. For example, in Canada, the steel industry employed about 35,000 workers and added $11 billion to the GDP in 2007 (Canadian Industry Program for Energy Conservation et al., 2007). In the United States, structural steel accounted for 46% to 58% of the construction material market share between 2009 and 2017 (AISC, 2018) and contributed to approximately 8% of the construction market (Grand View Research, 2018). On a global scale, the construction industry is a major driver of economic growth, valued at $10 trillion in 2015 and projected to reach $14 trillion by 2025. However, despite its potential for growth, the industry has been plagued by long-standing productivity issues (McKinsey Global Institute, 2017). These issues stem from various sources, including the unique and highly variable nature of construction projects, the reliance on paper-based drawings and documentation, and the fragmentation of work and activities involving different disciplines and stakeholders. The involvement of multi-disciplinary teams can sometimes lead to clashes among their respective methodologies, causing delays and conflicts in the project's execution. Moreover, errors and omissions in paper-based design documents, coupled with communication gaps among stakeholders, can exacerbate these conflicts and lead to further schedule conflicts, project delays, and unanticipated field costs (Rahman, 2014; Ruby, 2008).

The McKinsey Global Institute recommends that to overcome industry challenges, such as productivity constraints, key aspects of the architectural, engineering, and construction (AEC) industry, including design, engineering, and technology, should be improved. According to McKinsey, a potential 25% increase in productivity resulting from such improvements could significantly boost the industry's growth and the global economy as a whole (McKinsey Global Institute, 2017). Responding to cost pressures, efficiency concerns, and the need to meet sustainability targets, the construction industry is increasingly adopting new technologies in design and construction (Loosemore, 2014). Construction technologies encompass a wide range of tools, machines, and modifications that can address specific goals, functions, or problems. By leveraging these technologies, the industry can improve performance and productivity in the AEC sector, responding effectively to present and future demands (Duncan et al., 2018; Sepasgozar and Davis, 2018).

This research aims to develop a robust optimization tool capable of designing and detailing steel shear connections. The tool should be able to consider the preferences of steel fabricators, be flexible enough to allow for changes, and produce a multi-set of potential options for connection designers to choose from or work off of. The tool should also be able to output design results and have a level of automation that reduces the workload on the designer. Ultimately, this research seeks to improve the efficiency and cost-effectiveness of connection design, benefiting not just the steel construction industry but the broader construction industry as well.

1.2 Objectives and Scope

The objective of this research is to explore the potential of utilizing emerging technologies, including parametric design, generative design, and metaheuristics, to develop an optimization tool for steel connection design. The tool aims to enhance automation and decision-making in the connection design process, resulting in an end product that exceeds the capabilities of traditional methods under time and budget constraints.

It is crucial to understand that this research operates under the assumption that the principal structural elements of the building framework have already been designed by the structural engineer. While the ideal optimization of a steel building design would include the simultaneous optimization of steel framing geometry, member sizes, and connections to minimize overall cost and produce a more efficient design, the scope of this research is limited to optimizing connections only after the building framework design has been finalized.

In order to achieve the objectives, the research involves the following steps:

1. A literature review on the current state of the art in optimization techniques for steel connection design and broader structural engineering.
2. The development of an optimization tool that incorporate metaheuristics as well as parametric and generative design techniques.
3. The validation of the tool through case studies with previously designed building projects in collaboration with steel fabricators.
4. The analysis of the results to determine the effectiveness of the tool in achieving the objectives.

The research is conducted in collaboration with steel fabricators to ensure that the tool can be tailored to their specific needs and preferences, as well as to provide assurance of its effectiveness when applied to real projects. The goal of this research is to provide steel fabricators with a tool that streamlines the connection design process, reduces the workload on designers, and results in more efficient and cost-effective designs.

1.3 Organization of Report

This M.Sc. thesis is structured into six chapters. Chapter 2 is a comprehensive literature review on cost estimation, metaheuristics, and computational design approaches within the field of structural engineering and steel fabrication. In Chapter 3, the optimization tool and its framework are presented, highlighting the key components involved. Chapter 4 delves into the design and optimization of the connections, along with an in-depth examination of the overarching optimization procedure employed in the tool. In Chapter 5, the case studies conducted with the optimization tool are explored, and the corresponding results are discussed. Finally, Chapter 6 summarizes the conclusions drawn from this research project and recommendations for future studies.

2 LITERATURE REVIEW

2.1 Introduction

This chapter offers an overview of the existing technological challenges in the steel construction industry and current cost estimating techniques. Furthermore, this chapter provides an examination of the optimization and generative design techniques that are commonly used in both the AEC industry and steel connection design.

2.2 Cost Estimation and Optimization

The steel construction industry faces persistent challenges that are extensively documented in literature. These include potential cost overruns and delays, limited interaction with suppliers despite high dependence, a strong focus on repetitive processes, frequent design changes during construction, and a high degree of diversity among disciplines, leading to fragmentation, among other issues. (Dubois and Gadde, 2002; Håkansson and Ingemansson, 2013; Kannimuthu et al., 2018; Muñoz-La Rivera et al., 2021; Shen and Lin, 2014; Yap et al., 2019; Yeganeh et al., 2019; Zidane and Andersen, 2018) In recent years, the literature has seen the emergence of methodologies and technologies that can manage vast amounts of information to optimize design and construction processes. These new methods aim to reduce economic losses, minimize design time, avoid conflicts during construction, and enable model simulations. Muñoz-La Rivera et al. (2021) suggest that these emerging tools, such as artificial intelligence, building information modelling, and cloud computing, offer a departure from traditional ways of working.

According to Evers and Maatje (2000), the cost of a structure can be categorized as shown in Figure 2-1. They note that pre-design, detailed design, detailing, materials, fabrication, coating, and erection all have costs that are associated to the connections. They emphasize 50% of the total costs in a steel structure can be related to its connections and underscore the significance of conducting a thorough evaluation of connections throughout the stages of design to achieve cost savings.

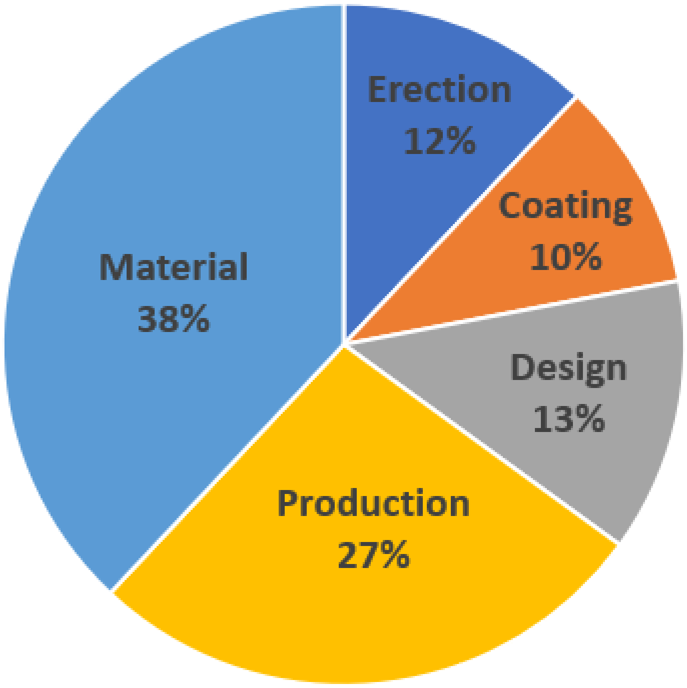
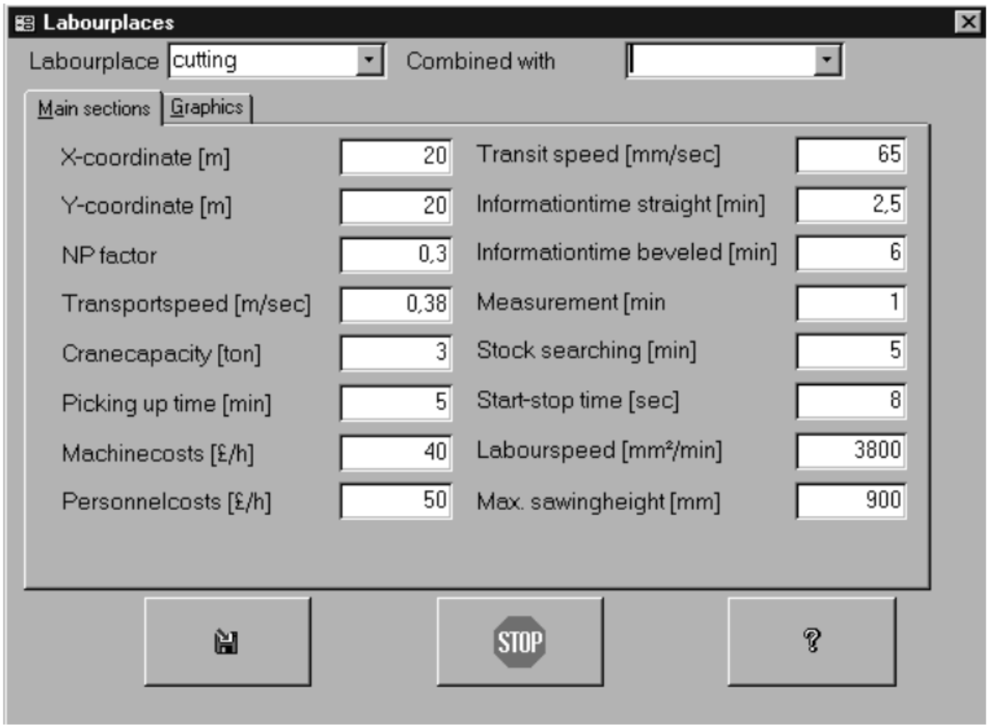


Figure 2-1: Cost breakdown of an average steel structure (Evers and Maatje, 2000)

Evers and Maatje's study focuses on ICCS-TVC, a software package designed to estimate steel structure costs based on the infrastructure and production capabilities of a specific steel fabricator. This software package comprises four distinct modules:

1. A simple "manual" tool that allows for quick cost estimations based on user inputs such as material, labour hours, and productivity.
2. A 3D graphical interface that provides a visual representation of the steel structure and its connections.
3. A connection generator module that contains nine types of connections, enabling estimation of the required plates, bolts, and welds for specific joints. This module does not design connections, but rather determines a rough estimate of the total number of parts required for construction of the steel structure, considering various factors such as drilling, shearing, cutting, and other related tasks, which can later be used for cost estimation.
4. A production cost engine that determines labour costs associated with the structure based on labour and material costs provided by the user, as shown in Figure 2-2.



Labourplace cutting   Combined with

X-coordinate [m]	20	Transit speed [mm/sec]	65
Y-coordinate [m]	20	Informationtime straight [min]	2,5
NP factor	0,3	Informationtime beveled [min]	6
Transportspeed [m/sec]	0,38	Measurement [min	1
Cranecapacity [ton]	3	Stock searching [min]	5
Picking up time [min]	5	Start-stop time [sec]	8
Machinecosts [£/h]	40	Labourspeed [mm^2/min]	3800
Personnelcosts [£/h]	50	Max. sawingheight [mm]	900

Figure 2-2: User interface for inputting production and labour costs of a steel fabricator (Evers and Maatje, 2000)

To evaluate the potential of an integrated design of a portal frame, a case study was carried out utilizing 3D modelling and analysis programs in conjunction with the estimating software, ICCS-TVC. This study demonstrated the advantages of employing modern technology to enhance the competitiveness of steel fabricators (Evers and Maatje, 2000).

Structural optimization primarily aims to minimize costs, which is typically achieved by optimizing an objective function, such as reducing the weight of the structure. Nonetheless, this objective function usually overlooks the expenses incurred during the fabrication and erection of the structure. Pavlovčič et al. (2004) present an objective function that considers the self-manufacturing costs of the entire structure. This is achieved by utilizing a cost function that encompasses all critical fabrication and erection activities that can be defined by the user and easily adapted to different production lines. The cost function takes into account various aspects of the structure's overall cost that are typically overlooked in structural optimization. These include welding costs for the elements, expenses incurred during welding manufacturing and material procurement, assembly and tack welding costs, expenses for painting the elements, surface preparation, hole formation, as well as transportation and erection costs, among others (Pavlovčič et al., 2004). To evaluate the effectiveness of their cost function optimization approach, the authors performed a case study on a two-storey, two-bay frame subjected to both gravity and lateral loads. The assessment involved comparing their approach to a classical design method, which aims to use the lightest cross-sections, and a simple volume optimization technique, where the goal is to minimize the amount of steel used. The case study showcases the efficacy of using the cost function approach to minimize cost as shown in Figure 2-3. A similar case study was conducted on a four-storey, five-bay frame which produced similar results.

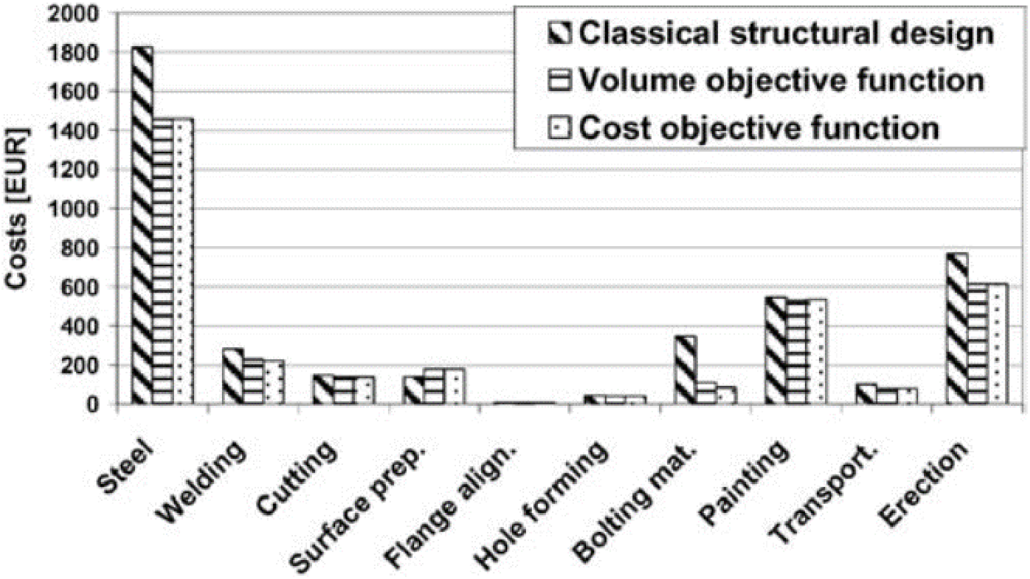


Figure 2-3: Comparison of cost contributions as a result of three different calculations of two-storey, two-bay frame (Pavlovčič et al., 2004)

It is important to note that design decisions made during the early stages of the design process have a far greater impact on cost compared to later decisions (Barg et al., 2018; Ruby, 2008). To address this issue, Barg et al. (2018) propose an integrated steel design method that streamlines the process. The method involves uploading a structural model consisting of a frame layout with preliminary member sizes and preferred connection types, and then automatically detailing the frame connections to produce a bill of quantities suitable for fabrication. Unit rates maintained by suppliers are then used to estimate the total installed cost. Finally, cost feedback is provided in near real-time, allowing engineers to make informed design decisions that minimize cost without compromising the design (Barg et al., 2018). The flowchart of the optimization process is presented in Figure 2-4.

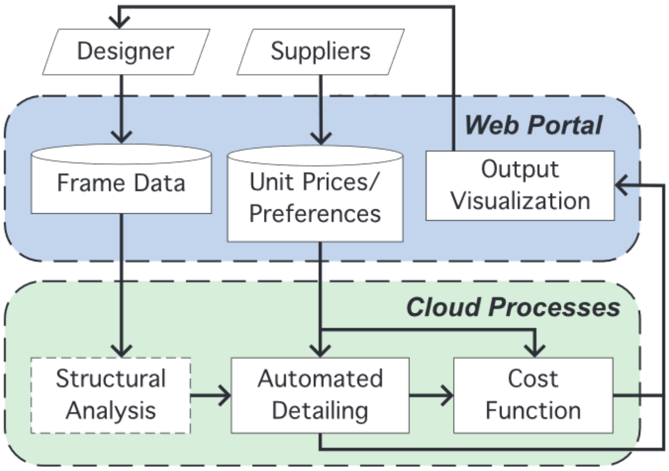


Figure 2-4: Proposed analytical method to estimate total installed cost of steel frames during early design (Barg et al., 2018)

This method was demonstrated on two example frames: a one-storey, one-bay frame and a five-storey, five-bay frame. The near real-time cost feedback provided valuable insights for decision making and resulted in the identification of lower-cost alternatives. These alternatives were achieved through adjustments to the steel section sizes and connection types of the frames during the optimization process (Barg et al., 2018).

During the later stages of design, an activity-based cost estimation approach can be employed to comprehensively examine a product design and its associated supply chain processes. This can lead to more precise cost estimation and better overall cost analysis (Aram et al., 2014). H'mida et al. (2006) conducted a study in which the manufacturing cost of a product was estimated through the modelling of resources required for each operation. These individual costs were then aggregated to determine the overall cost of the product's manufacturing process. To achieve this, a product model was developed by identifying the various available operations and alternative machine uses for each feature (H’mida et al., 2006). A similar process was used to develop the cost function for the fabrication and erection of a steel frame and its joints by Barg (Barg, 2020). In order to carry out a detailed and precise cost analysis, it is necessary to gather data related to the product, process, projects, functions, and cost drivers from various sources. As discussed in the literature, this data can help to identify the cost driving parameters and their impact on the total cost of each activity. It is crucial to assess the quality of the collected data to ensure its measurability, reliability, and completeness, as this is a key factor in establishing accurate cost functions (Aram et al., 2014; Cavalieri et al., 2004).

2.3 Metaheuristics in Structural Engineering

Structural optimization is a discipline within structural engineering that approaches structural design as a decision-making problem. The aim is to select the optimal course of action from multiple alternatives. To achieve this, decision-making problems are modeled to minimize or maximize an objective function, which measures the quality of the solution under certain limitations. Decision variables, such as the steel profile or thickness of a concrete slab, are adjusted to obtain the optimal solution while ensuring that constraints, such as limitations on resources or other factors, are met. The ultimate goal is to identify the best possible values of decision variables that lead to the optimal solution, where the objective function attains its extremum value and all constraints are satisfied (Saka et al., 2016; Singh and Choudhary, 2021).

To understand why traditional mathematical optimization is not appropriate for structural design applications, it is essential to consider some key limitations. First, these techniques assume continuous design variables, which is problematic as design variables in structural design are typically discrete. For example, cross-sectional dimensions of an angle connection could theoretically be any real number within an upper and lower bound, but in practice, the section is chosen from a list of available steel profiles with discrete dimensions. Second, most mathematical optimization techniques require gradient computations of the objective function and constraints, which can be challenging as constraint functions are often not continuous, and gradients may not exist. Additionally, these functions may be mathematically complex, making them difficult to model accurately in an optimization problem. Finally, these techniques require an initial estimate to start the iterations, and the performance of the algorithm is heavily influenced by this initial guess, often leading to local optima instead of the global optimum. (Mei and Wang, 2021; Saka et al., 2016; Saka and Geem, 2013; Singh and Choudhary, 2021).

According to Saka et al. (2016), metaheuristics is an “iterative generation process which makes use of certain guides in the search process of the design domain” (Saka et al., 2016). This method is more appropriate for structural design problems, as it overcomes many of the challenges associated with traditional mathematical optimization techniques. Unlike deterministic methods that always converge to the same solution with the same initial guess, metaheuristics introduces randomness in its search process, potentially resulting in a different outcome each time it is executed. As a result, metaheuristics is considered a stochastic optimization approach.

Metaheuristic algorithms do not rely on gradient information or the convexity of objective functions and constraints. Instead, they employ probabilistic transition rules rather than deterministic rules, and they do not necessitate an explicit relationship between the objective function and constraints. These algorithms are based on stochastic search strategies, making them highly effective and versatile in handling the combinatorial explosion of possibilities. Metaheuristic algorithms emulate strategies observed in nature, drawing inspiration from biology and laws of physics, in order to direct the search process aiming to efficiently explore the search space using these mechanisms to find near-optimal or global optimum solutions (Khanduja and Bhushan, 2021; Luke, 2013; Singh and Choudhary, 2021; Voß, 2001; Yang, 2010; Yang, 2010).

In the field of structural engineering, problems often involve non-linear, non-differentiable, or noisy objectives, which make it beneficial to use metaheuristics. While metaheuristics cannot guarantee optimal solutions, they typically produce solutions that are close to optimal with minimal computational effort. However, the ultimate goal of structural engineering design is to develop a design that is cost-effective, while complying with design code provisions (Zavala et al., 2014). This results in optimization problems that have more than one objective, which are known as multi-objective optimization problems (MOOPs). In contrast to single-objective optimization problems, different solutions in MOOPs will produce different trade-offs, meaning that a solution that improves one objective may worsen another. As a result, the Pareto front, also known as the Pareto-Optimal front, is a set of optimal solutions rather than a singular optimal solution. The Pareto front shows the balance between various objectives and provides decision-makers with a range of optimal alternatives, each with its own advantages. The size and shape of the Pareto front depend on the problem and the number of objectives and can range from a single point to a curve, surface, or high-dimensional shape. Using multi-objective optimization techniques can generate a range of optimal solutions, enabling designers to select the one that best aligns with their specific needs and requirements (Deb, 2001; Deb et al., 2002; Duong et al., 2022; Miettinen, 1998; Sanchis et al., 2008).

As a form of metaheuristics, evolutionary optimization (EO), also known as genetic algorithms (GA), have become increasingly popular (Salehi and Burgueño, 2018). EOs attempt to mimic the bio-mechanisms of mutation, recombination, and natural selection, all of which form the basis of evolution (Tian et al., 2017). These algorithms employ a population-based approach, which involves multiple solutions in each iteration, leading to the creation of a new population of solutions in every cycle. Unlike classical optimization algorithms that update one solution per iteration, EO procedures utilize a population-based approach that enables them to identify multiple optimal solutions, facilitating multi-objective optimization (Deb, 1999, 2011). One of the strengths of EO algorithms is their flexibility, which allows users to choose appropriate operators and problem-specific information to address specific optimization problems. However, this flexibility also places a responsibility on the user to select tangible operators that can produce an efficient and consistent search. Nevertheless, the advantages of employing a flexible optimization procedure over rigid and specific algorithms are particularly relevant in tackling complex real-world optimization challenges, including non-differentiable objectives and constraints, non-linearities, multiple optima, large problem sizes, uncertainties in decision variables, and mixed variable types, all of which are prevalent in the structural engineering optimization (Deb, 2011; Goldberg, 1989).

Evolutionary multi-objective optimization can be classified into three categories based on when decision-maker (DM) preferences are incorporated: *a priori*, *interactive*, and *a posteriori* approach. In the *a priori* approach to decision making, the DM preferences are integrated into the optimization process prior to conducting the search. Several techniques, such as weighted sum, utopia point, goal attainment, or physical programming, are commonly used for this purpose (Sanchis et al., 2008). While an *a priori* method can be effective and efficient when the DM preferences can be accurately captured in a mathematical model, this is often not the case. The *interactive* approach to decision making involves the gradual integration of DM preferences throughout the optimization process. This approach allows DMs to “learn” about the problem and refine their preferences as needed, guiding the search towards optimal solutions while avoiding non-ideal ones. However, a significant drawback of this method is that it often requires the DM to be actively engaged and involved in the search process (Purshouse et al., 2014). The *a posteriori* approach to decision making involves incorporating DM preferences after the search process is complete. This approach first involves approximating the Pareto front, followed by the presentation of a set of trade-off solutions from which the DM can choose their preferred solution. The *a posteriori* approach is often effective for MOOPs with 2 or 3 objectives, as a reliable approximation of the Pareto front can be obtained and presented to the DM. However, this method becomes less effective as the number of objectives increases in MOOPs (Purshouse et al., 2014).

The *a posteriori* approach has been widely used in literature to optimize various structural design problems (Alkhadashi et al., 2022; Barraza et al., 2017; Duong et al., 2022; García-Segura and Yepes, 2016; Gonçalves et al., 2015; Hasançebi et al., 2009; Kripakaran et al., 2011; Neves et al., 2022; Stanković et al., 2012; Winslow et al., 2010). Some specific examples are discussed below.

Kripakaran et al. (2011) presented a computational approach that can enhance the decision-making process for designing moment-resisting steel frames. By utilizing genetic algorithms, the authors conducted a comprehensive trade-off study of a five-bay, five-storey frame that accounts for the cost of connections, in addition to the cost of the steel frame, in their optimization model. This cost function considers whether the connections are rigid or hinged, introducing additional variables beyond the typical optimization problem that focuses only on columns and beams as decision variables. To begin the optimization process, the minimum weight solution is first identified, where all connections in the frame are rigid. To do so, an optimization procedure is employed to determine the section sizes that meet both strength and serviceability requirements, while minimizing weight as outlined in Figure 2-5 and Figure 2-6, respectively.

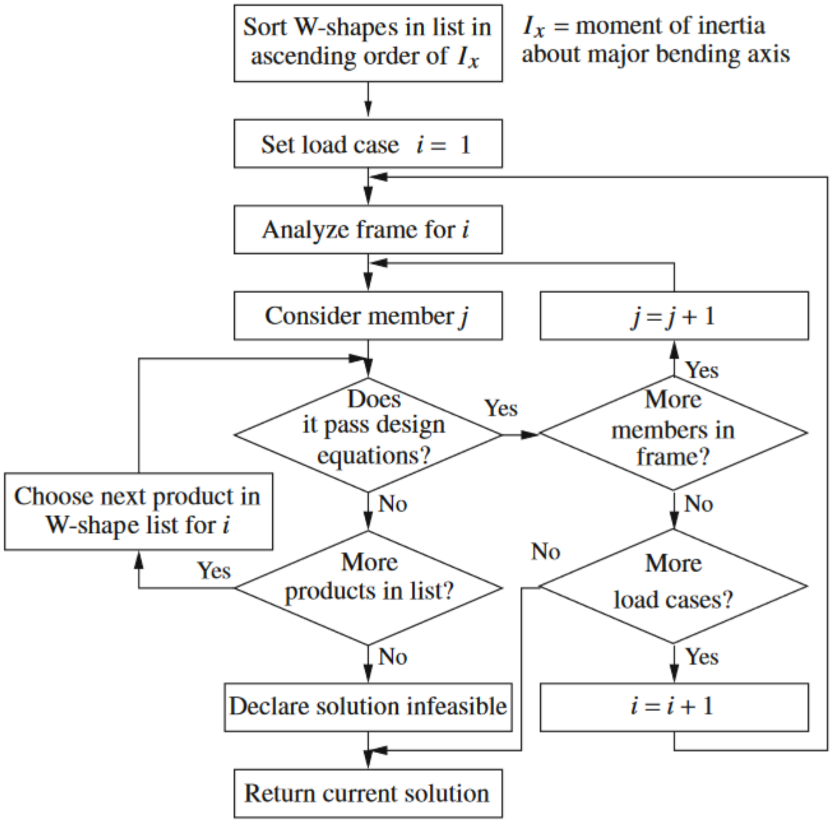

Sort W-shapes in list in ascending order of I_x
I_x = moment of inertia about major bending axis
Set load case $i = 1$
Analyze frame for i
Consider member j
$j = j + 1$
Yes
Does it pass design equations?
Yes
More members in frame?
Choose next product in W-shape list for i
No
No
No
Yes
More products in list?
More load cases?
No
Yes
Declare solution infeasible
$i = i + 1$
Return current solution

Figure 2-5: Algorithm for correcting solutions that violate strength requirements (Kripakaran et al., 2011)

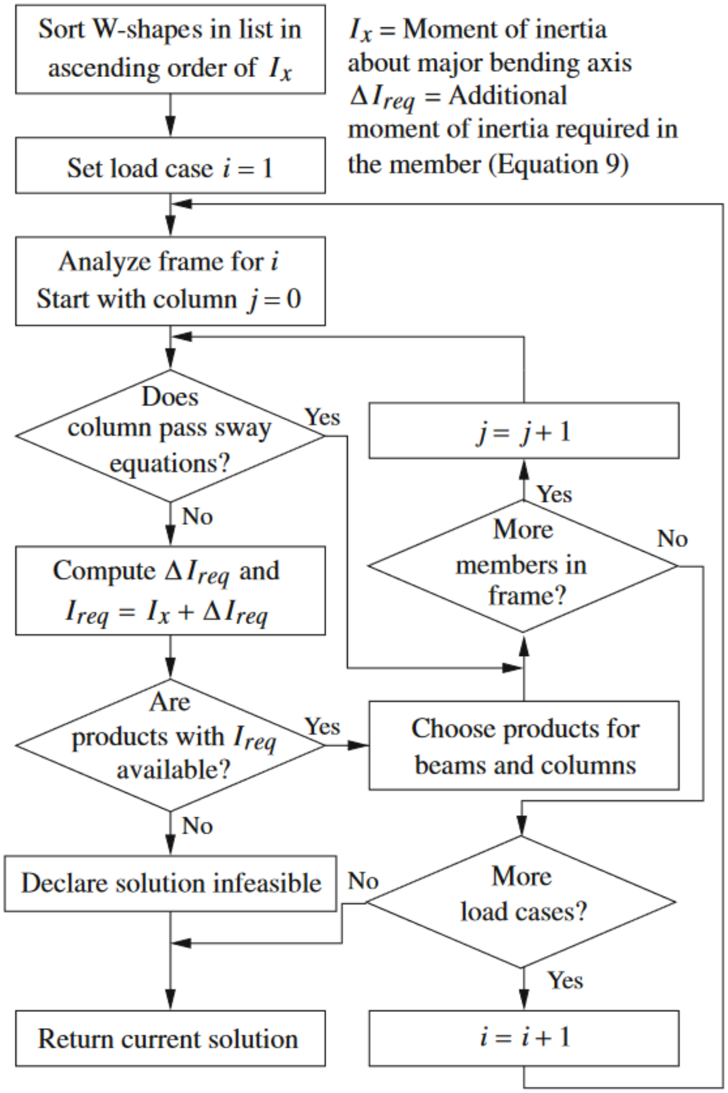

Sort W-shapes in list in ascending order of I_x
I_x = Moment of inertia about major bending axis
ΔI_{req} = Additional moment of inertia required in the member (Equation 9)
Set load case $i = 1$
Analyze frame for i
Start with column $j = 0$
Does column pass sway equations?
Yes
No
$j = j + 1$
Yes
More members in frame?
No
Compute ΔI_{req} and $I_{req} = I_x + \Delta I_{req}$
Are products with I_{req} available?
Yes
Choose products for beams and columns
No
Declare solution infeasible
No
More load cases?
Yes
Return current solution
$i = i + 1$

Figure 2-6: Algorithm for correcting solutions that violate sway constraints (Kripakaran et al., 2011)

To create a trade-off curve, depicted in Figure 2-7, that effectively balances the number of required rigid connections, r_{req}, with the total cost, a genetic algorithm (GA) is employed. The GA is utilized to identify the optimal location of rigid connections, given a specified r_{req}. Multiple runs of the GA are conducted at varying r_{req} values, while section sizes are optimized based on the algorithms presented in Figures 2-5 and 2-6. It is important to note that each solution must meet certain strength and sway constraints; failure to do so results in the application of a penalty cost that significantly reduces the fitness of the individual solution.

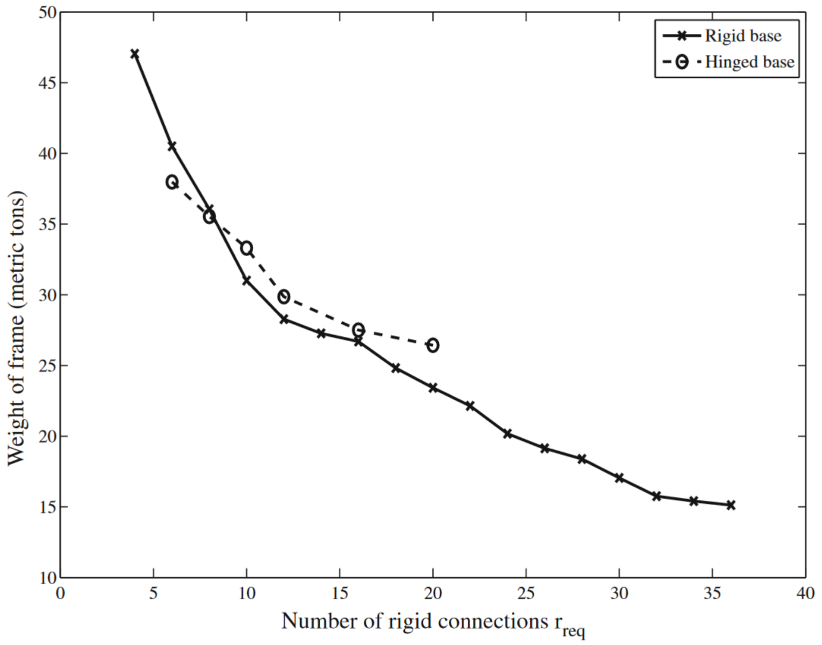


Figure 2-7: Trade-off between weight of frame and number of rigid connections (Kripakaran et al., 2011)

The case study identifies that there exists an optimal number of rigid connections that minimizes the total cost. Moreover, the approach showcases the ability to provide a range of alternative options that can be further refined or selected, enhancing its overall practicality and utility (Kripakaran et al., 2011).

Barraza et al. (2017) conducted a study demonstrating the efficacy of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) and Particle Swarm Optimization (PSO) in designing moment-resisting steel structures subjected to earthquake loading in both two and three dimensions. The NSGA-II is a widely-used GA that leverages the principles of evolution to guide its optimization process (Deb et al., 2002), while the PSO is a swarm optimization method inspired by the collective behaviour of birds flocking or fish schooling (Kennedy and Eberhart, 1995). A more detailed discussion on both these algorithms can be found in Section 3.5. The objective of the study was to achieve an optimal solution that minimizes structure weight and inter-storey drift, while also meeting strength requirements outlined by the Load and Resistance Factor Design (LRFD) specifications provided by the American Institute of Steel Construction (AISC, 2010). In the 2-D case study, a four-storey, two-bay frame with a load configuration as depicted in Figure 2-8 was utilized, and 256 W-shape sections from the AISC database were considered. Notably, for this frame only four distinct W-sections could coexist at any given time. Specifically, columns C1-C6 were required to have the same section, columns C7-C12 were required to have the same section, beams V1-V4 had to have the same section, and beams V5-V8 were also required to have the same section. Using the NSGA-II, an initial population of 100 steel frames was generated, with two objectives specified: minimize the structural weight, F_{O1}, and minimize the maximum inter-storey drift, F_{O2}.

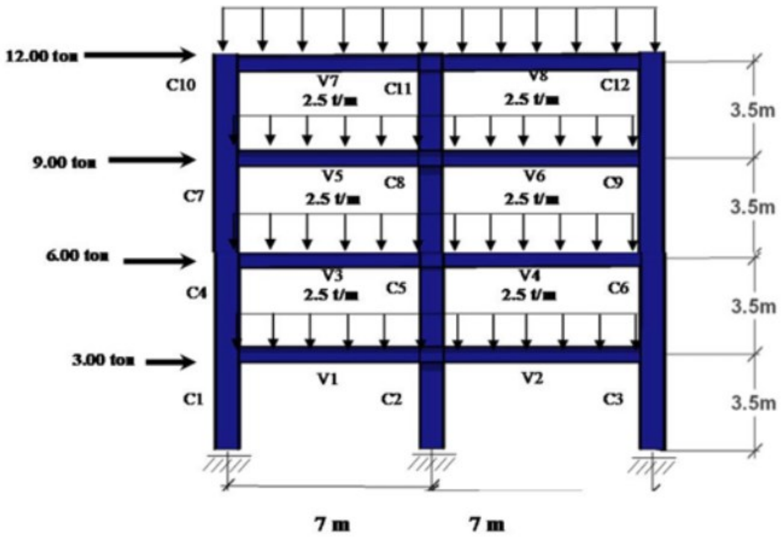
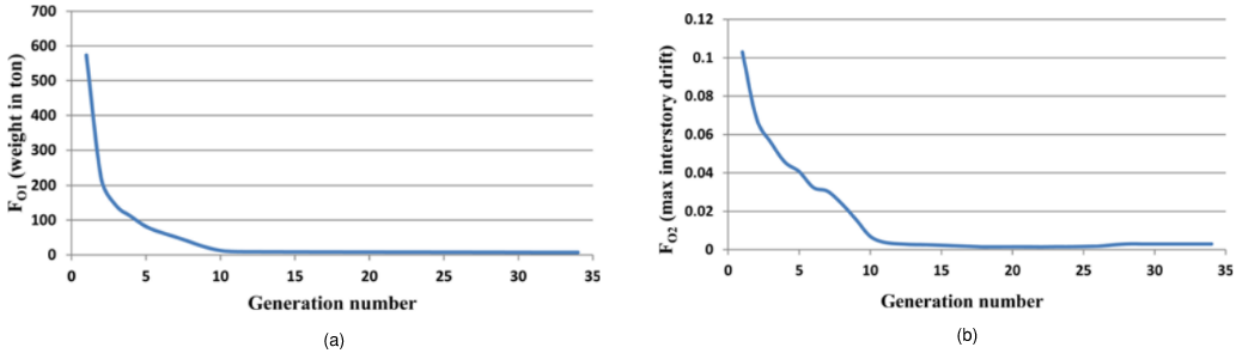


Figure 2-8: Geometric characteristics and lateral loads for structural 2D steel frame (Barraza et al., 2017)

The results of the study demonstrated the effectiveness of NSGA-II in achieving moment-resisting steel frame building designs that minimize the objective functions of interest, namely weight and structural performance as measured by the maximum inter-storey drift. This is illustrated in Figure 2-9 by the decrease in the objective functions as the number of generations increases.



(a)

(b)

Figure 2-9: Average functions for 2D steel frame: (a) structural weight, F_{O1}; (b) maximum inter-storey drift, F_{O2} (Barraza et al., 2017)

The PSO algorithm also yielded comparable results for the 2-D steel frame. Furthermore, both the NSGA-II and PSO algorithms exhibited promising outcomes in minimizing the same objective functions in a 3-D case study. In conclusion, Barraza et al. (2017) assert that GAs and PSO are valuable tools in structural optimization.

Duong et al. (2022) explored the use of GAs to optimize structural steel connections based on their demand-to-capacity ratios. To accomplish this, Rhino3D (McNeel and Associates, 2023b) was employed to create a parametric model of various steel shear connections, including shear tabs, end plates, bolt–bolted angles, and welded–bolted angles. Subsequently, NSGA-II was utilized to optimize these connections based on a design force and to maximize the demand-to-capacity ratio, while adhering to strength constraints according to CSA-S16 (CSA, 2019) and AISC-360 (AISC, 2016) specifications. The authors conducted two solutions: a standards-based approach utilizing code equations and a finite element-based solution. Notably, in a shear end plate case study, the two different methods yielded similar results to those found in the Handbook of Steel Construction (CISC, 2021) showcasing its potential. The authors concluded that the use of GA in optimization of structural steel connections is effective; however, a finite element-based approach should be reserved for more complicated connections due to its computational cost (Duong et al., 2022).

To minimize the cost, reduce CO_2 emissions, and maximize the overall safety factor of a three-span continuous post-tensioned box-girder road bridge, García-Segura and Yepes (2016) suggested utilizing a multi-objective harmony search. In addition to exploring decision variables and assessing objective functions, the optimization procedure integrated CSiBridge (Computers and Structures Inc., 2022) for finite element analysis and MATLAB (The MathWorks Inc., 2022) for limit states verification, as illustrated in Figure 2-10. The optimization procedure generates a set of optimal solutions that are analyzed to reveal the correlation between objectives, identify the variables that can improve safety, and highlight the critical ultimate limit states. The findings indicate that using metaheuristics to optimize cost is a viable approach to achieving designs with lower CO_2 emissions, provided that the cost and CO_2 emission criteria result in reduced material consumption (García-Segura and Yepes, 2016).

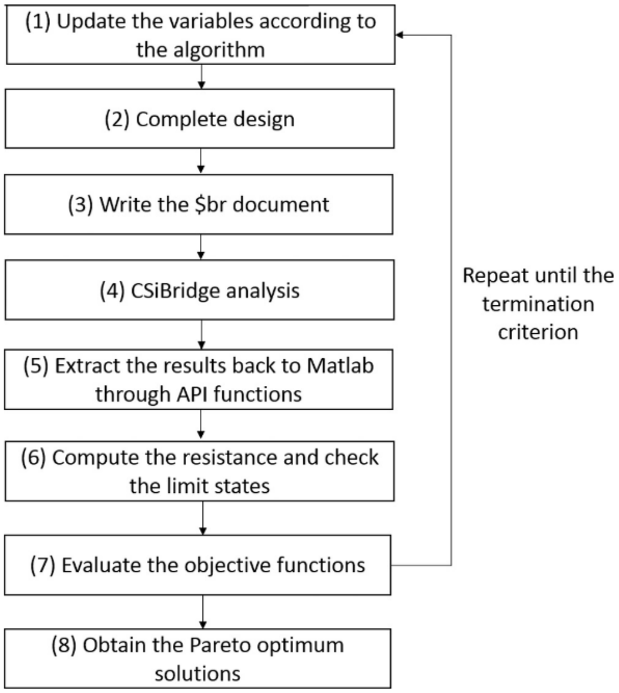

(1) Update the variables according to the algorithm
(2) Complete design
(3) Write the $br document
(4) CSiBridge analysis
(5) Extract the results back to Matlab through API functions
(6) Compute the resistance and check the limit states
(7) Evaluate the objective functions
(8) Obtain the Pareto optimum solutions
Repeat until the termination criterion

Figure 2-10: Flowchart for structural analysis and optimization (García-Segura and Yepes, 2016)

2.4 Parametric and Generative Approaches within Computational Design

Parametric design is a modelling approach where the relationships among various parameters of an object, such as shape, dimensions, and positioning, are predetermined. This approach allows the designer to easily modify some parameters, with the rest of the model being adjusted accordingly by the software based on predetermined rules. However, these permutations must be evaluated to ensure they meet the project's objectives (Jabi, 2013). Generative design has emerged as a new approach that allows designers and engineers to specify parameters such as materials, spatial constraints, manufacturing methods, and cost limitations to create algorithms or rule sets. This approach facilitates the exploration of various permutations of the model automatically, and the software generates the best design alternatives according to the objectives proposed by the user. In contrast to parametric design, where the user can easily modify the model's geometry for later evaluation, generative design involves the software taking inputs, evaluating them, and creating the best alternatives that meet the user's requirements.

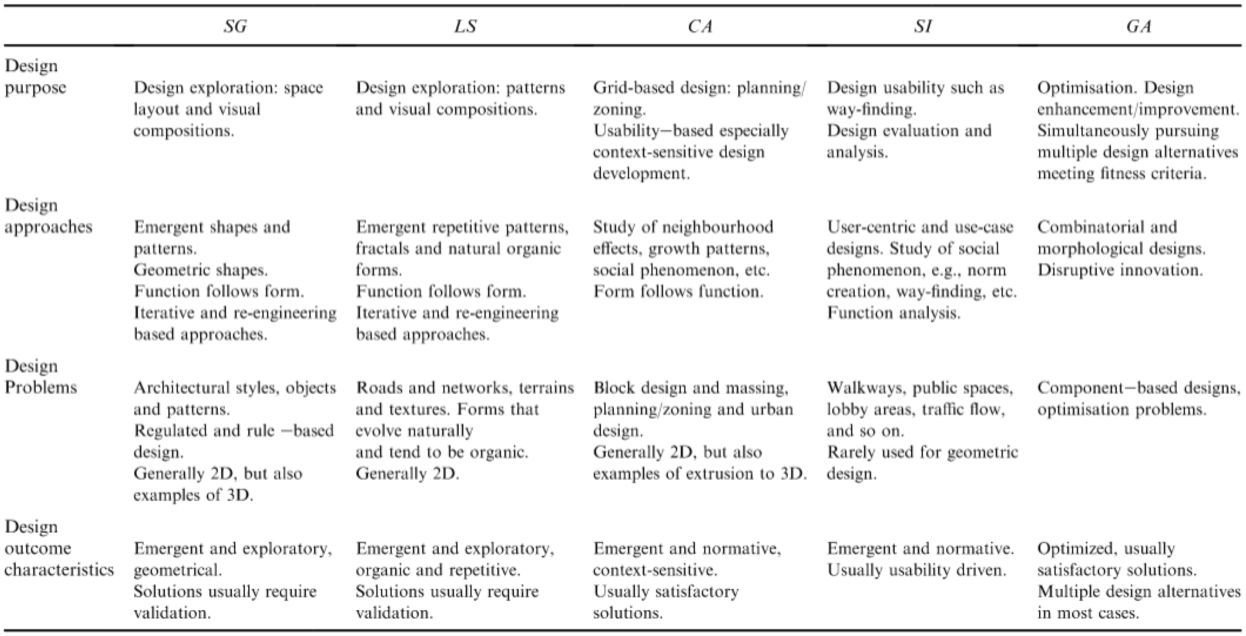
Generative design (GD) involves processes that use specific principles like genetic algorithms and topology optimization to optimize designs through a unique set of rules and algorithms. However, the open-ended nature of generative design often yields an overwhelming number of results, many of which can be highly unpredictable unless appropriate constraints are applied to limit the range of possibilities. Thus, an effective generative computational framework must include both mechanisms to generate possibilities and constraints to ensure that results are relevant and useful. When properly developed, this intelligent system approach can inform early design decisions and lead to more efficient and effective designs (Johan et al., 2019).

Oxman (2006) explains that the emergence of computational design (digital design) has introduced a new form of design that differs from traditional paper-based design. In this context, designers interact not only with the freeform of design but also with digital constructs such as design representations and elements. The level of control that designers have in computational design is dependent on their computational literacy, which has become a critical factor (Oxman, 2006). Furthermore, designers have taken on a new role as tool builders in computational design environments, with opportunities to develop tools that generate designs through scripting and editing computational mechanisms, in addition to generating designs themselves (Aish, 2003). Oxman (2006) emphasizes that generative design tools require an interactive module that allows designers to control and modify the GD mechanisms to achieve desired solutions. Custom design tools and compound models that integrate various digital design models are also essential, creating even greater opportunities for design (Oxman, 2006).

Design is widely considered a co-evolutionary process, whereby the problem and solutions mutually guide each other as the designer engages with the design problem (Dorst, 2019; Dorst and Cross, 2001; Lawson, 2010; Schon, 1992). Typically, the designer begins with an ill-defined problem, and as the design activity progresses, the problem and solutions evolve in tandem. This dynamic interplay, described as "reflective practice" by Schon (1992), leads to a fluid and constantly evolving view of the problem and solution as the designer interacts with the task. Gero (1998) explains how designers' past experiences and immediate task environment influence emergent design solutions, which is particularly relevant in structural engineering and steel connection design.

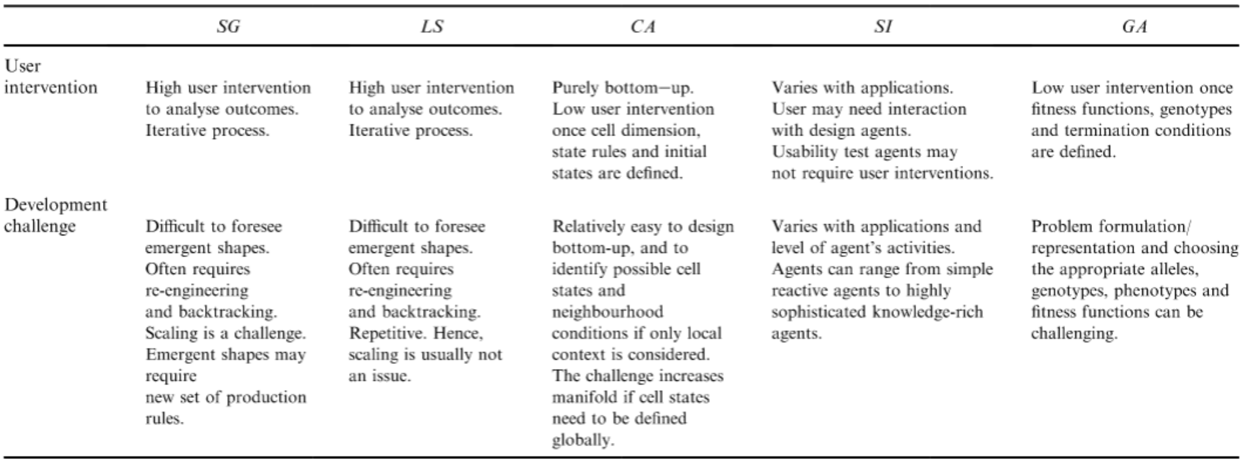
Singh and Gu (2012) reviewed the design exploration capabilities in architecture of five different GD techniques: shape grammars (SG) (Stiny, 1980), L-systems (LS) (Lindenmayer, 1968), cellular automata (CA) (Neumann, 1951; Wolfram, 2002), genetic algorithms (GA) (Holland, 1975; Sivanandam and Deepa, 2007), and swarm intelligence (SI) (Kennedy and Eberhart, 1995; J. Yang et al., 2020). They indicate that some GD techniques are more suited for specific design purposes than others. GAs, in particular, are best suited for design optimization problems that are commonly found in structural engineering and connection design applications. Table 2-1 and Table 2-2 showcase the design and system development aspects of the different GD techniques, respectively.

Table 2-1: Design aspects of generative design techniques (Singh and Gu, 2012)



	SG	*LS*	*CA*	*SI*	*GA*
Design purpose	Design exploration: space layout and visual compositions.	Design exploration: patterns and visual compositions.	Grid-based design: planning/ zoning. Usability—based especially context-sensitive design development.	Design usability such as way-finding. Design evaluation and analysis.	Optimisation. Design enhancement/improvement. Simultaneously pursuing multiple design alternatives meeting fitness criteria.
Design approaches	Emergent shapes and patterns. Geometric shapes. Function follows form. Iterative and re-engineering based approaches.	Emergent repetitive patterns, fractals and natural organic forms. Function follows form. Iterative and re-engineering based approaches.	Study of neighbourhood effects, growth patterns, social phenomenon, etc. Form follows function.	User-centric and use-case designs. Study of social phenomenon, e.g., norm creation, way-finding, etc. Function analysis.	Combinatorial and morphological designs. Disruptive innovation.
Design Problems	Architectural styles, objects and patterns. Regulated and rule —based design. Generally 2D, but also examples of 3D.	Roads and networks, terrains and textures. Forms that evolve naturally and tend to be organic. Generally 2D.	Block design and massing, planning/zoning and urban design. Generally 2D, but also examples of extrusion to 3D.	Walkways, public spaces, lobby areas, traffic flow, and so on. Rarely used for geometric design.	Component—based designs, optimisation problems.
Design outcome characteristics	Emergent and exploratory, geometrical. Solutions usually require validation.	Emergent and exploratory, organic and repetitive. Solutions usually require validation.	Emergent and normative, context-sensitive. Usually satisfactory solutions.	Emergent and normative. Usually usability driven.	Optimized, usually satisfactory solutions. Multiple design alternatives in most cases.

Table 2-2: System development aspects of generative design techniques (Singh and Gu, 2012)



	SG	*LS*	*CA*	*SI*	*GA*
User intervention	High user intervention to analyse outcomes. Iterative process.	High user intervention to analyse outcomes. Iterative process.	Purely bottom—up. Low user intervention once cell dimension, state rules and initial states are defined.	Varies with applications. User may need interaction with design agents. Usability test agents may not require user interventions.	Low user intervention once fitness functions, genotypes and termination conditions are defined.
Development challenge	Difficult to foresee emergent shapes. Often requires re-engineering and backtracking. Scaling is a challenge. Emergent shapes may require new set of production rules.	Difficult to foresee emergent shapes. Often requires re-engineering and backtracking. Repetitive. Hence, scaling is usually not an issue.	Relatively easy to design bottom-up, and to identify possible cell states and neighbourhood conditions if only local context is considered. The challenge increases manifold if cell states need to be defined globally.	Varies with applications and level of agent's activities. Agents can range from simple reactive agents to highly sophisticated knowledge-rich agents.	Problem formulation/ representation and choosing the appropriate alleles, genotypes, phenotypes and fitness functions can be challenging.

According to Abdallah et al. (2019), the field of structural engineering has been slow to adopt generative design applications for optimizing structures, in part due to limitations imposed by traditional construction processes that restrict design flexibility and result in structures with excessive material usage and negative environmental impacts. However, advancements in technology, such as 3D printers, have opened up possibilities for more complex geometries in construction and greater design flexibility. Abdallah et al. (2019) showcased the capabilities of generative design in reducing material usage for cantilevered concrete beams. Through the manipulation of variables such as beam depth (h), length (L), and step increment size (x) (illustrated in Figure 2-11), they achieved a reduction in both cost (ranging from 40% to 52%) and CO_2 emissions (ranging from 39% to 51%) per beam in comparison to a traditionally designed cantilever concrete beam (Abdallah et al., 2019).

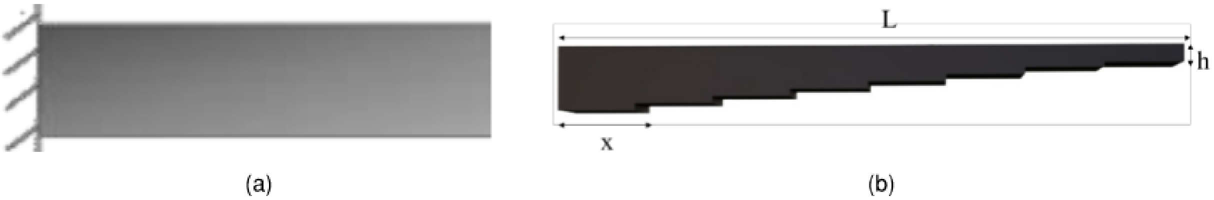


Figure 2-11: Cantilever beam designs: (a) traditionally designed; (b) optimized (Abdallah et al., 2019)

Hofmeyer and Delgado (2015) explored the use of a genetic algorithm-based method for optimizing building spatial and structural designs. The approach involved generating a population of accompanying building and spatial designs and evaluating them using the finite element method, with specific performance indicators such as total strain energy, spatial volume, and number of spaces. The findings indicate that collaborative design, through modification of one domain (spatial) to optimize another (structural), can be just as effective as monodisciplinary optimization. Additionally, the genetic algorithm-based method produced a multitude of design variations, which helps prevent overlooking design possibilities (Hofmeyer and Davila Delgado, 2015).

Martinez-Rocamora et al. (2020) introduce a novel approach for creating cost-efficient and customizable curved walls using parametric programming and 3D printing technology. They argue that conventional building design and construction methods can be costly, time-consuming, and often limit design possibilities. With parametric programming, which Lim et al. (2016) describe as a form of visual programming that enables nonspecialized users to program code in an intuitive way, Martinez-Rocamora et al. (2020) present a method for designing and fabricating curved walls based on a set of parameters that define the wall's shape and size. This approach offers designers the flexibility to create a range of shapes, from simple curves to more complex designs with multiple curves. The authors demonstrate the practicality of their method through a case study of a building in Mexico City that features curved walls designed using parametric programming and printed using a 3D printer, resulting in faster and more cost-efficient construction than traditional methods (Martínez-Rocamora et al., 2020).

Johan et al. (2019) explored using material-based constraints as a key driver in generative frameworks for designing planar space trusses made of steel, wood, and bamboo. To achieve this, they developed a generative framework using tools such as C# programming language (Microsoft, 2023), Grasshopper (McNeel and Associates, 2023a), and Galapagos (Rutten, 2019), an evolutionary solver. This framework demonstrates the potential to produce fast iterations, while ensuring the demand does not exceed the capacity of the system, making it a useful starting point for further design development. To account for solutions that do not meet this criterion, a penalty function is implemented. This function enables Galapagos to disregard such solutions during the optimization process. Figure 2-12 demonstrates how the research project was able to generate a subset of optimal planar space trusses that can be subjected to further analysis and refinement by the engineer. The generative approach can also progressively improve the intelligence of the process by evaluating solutions and informing subsequent iterations, resulting in a dynamic relationship between scripting potential, parametric environments, and traditional analyses during the development stage (Johan et al., 2019).

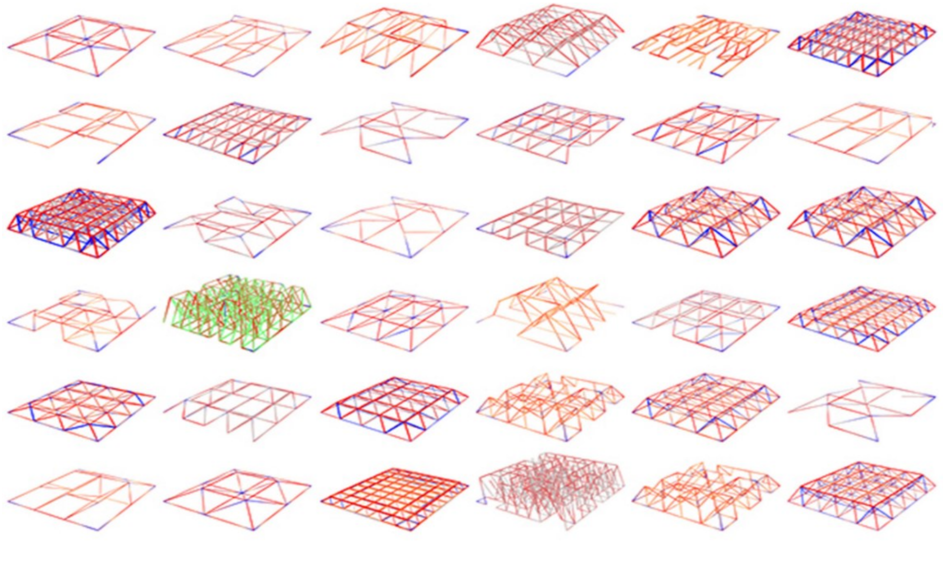


Figure 2-12: Various optimization results for planar space truss systems (Johan et al., 2019)

Rodrigues et al. (2015) proposed an approach to optimize the thermal performance of a group of buildings in an urban quarter. They used a generative design technique that combines an evolutionary strategy with a stochastic hill climbing method (Rodrigues et al., 2013a, 2013b) to generate alternative floor plans according to user preferences and design program requirements. The results of this study demonstrated the immense value of such approaches in urban and architectural design processes. Not only do they generate designs for further development, but they also provide opportunities to explore urban configurations that meet functional and performance objectives (Rodrigues et al., 2015).

According to Touloupaki and Theodosiou (2017), GD has emerged as a prominent trend in architecture over the past few decades, proving to be an invaluable tool for effectively exploring potential design options. This method provides dynamic control over geometry and components when designing forms or systems, allowing designers to find appropriate solutions to complex problems by assessing multiple variants simultaneously. Visual/graphical coding tools for design, such as Dynamo Studio for Autodesk Revit (Autodesk, 2023) or Grasshopper for Rhinoceros 3D (McNeel and Associates, 2023a), enable the implementation of parametric design concepts using visual logic, thereby automating complex tasks. The integration of parametric design, energy simulation, form optimization, and genetic algorithms in net zero energy buildings demonstrates potential for improving the AEC industry. This potential is particularly pronounced when accompanied by the advancement of more user-friendly procedures and software (Touloupaki and Theodosiou, 2017).

3 COMPONENTS OF OPTIMIZATION TOOL

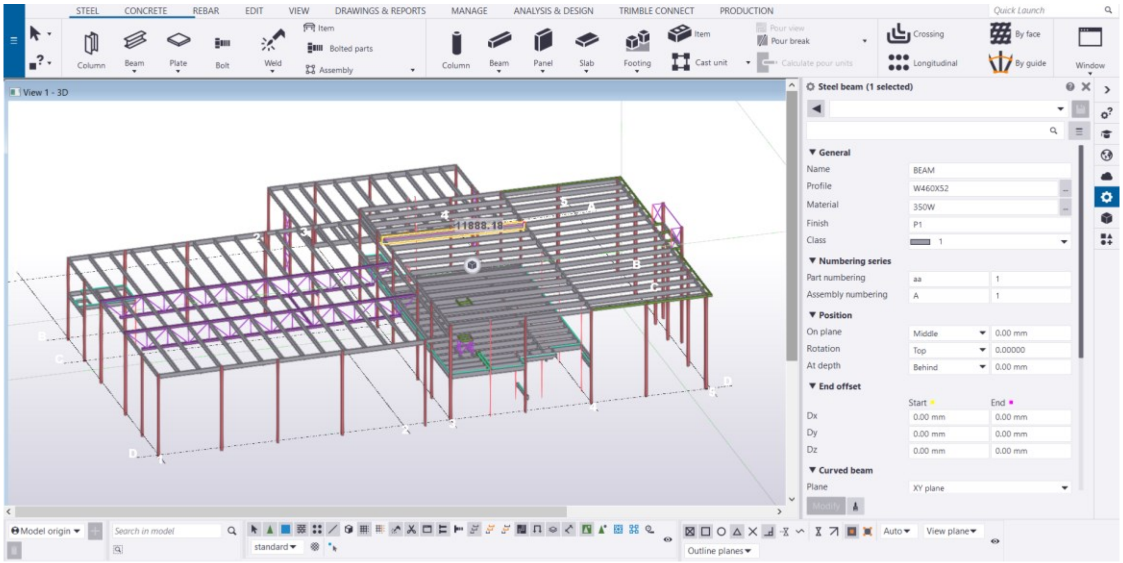
3.1 Introduction

This chapter explores the components of the optimization tool that facilitates a new approach to automate the design of steel connections in a steel building structure, taking into account material and labour costs, as well as fabricator preferences. The optimization tool is developed using Grasshopper3D (McNeel and Associates, 2023a), a visual scripting plugin for Rhino3D (McNeel and Associates, 2023b), Tekla Structures (Tekla, 2022), a widely used Building Information Modelling (BIM) software for structures, and C# programming (Microsoft, 2023), a programming language capable of accessing Tekla's application programmable interface and generating custom components within Grasshopper. To begin the connection design process using the tool, the designer uploads an existing Tekla stick model that represents the structure without detailed connections. Next, the user inputs their preferences and unit rates directly into Grasshopper. Finally, the optimization procedure is initialized, and this procedure is aimed at designing and optimizing all connections in the structure with the goal of minimizing both cost and material weight. Given that this is a bi-objective optimization challenge, the procedure generates a set of solutions for the user's evaluation. The user can then select from or refine these solutions based on their individual preferences. It is important to acknowledge that, at present, the optimization tool is limited to designing shear connections. Nonetheless, the framework detailed in this chapter provides a solid foundation for future researchers to extend the tool's capabilities to encompass various other types of connections.

This chapter also examines the cost function utilized to estimate the overall cost of the structure. Developed through a collaborative effort with an industry partner steel fabricator and drawing insights from a range of literature sources, notably the work of Barg et al. (2018). The cost function incorporates unit rates derived from the specific shop activities associated with the designated connection type. In conjunction with the user-provided labour rates, it provides an estimation of the overall cost of the structure. This estimation plays a crucial role in the second layer of optimization, facilitating the generation of optimal solutions for the user to consider.

3.2 Tekla Structures

Tekla Structures, a screenshot of which is shown in Figure 3-1, is a highly versatile structural Building Information Modelling (BIM) software that enables the modelling of structures using a wide range of building materials, such as steel, concrete, and timber. The software is a popular choice among fabricators in North America for detailing and producing shop drawings for various building projects. This research project is based on the assumption that the structure has been designed by the structural engineer, making Tekla Structures an ideal choice for the optimization tool. As steel fabricators would typically create a stick model based on the structural and architectural drawings before proceeding with connection design, the software aligns with the existing workflow of many Canadian steel fabricators. Consequently, the optimization tool can be effectively incorporated into various building projects.



Steel beam (1 selected)

General

Name	BEAM	
Profile	W460X52	
Material	350W	
Finish	P1	
Class	1	

Numbering series

Part numbering	aa	1
Assembly numbering	A	1

Position

On plane	Middle	0.00 mm
Rotation	Top	0.00000
At depth	Behind	0.00 mm

End offset

	Start	End
Dx	0.00 mm	0.00 mm
Dy	0.00 mm	0.00 mm
Dz	0.00 mm	0.00 mm

Curved beam

Plane	XY plane

Figure 3-1: Example of a model in Tekla Structures 2022

Tekla Structures has a robust Application Programmable Interface (API). In short, an API enables the communication between two or more computer programs. It allows the optimization tool to extract information from a Tekla model to any other computer programs and vice versa. Given that the optimization tool automates the design of steel connections, it is critical to access the modelling software's functionality via programming. This ensures that the optimization process can import and export information from the model as needed, facilitating a streamlined and efficient workflow. As an example, suppose a user wishes to modify all cross-section sizes in their model to a particular profile. Instead of manually altering the cross-section sizes in the Tekla model, the user can create a script that automatically adjusts the cross-section sizes upon execution.

The Tekla API provides the capability to extract beams and columns from a model as 3D vectors, which can be evaluated using linear algebra to determine the connection space at each joint. This information is mandatory in designing and optimizing the connections. Furthermore, the API facilitates the determination of parameters such as elevation difference, slope angle, skew angle, and coping requirements between the support and supported members by analyzing these vectors. Additionally, Tekla's API allows for programmatic addition, modification, or deletion of connections directly in the model, which is a valuable feature for automation in the optimization tool.

3.3 Grasshopper3D

Grasshopper3D is a visual programming language that runs directly inside Rhino3D, a commercial 3D computer graphics and computer-aided design software. Although Grasshopper is primarily used as a parametric modelling tool for architecture, it has seen a rise in popularity among the structural engineering industry to optimize simple to complex structural design problems. For the optimization tool, Grasshopper serves as a user interface due to its simplistic and intuitive visual programming features. Normally, Grasshopper is used to create parametric algorithms in Rhino. However, the Tekla development team has produced a package known as Grasshopper-Tekla Live Link, which enables real-time communication between the Tekla Structures application and Grasshopper. This feature enables users to utilize the visual programming capabilities of Grasshopper to access the functionality of Tekla Structures.

Figure 3-2 displays schematically the Grasshopper script containing the optimization tool. Each "box" within the script, formally known as a component, corresponds to a function that performs a specific action given specific inputs. The component may generate one or more outputs that can subsequently serve as inputs for other components, allowing the user to create algorithms ranging from simple to complex. One major advantage of Grasshopper is the ability to create custom components through C# programming, which expands upon the default library of components. This means that the user can design components to perform virtually any desired function. Therefore, besides the components offered by the Grasshopper-Tekla Live Link, the optimization tool also leverages additional components developed using C# programming and Tekla’s API.

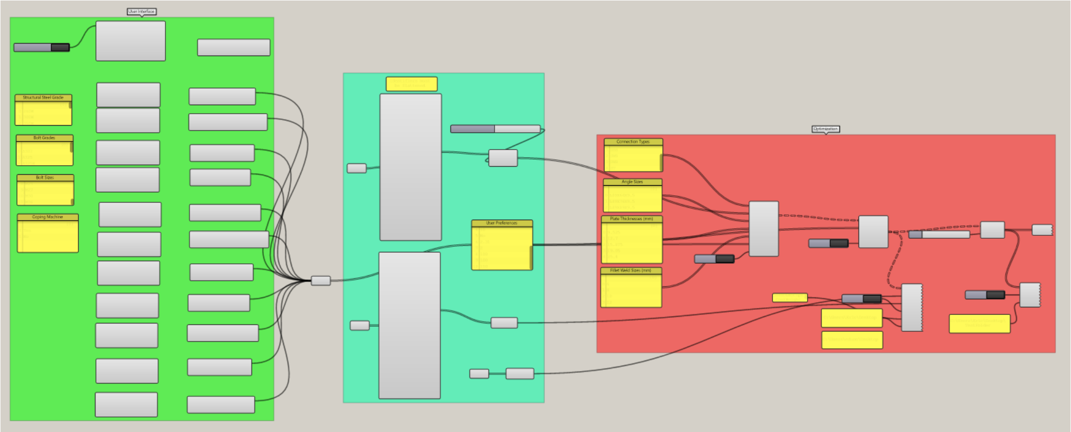
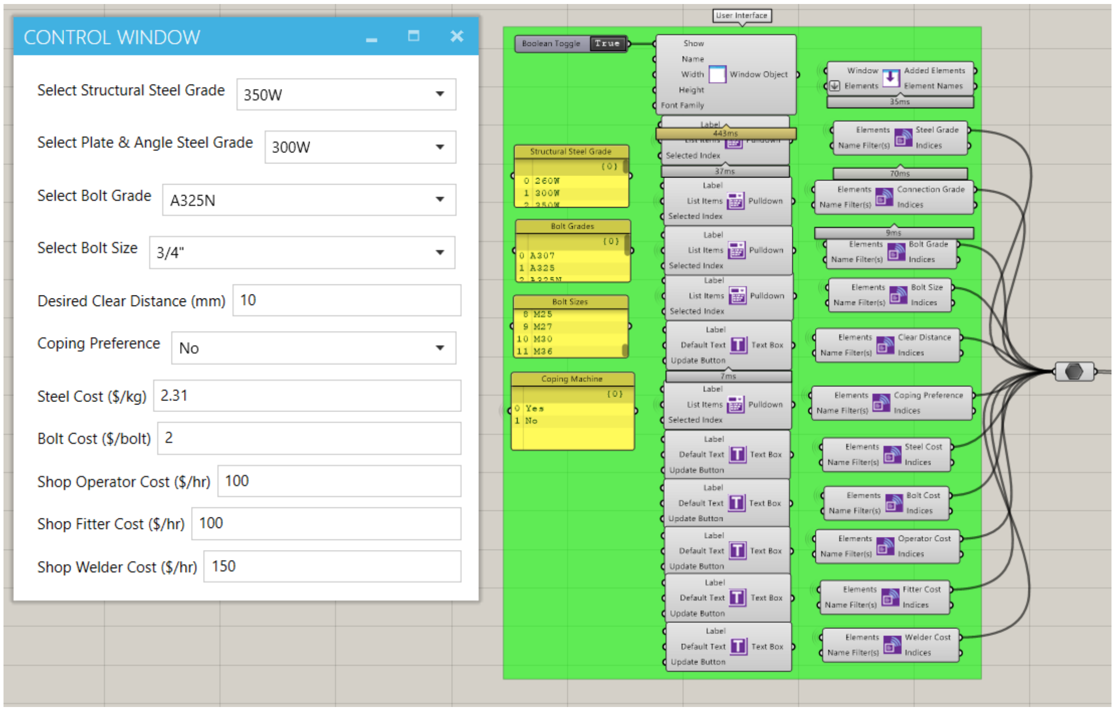


Figure 3-2: Optimization tool in Grasshopper

A control window where the steel fabricator can specify their preferences along with their unit rates is shown in Figure 3-3. The window prompts the user to select various parameters such as the structural steel grade, steel grade of the connection elements, bolt grade, bolt size, coping preference, as well as the cost of steel, bolts, and shop labour. The parameters selected by the user in the control window are utilized in the subsequent stages of design and optimization of the steel connections. Furthermore, these parameters help to generate a reliable cost estimate for fabricating the structural steel framing and its connections.



CONTROL WINDOW

Select Structural Steel Grade: 350W

Select Plate & Angle Steel Grade: 300W

Select Bolt Grade: A325N

Select Bolt Size: 3/4"

Desired Clear Distance (mm): 10

Coping Preference: No

Steel Cost ($/kg): 2.31

Bolt Cost ($/bolt): 2

Shop Operator Cost ($/hr): 100

Shop Fitter Cost ($/hr): 100

Shop Welder Cost ($/hr): 150

Figure 3-3: Control window script in Grasshopper

Figure 3-4 shows the portion of the Grasshopper script that extracts information from the Tekla model. After the user selects all the relevant members, the components extract the joints that require connections along with their available connection space and associated design loads. Typically, structural drawings provide a general statement for the required design force of connections, with specific design forces only specified for atypical cases. Therefore, to obtain the design load for each individual member, three approaches can be utilized: first, calculating the design load as the force required to reach 50% or 60% of the plastic moment capacity of the supported member (assuming the member is braced); second, calculating the design load as the force required to reach 50% or 60% of the shear capacity of the supported member adhering the design specifications of CSA S16-19 (CSA, 2019); or third, manually specifying the design loads by the user. Moreover, the data extracted from the Tekla model is employed to compute the total volume and weight of steel within the project, which is then utilized in both the optimization process and cost estimation discussed in Chapter 4.

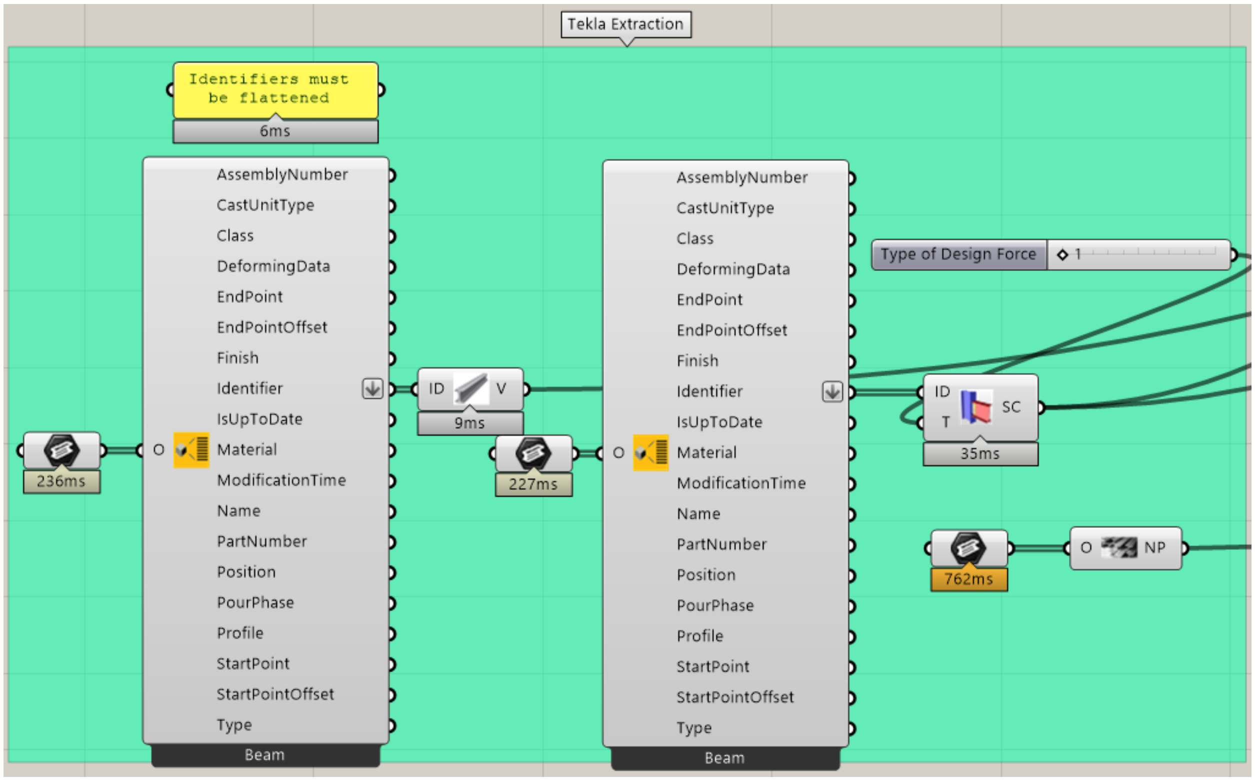


Figure 3-4: Tekla extraction in Grasshopper script

The optimization algorithm is represented in the Grasshopper script by the portion illustrated in Figure 3-5. The yellow boxes located on the left-hand side correspond to additional user preferences that can be readily customized by the steel fabricator. These preferences include options such as the type of connections, angle sizes, plate thicknesses, and fillet weld sizes, all of which can be specified according to the user's design and optimization requirements. For example, if the fabricator possessed a surplus of a particular angle size and plate thicknesses in their fabrication shop, they could specify only those options, and the optimization process would exclusively consider those choices. The remainder of the components correspond to the optimization algorithms and the post-processing procedures executed upon completion of the optimization process. Following the optimization process, the resulting solutions are exported to a spreadsheet, enabling the user to examine the various options and select the most appropriate choice that aligns with their specific needs. Once the design choice has been made, the corresponding connections are automatically detailed back into the Tekla model, and the relevant design calculations for each connection are exported into text files that can be examined by the connection designer.

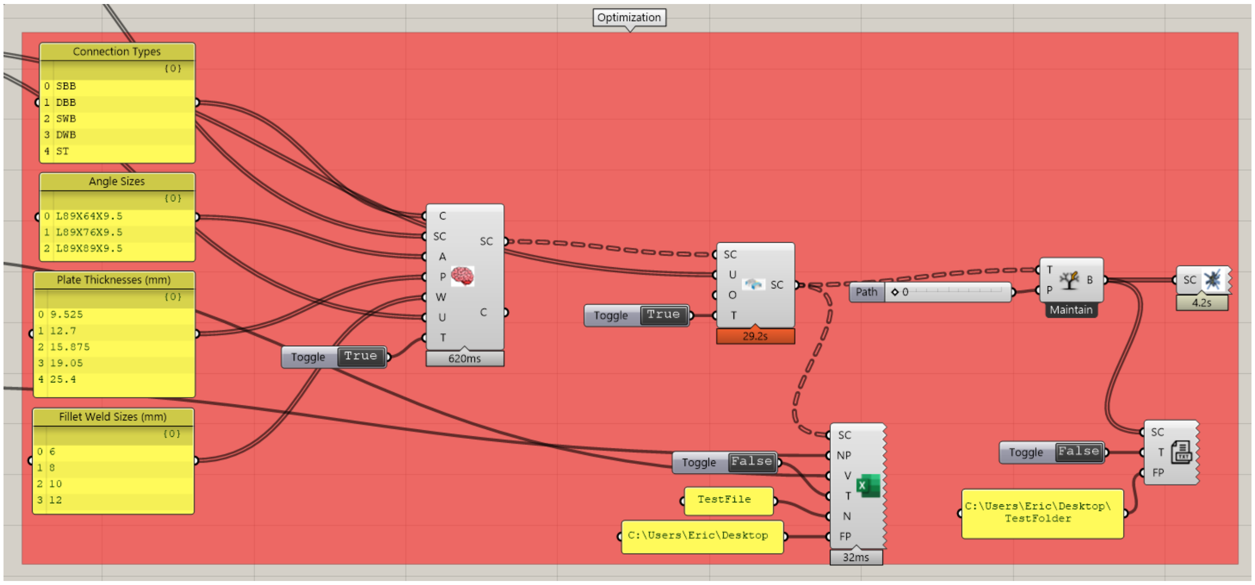


Figure 3-5: Optimization algorithm in Grasshopper script

3.4 Connection Design Modules

In this research project, the connections are limited to shear connections, which constitute the majority of connections in typical buildings, to ensure that the project scope remained feasible. Nonetheless, the optimization tool's underlying framework enables future researchers to incorporate additional connections, such as bracing, moment, axial, and other connection types, into the tool in a straightforward manner. Figure 3-6 illustrates the specific shear connections selected for the optimization tool, which include all bolted double– and single–angle connections, welded–bolted double– and single–angle connections, and shear tabs. The shear connections evaluated in this project were chosen based on an industry survey conducted by the Steel Centre at the University of Alberta (Oosterhof, 2014). The survey was sent to local steel fabricators in Edmonton, Canada, where connection designers ranked common steel shear connections on a scale of one to five, with one corresponding to "rarely used" and five corresponding to "heavily used". The results of the survey were averaged, and the top five connections were incorporated into the optimization tool. A connection design module was developed in C# for each of the five types of connections. These modules evaluate the capacity of the connections based on their parameters in accordance with the Canadian steel design standards CSA S16-19 (CSA, 2019), and American specification for structural steel buildings AISC 360-16 (AISC, 2016). The modules also consider various geometric constraints associated with each connection, such as bolt clashing, sufficient connection space, erection constraints, and code-required detailing requirements. The ultimate limit states that are considered for each connection type are tabulated in Table 3-1.

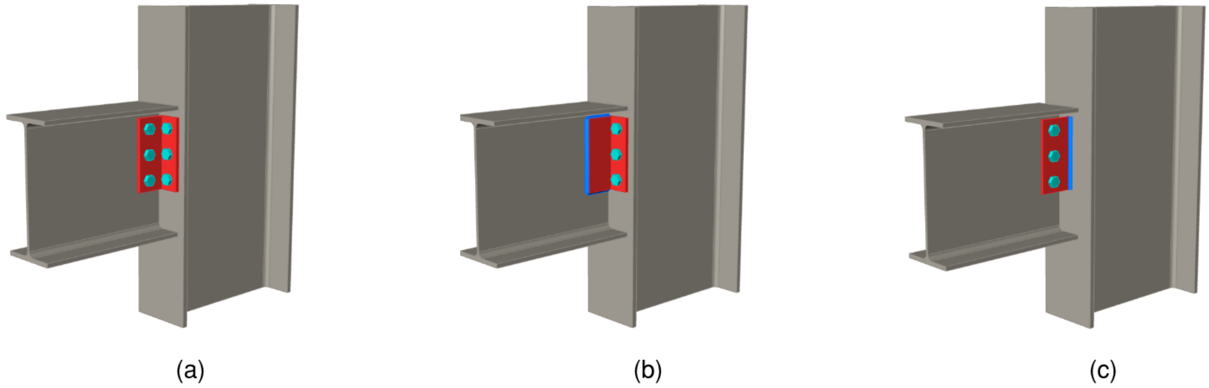


Figure 3-6: (a) Bolted angle connection, (b) welded–bolted angle connection, (c) shear tab connection

Every connection type is assigned its own dedicated design module. While some connection types may have similar limit states, they often have distinct geometric and erection constraints. Therefore, it is more practical to separate them into their own distinct modules. As a result, it becomes possible to incorporate additional connection design modules without impacting any of the existing ones. By developing the design modules in the C# programming language instead of spreadsheets, which is a common tool for connection designers, we gain the advantage of faster evaluation of limit states. This is especially beneficial for evaluating eccentric bolt and weld groups, which require an iterative approach known as the instantaneous center (IC) of rotation method (Kulak et al., 1987; Lesik and Kennedy, 1990). Consider the following scenario: depending on the complexity of the spreadsheet, integrating VBA (Visual Basic for Applications) code for the IC method may require a few seconds to execute due to the need for every cell to update between iterations. When coupled with the requirement to run the IC method hundreds of times to optimize a specific connection, it could take minutes to even design one optimal connection. In contrast, C# executes the IC method within a few milliseconds. However, it worth noting that coding these ultimate limit state calculations can be considerably more laborious and time-consuming in comparison to implementing them in a spreadsheet environment. Nevertheless, the use of C# is heavily integrated throughout the optimization tool, eliminating the need to also incorporate design spreadsheets into the optimization process.

Table 3-1: Ultimate limit states considered for shear connections

Connection Type	Limit States Considered
Single angle bolted–bolted (SBB) Double angle bolted–bolted (DBB)	• Bolt shear (S16-19 Cl. 13.12.1.2c) • Bolt bearing (S16-19 Cl. 13.12.1.2a) • Angle shear yielding (AISC 360-16 Eqn. J4-3) • Angle shear rupture (S16-19 Cl. 13.11) • Angle block shear (S16-19 Cl. 13.11) • Angle flexural yielding (AISC 360-16 Eqn. J4-5) • Angle flexural rupture (AISC 360-16 Eqn. J4-4)
Single angle welded–bolted (SWB) Double angle welded–bolted (DWB)	• Bolt shear (S16-19 Cl. 13.12.1.2c) • Bolt bearing (S16-19 Cl. 13.12.1.2a) • Angle shear yielding (AISC 360-16 Eqn. J4-3) • Angle shear rupture (S16-19 Cl. 13.11) • Angle block shear (S16-19 Cl. 13.11) • Angle flexural yielding (AISC 360-16 Eqn. J4-5) • Angle flexural rupture (AISC 360-16 Eqn. J4-4) • Weld strength (Instantaneous centre of rotation method (Lesik and Kennedy, 1990))
Shear tab (ST)	• Bolt shear (S16-19 Cl. 13.12.1.2c) • Bolt bearing (S16-19 Cl. 13.12.1.2a) • Maximum plate thickness (AISC SCM 15th Ed. Eqn. 10-3) • Plate shear yielding (AISC 360-16 Eqn. J4-3) • Plate shear rupture (S16-19 Cl. 13.11) • Plate block shear (S16-19 Cl. 13.11)

	• Plate flexural yielding (AISC SCM 15th Ed. Section F11) • Plate flexural rupture (AISC SCM 15th Ed. Section F11) • Plate interaction (AISC SCM 15th Ed. Eqn. 10-5) • Weld strength (S16-19 Cl. 13.13.2.2)
All connections (coped)	• Shear resistance of supported member (S16-19 Cl. 13.4.1.1) • Shear rupture of supported member (AISC 360-16 Eqn. J4-4) • Block shear of supported member (S16-19 Cl. 13.11) • Bending of coped section (AISC 15th Ed. Part 9 and AISC SCM 15th Ed. Section F11)

3.5 Cost Estimation

Cost estimation involves the task of anticipating the financial resources needed to accomplish a designated scope of work. In the context of fabricating steel structures, cost estimation becomes an exceedingly challenging undertaking. Nevertheless, achieving an accurate cost estimate throughout the design phase serves to minimize the likelihood of budget overruns, while simultaneously enabling fabricators to efficiently allocate resources across their various projects. In conventional practice, designers traditionally estimate the total installed cost of a steel frame structure by considering the weight of the steel during the early design stages (Haapio, 2012). Skitmore and Ashworth (1983) describe this approach as a rough estimate, as it relies on only a single variable. They highlight the need for additional variables to enhance the accuracy of the estimate, as the total installed cost of a steel frame is not a linear function of its weight. The total installed cost of a steel structure is primarily influenced by three key factors: the material, fabrication, and erection processes.

Material costs encompass a range of components, including structural shapes, plates, bolt products, weld products, painting products, steel joints, steel deck, and any other necessary materials incorporated into the project. Among these, the weight of the structural shapes most significantly influences the material cost.

Fabrication costs involve the detailing and labour process of preparing and assembling shop components such as structural shapes, plates, bolts, welds, and other materials. It accounts for all tasks associated with ensuring the components are ready for subsequent erection on-site, such as shop welding, fitting, and plate cutting, among others.

Erection costs encompass the labour required for unloading, lifting, placing, and connecting the various components of the structural steel frame (Carter and Schlafly, 2008). In summary, it can be represented by the field time required to assemble the steel structure. In addition to the aforementioned categories, cost estimation for a steel structure encompasses various other expenses that are not explicitly addressed. These include additional costs related to risk management, the need for contingency planning, fireproofing, and engineering design (Carter and Schlafly, 2008). A detailed breakdown of these categories is presented in Figure 3-7.

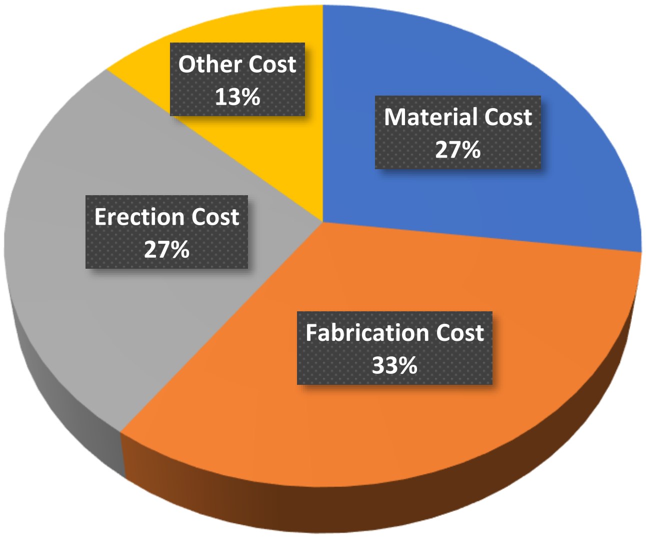


Figure 3-7: Typical distribution of total installed cost of a typical building steel frame (Carter and Schlafly, 2008)

For this research project, explicit consideration was given to material and fabrication costs, while the inclusion of erection costs was omitted primarily due to the lack of available data for erection-related tasks. Nonetheless, certain erection-specific issues were addressed using alternative approaches. For shear connections, it is common practice to shop weld the connection elements and perform on-site bolting when feasible, thus minimizing the influence on erection expenses as reflected in the optimization tool. However, certain erection considerations may arise, such as the presence of angle connections on both sides of the web of a W-shape column. These specific considerations are addressed within the optimization process outlined in Section 4.4.

The cost function utilized in the optimization tool is a result of a collaborative effort with a local steel fabricator. It was derived through a two-week study conducted within their fabrication shop. Valuable insights were gathered by engaging in discussions with operators, fitters, and welders regarding the complexities of fabricating shear connections for steel structures. Timings of various tasks, outlined in Table 3-2, were recorded multiple times and averaged across different operators, fitters, and welders to ensure a comprehensive and representative evaluation. Following this, discussions were made with the estimators to compare the collected timings with their existing estimation practices. This collaboration contributed to enhancing the accuracy of the cost function. Notably, certain tasks displayed significant variability and were consolidated into broader categories, such as plate handling and section handling, to accommodate their diverse nature. This grouping approach enables a more comprehensive representation within the cost function, thereby facilitating a more precise estimation of associated expenses.

To estimate the material and fabrication cost of the steel structure and its shear connections, the sections are extracted from the Tekla model. The lengths and profiles of these sections are determined, and the total weight is calculated. This weight is then multiplied by the user-defined rate for the material cost of the structural steel.

For each individual shear connection, the optimization tool analyzes the parameters to determine the total time required to fabricate the connection. The rates outlined in Table 3-2 are used for this analysis. Additionally, the number of bolts for each connection is also consolidated. The resulting values are multiplied by the unit rates of the corresponding shop labour, generating a final cost estimate.

In the case of manual coping, it is considered only if the fabricator specifies the absence of a beam line capable of automatically coping its members. In such situations, manual coping performed by the fitter is necessary, leading to increased time and costs. Alternatively, if automatic coping is available, it is included within the section handling process.

As mentioned earlier, the current cost function concentrates exclusively on material and fabrication costs, omitting erection expenses. Nevertheless, its main objective is to provide a dependable estimate for the fabrication expenditures associated with a steel building. Notably, the labour and material rates incorporated in the cost function are adjustable, enabling users to customize them to suit their specific needs. The C# framework also facilitates the modification of rates outlined in Table 3-2, enhancing the flexibility of the cost function. The significance of this cost function becomes evident during the optimization process, outlined in Chapter 4, where the primary aim is to minimize costs associated with fabricating a steel structure.

Table 3-2: Price rates for the cost function for various tasks

Material Element	**Rate**	**Description**
Hot-rolled sections	User defined ($/kg)	Price rate for hot-rolled sections.
Bolts	User defined ($/bolt)	Price rate for individual bolts.
Fabrication Element		
Shop operator	User defined ($/hr)	Price rate for labour associated with the operator.
Shop fitter	User defined ($/hr)	Price rate for labour associated with the fitter.
Shop welder	User defined ($/hr)	Price rate for labour associated with the welder.
Angle handling	15 min/stock angle	Minutes of shop time to load, unload, and move stock angles around the shop.
Angle hole punching	0.1 min/hole	Minutes to punch a hole in an angle.
Angle shearing	0.17 min/shear	Minutes to shear an angle from the stock angle.
Fit and QC angle	4 min/angle	Minutes to fit the angle to the appropriate member and provide an inspection for quality control.
Plate handling	7.2 min/plate	Minutes of shop time to move the plates around the shop and layout operations.
Plate cutting	0.0033 min/cm^2	Minutes to cut the perimeter of the plate multiplied by the plate's thickness
Plate hole drilling	0.033 min/cm^2	Minutes to drill a hole in a plate.
Section handling	45 min/section	Minutes to load, unload, and move steel sections around the shop. Coping is also included if the adequate machinery is available.
Section hole drilling	1 min/hole	Minutes to drill a hole in a steel section.
Manual coping	45 min/section	Minutes to manually cope a member.
Shop fillet welds	0.262 min/cm	Minutes to weld fillet connections including set-up time. Rate is based on the length of the weld. Larger fillet welds that require multiple passes are accounted for.
Shop bolt install	1 min/bolt	Minutes to install a bolt in the shop.

3.6 Metaheuristics

As discussed in Chapter 2, metaheuristic algorithms use stochastic search strategies to efficiently explore the search space to find near-optimal or global optimum solutions. They are an effective way to find a range of optimal solutions for multi-objective optimization problems. Existing evolutionary solvers such as Galapagos (Rutten, 2019) or Wallacei (Makki et al., 2022) are among the most commonly used methods of employing metaheuristics in Grasshopper. However, these solvers have several drawbacks, including their limitations to only one algorithm, difficulty optimizing general variable types, and their inability to export results outside of Rhino. These limitations significantly hinder the ability to implement an appropriate optimization procedure for this research project, as explained further in Chapter 4. Therefore, this project uses jMetal.NET, an object-oriented framework for multi-objective optimization, instead.

jMetal, which stands for Metaheuristic Algorithms in Java, is a well-established research project that provides a suite of classical and state-of-the-art metaheuristic algorithms. The .NET version of jMetal allows for the use of C# to implement the framework, making it a versatile tool for solving real-world problems and conducting experiments in a Windows-based environment. The object-oriented framework of jMetal further simplifies the implementation of the algorithms, enabling researchers to explore different optimization techniques with ease (Durillo et al., 2010; Durillo and Nebro, 2011). More information regarding jMetal and its toolbox of available metaheuristic algorithms can be found here: https://jmetal.sourceforge.net/.

This research project investigates three algorithms for optimization: Non-dominated Sorting Genetic Algorithm II (NSGA-II), Strength Pareto Evolutionary Algorithm 2 (SPEA2), and a specific Multi-objective Particle Swarm Optimization (MOPSO) known as the Speed-constrained Multi-objective Particle Swarm Optimization (SMPSO). The selection of these three algorithms was informed by a survey conducted by Zavala et al. (2014), which analyzed the application of multi-objective metaheuristics in structural optimization across 50 publications published between 1992 and 2012. According to the researchers, the NSGA-II was the most commonly used technique, followed by SPEA2 and MOPSO. While this study focused solely on these three algorithms, it is worth noting that the optimization tool can be extended to include other metaheuristics in future research. Below, a summary of the three algorithms is provided. For in-depth information on NSGA-II, refer to Deb et al. (2002); for SPEA2, consult Zitzler et al. (2001); and for further details on SMPSO, refer to Nebro et al. (2009).

Over the last two decades, a popular sorting-based multi-objective evolutionary algorithm called Non-dominated Sorting Genetic Algorithm II (NSGA-II) has seen lots of success in the field of engineering (Madeira et al., 2005; Sharma et al., 2011; Stankovic et al., 2011; Stanković et al., 2012; Tang et al., 2011; Winslow et al., 2010). NSGA-II solves MOOPs through Pareto dominance and a sorting procedure known as non-dominated sorting. A solution is considered to dominate another solution if all its fitness values, determined from the user-defined objective criteria, are lower than or equal to another solution's fitness values. An example of non-dominated sorting for a double objective problem is illustrated in Figure 3-8. During non-dominated sorting, solutions are compared against each other and are assigned ranks based on dominance. Lower ranks refer to better fitness values and are associated with solutions that have higher dominance. In MOOPs, a solution is said to be a Pareto optimal solution if it is not dominated by any other solution of the MOOP. There exists more than one Pareto optimal solution for an MOOP due to the conflicting nature of objectives. At each generation of NSGA-II, non-dominated sorting is first employed to select solutions with lower ranks from the parent and offspring populations, and crowding distance is used as the secondary metric to distinguish solutions in the same rank by favouring solutions with a large crowding distance (Deb et al., 2002; Tian et al., 2017).

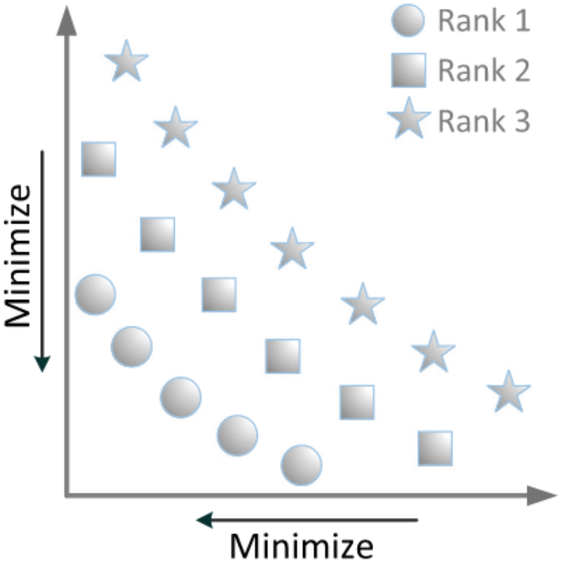


Figure 3-8: Illustration of non-dominated sorting with the population divided into three ranks (Tian et al., 2017)

Another elitist multi-objective evolutionary algorithm that has seen success in the engineering field is the SPEA2 (Greiner et al., 2011; Kicinger et al., 2007; Kunakote and Bureerat, 2011; Noilublao and Bureerat, 2009). Originally developed by Zitzler et al. (2001), SPEA2 shares similarities with the NSGA-II as it adopts an evolutionary Pareto-based approach to generate a set of optimal solutions for MOOPs. The algorithm begins by creating an initial population where each individual is evaluated and has a fitness value that is the sum of its strength raw fitness plus a density estimation. Non-dominated individuals are then copied to an archive or external set. If the archive exceeds its capacity, a truncation operator using *k*-th nearest neighbour (Silverman, 1986) is implemented, ensuring that individuals are removed in a manner that preserves the distinctive characteristics of the non-dominated front, as shown in Figure 3-9. Subsequently, a binary tournament selection mechanism is employed to populate the mating pool, followed by the application of recombination and mutation operators to generate a new parent population for the subsequent iteration (Zitzler et al., 2001). The researchers noted that the SPEA2 seems to have advantages over the NSGA-II in higher dimensional objective spaces.

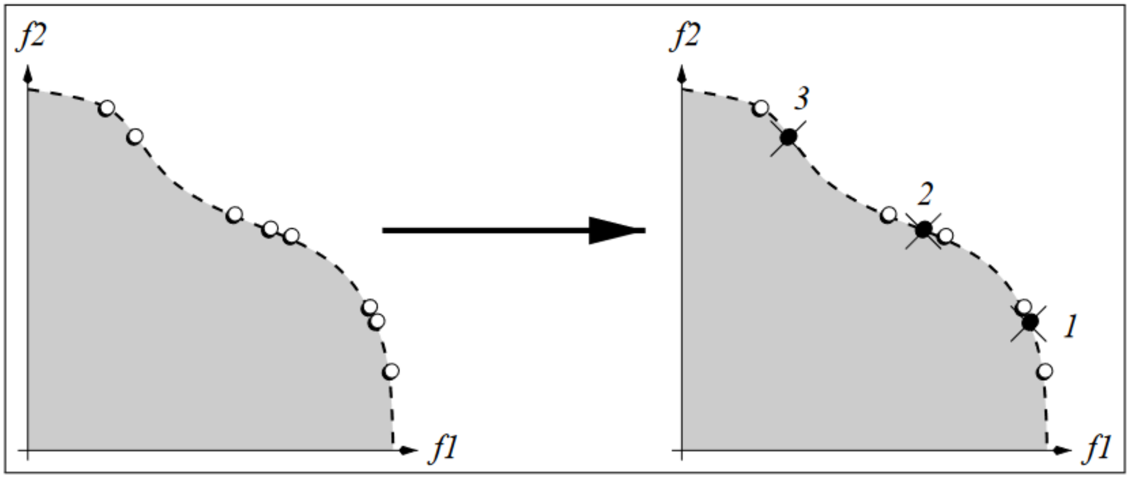


Figure 3-9: Illustration of the archive truncation method used in SPEA2. On the left, a non-dominated front set is shown. On the right, solutions are removed ordered by the truncate operator (assuming the archive size is 5)

Particle Swarm Optimization (PSO) is a metaheuristic inspired by the collective behaviour of birds flocking or fish schooling (Kennedy and Eberhart, 1995). By simulating the social interactions and adaptive movements observed in these natural phenomena, PSO creates a population of particles that explore the search space in a cooperative manner. Each particle adjusts its position based on its own experience as well as the knowledge shared within the group known as the swarm. In multi-objective optimization, each particle represents one potential solution and all the particles in the swarm can search for different parts of the Pareto front simultaneously (Durillo et al., 2009; Wang and Qian, 2008; Kennedy and Eberhart, 1995). One drawback of MOPSO is the potential for excessively high particle velocities for certain optimization problems leading to unsatisfactory solutions. Durillo et al. (2009) proposed using a velocity constriction procedure to solve the aforementioned problem, which forms the basis for the SMPSO algorithm that is used in this research project (Durillo et al., 2009).

As previously mentioned, a variety of metaheuristics are available for potential utilization in this research project. The performance of different algorithms varies depending on the specific optimization problem at hand. In the context of minimizing the fabrication cost of a steel structure, the NSGA-II algorithm emerged as the most effective among the three algorithms discussed in this chapter. Further elaboration on this topic will be provided in Chapters 4 and 5.

4 APPLICATION OF OPTIMIZATION TOOL

4.1 Introduction

This chapter delves into the design of each type of connection, as well as the optimization procedure employed by the optimization tool. This approach utilizes a two-tiered optimization strategy as depicted in Figure 4-1, aiming to minimize the combined cost and weight of a steel structure and its connections in a designated building project. In the initial optimization layer, the focus lies on designing and optimizing multiple connections at each necessary joint, taking into account the user's preferred connection type selection. In the secondary optimization layer, metaheuristics are employed to identify the most optimal combination of connections throughout the entire structure, with the overarching goal of minimizing the overall cost and weight.

Figure 4-1 employs distinct colors to represent each stage: the initial parameters provided by the user are represented in blue, followed by the first layer of optimization in yellow, the second layer of optimization in red, and finally, the post-processing stage in green, where the optimization tool displays the results for the user.

In this chapter, the design of each connection is initially explored, followed by an in-depth investigation into the optimization process of each individual connection. Subsequently, the focus shifts to the second layer of optimization, encompassing an overarching analysis of the entire optimization procedure. This is followed by a detailed discussion of the specific parameters associated with the employed metaheuristics.

4.2 Design and Optimization of Connections

The optimization tool incorporates five distinct types of shear connections, encompassing all bolted double- and single-angle connections, welded–bolted double- and single-angle connections, and shear tab connections. Each of these connections is evaluated using the developed connection design modules, adhering to the Canadian steel design standard CSA S16-19 (CSA, 2019) and the American specification for structural steel buildings AISC360-16 (AISC, 2016), as detailed in Section 3.4. As previously mentioned, the optimization tool exclusively focuses on shear connections. Nevertheless, the design modules for connections are developed in C# as classes within an object-oriented framework, facilitating ease of integration of additional connection types, including axial, moment, or bracing connections. This can be achieved by introducing another class to this framework.

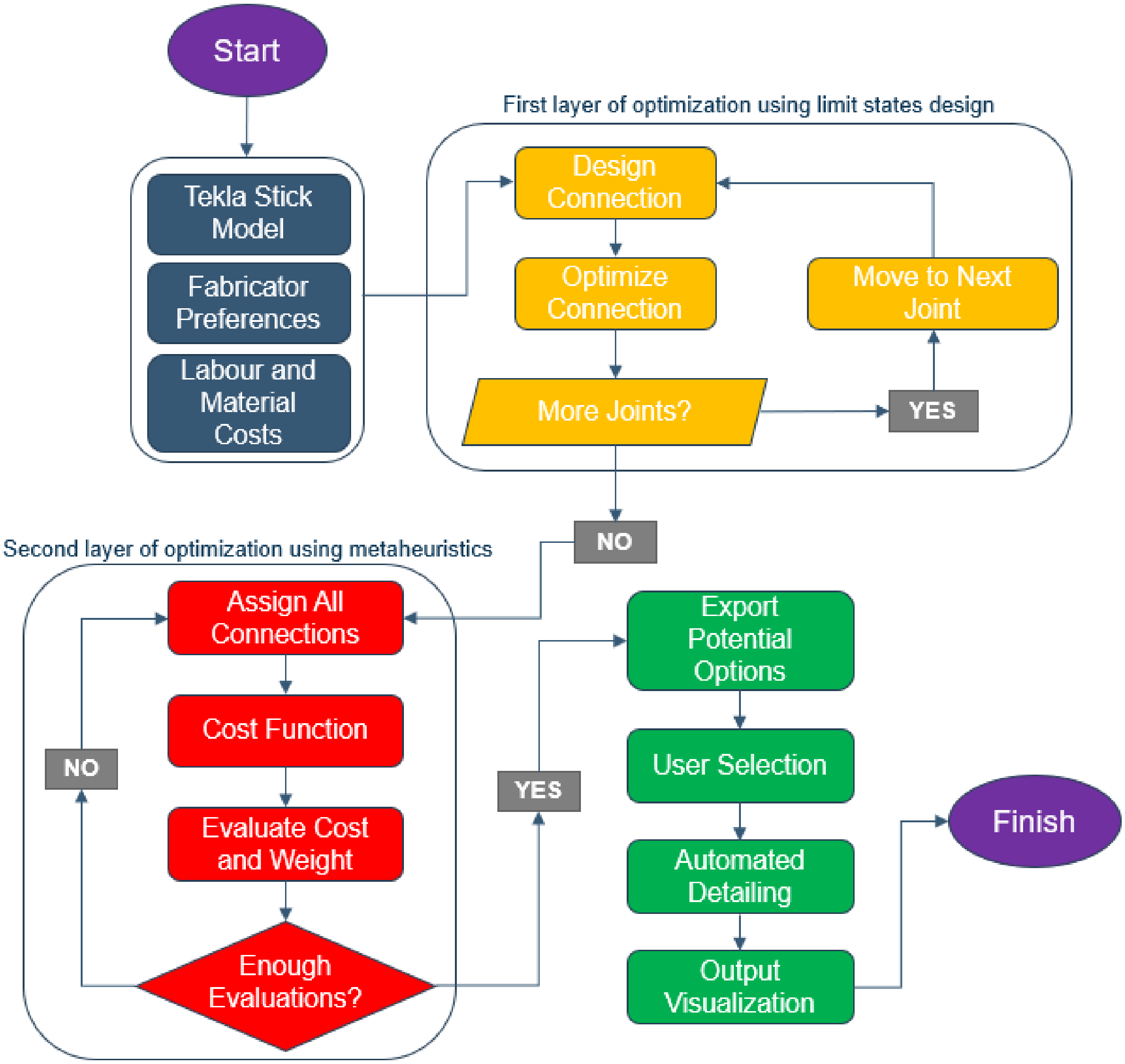


Figure 4-1: Flowchart of optimization procedure utilized by the optimization tool

The joints extracted from Tekla can be categorized into four distinct framing configurations: beam-to-column-flange, beam-to-column-web, beam-to-HSS-column, and beam-to-beam. These four framing configurations are illustrated in Figure 4-2. Additionally, there are other framing configurations that require design considerations beyond shear alone, although these are currently not accounted for in the optimization tool. It is worth emphasizing the significance of extracting this information from the Tekla model, as certain connection types offer distinct advantages based on the framing configuration. Moreover, fabricators often have their preferred connection types, further highlighting the importance of aligning with their preferences.

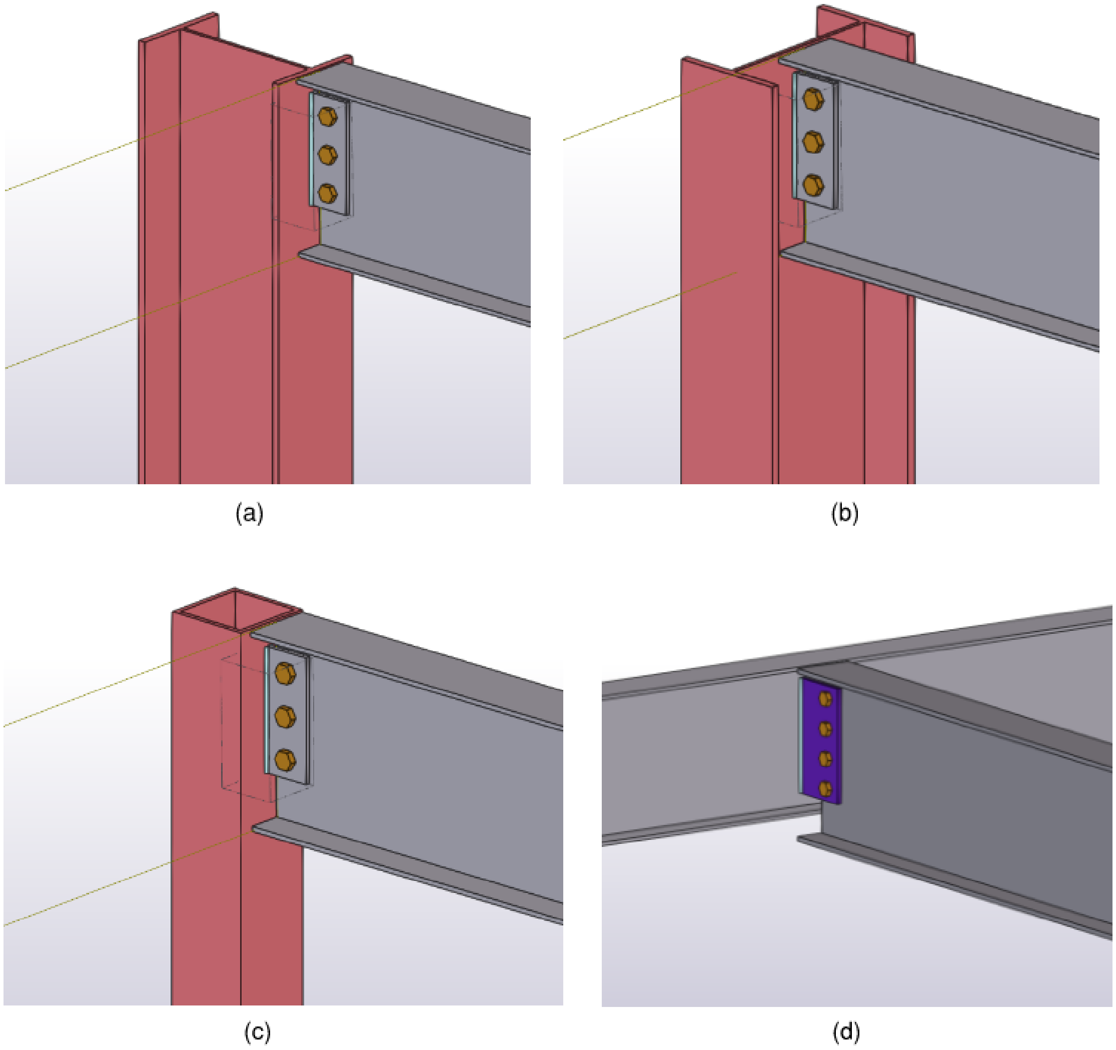


Figure 4-2: Typical framing configurations for shear connections: (a) beam-to-column-flange, (b) beam-to-column-web, (c) beam-to-HSS-column, (d) beam-to-beam

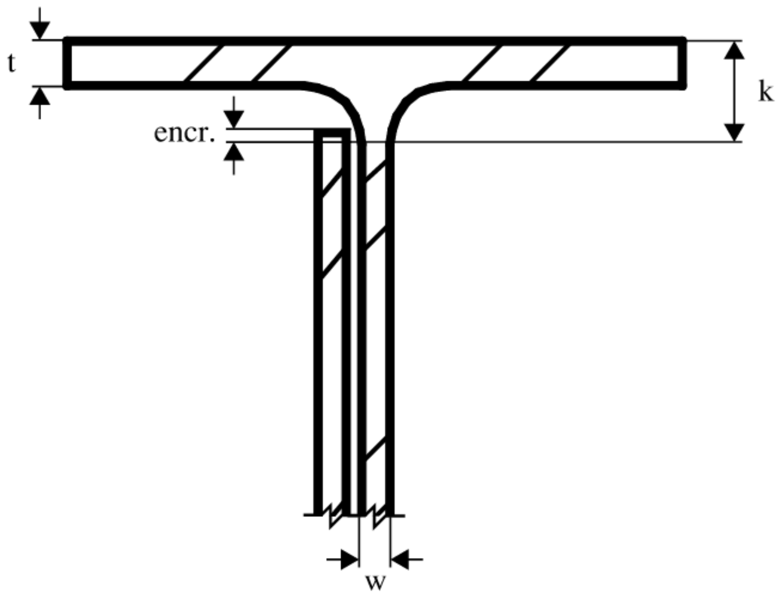
Additional information, including the slope and skew angle of the supported member in relation to the support member, the elevation difference between the support and supported member, as well as the steel profiles of both members, is also extracted from the Tekla model. This crucial information plays a vital role in determining the geometric constraints which are used extensively during the design and optimization process.

As outlined in Section 3.3, there are three approaches available in the tool for determining the factored design load for each individual connection. These approaches are:

1. Calculating the design load as the reaction force consistent with achieving 50% of the factored plastic moment capacity of the supported member, in accordance with CSA S16-19, assuming the member is laterally braced. 60% of the factored plastic moment capacity of the supported member may also be selected.
2. Calculating the design load as 50% of the shear capacity of the supported member, in accordance with CSA S16-19. 60% of the shear capacity of the supported member may also be selected.
3. Manually specifying the design loads directly within the Tekla model.

Additionally, AISC (2019) recommends that the minimum depth of simple shear connections exceed one-half of the T-dimension of the supported member to achieve connection stability. The T-dimension refers to the distance between the web toes of the fillet of the top and bottom flanges. In the case of a coped supported member, this criterion is adjusted to be one-half of the remaining web depth of the supported member (AISC, 2019). The connection element may encroach upon the fillet of the supported member slightly, as shown in Table 4-1.

Table 4-1: Allowable fillet encroachment (CISC, 2021)

	Fillet radius k – t (mm)	Encroachment (mm)
	8	3
	9	4
	10	4
	12	4
	14	5
	16	5
	18	5
	20	6
	22	6
	24	6
	26	7

The specific ultimate limit states considered for each connection type are outlined in Table 3-1. While the details of these limit states are not discussed here, readers seeking more information can refer to CSA S16-19 and AISC360-22 (AISC, 2022) for comprehensive explanations. The design and optimization process of the connections are implemented using C#. Notably, the optimization of connections is accomplished through a script rather than employing metaheuristics. This decision was primarily motivated by the significant difference in optimization speed. Given the relatively large search space for a seemingly simple design problem, metaheuristics like the NSGA-II required a considerable amount of time to properly optimize a connection. In contrast, the script enabled optimization of an individual connection within one millisecond, whereas metaheuristics required approximately five seconds per connection, about five thousand times faster. Additionally, it should be emphasized that during a single-objective optimization of the connection, aiming to maximize the demand-to-capacity ratio while adhering to the constraint of not exceeding 1.0, the NSGA-II algorithm consistently produced identical results to those achieved by the optimization script for all five connection types.

The subsequent sections provide a comprehensive explanation of the design and optimization process for each specific connection type.

4.2.1 Bolted–Bolted Angle Connections

The design and optimization of a bolted–bolted angle connection incorporates several key parameters:

1. Angle designation: This parameter refers to the size of the angle that will be used. It is selected from a list of angles specified by the user before the optimization process.
2. Number of bolt rows: This parameter determines the number of (horizontal) rows of bolts that will be used in the connection.
3. Number of bolt columns in the short leg: This parameter indicates the number of (vertical) columns of bolts in the short leg of the angle, with a maximum of 2.
4. Number of bolt columns in the long leg: This parameter represents the number of columns of bolts in the long leg of the angle, also with a maximum of 2.
5. Vertical bolt pitch: This parameter denotes the vertical centre-to-centre distance between the bolt rows.

6. End distance: This parameter specifies the distance between the centres of the top and bottom bolts and the top and bottom edges, respectively, of the angle.
7. Orientation of the legs: This parameter determines how the short and long legs of the angle are connected to the supporting and supported members.

These parameters are adjusted during the optimization of the connection such that the demand-to-capacity ratio is maximized, while maintaining a ratio equal to or below 1.0. The end distance is fixed to the minimum allowable distance as prescribed by S16-19. For bolted–bolted single angle connections, eccentricity must be taken into account for the outstanding framing leg at all times. However, eccentricity in the web framing leg only needs to be considered when the gauge exceeds 75 mm or when two columns of bolts are utilized (AISC, 2019). In Figure 4-3, the requirement for eccentricity is indicated by the letter E, while g represents the maximum gauge before eccentricity needs to be taken into consideration for that particular leg. For bolted–bolted double angle connections, there is no need to consider eccentricity for the outstanding framing leg. However, the web framing leg of the double angle connection follows the same criteria as the single angle connection. For this reason, the short leg of the angle is always assumed to be oriented such that it connects to the support member during the optimization procedure. The gauges for the specific angle sizes are taken from the CISC Handbook of Steel Construction (CISC, 2021). An illustration of this type of connection is shown in Figure 4-4.

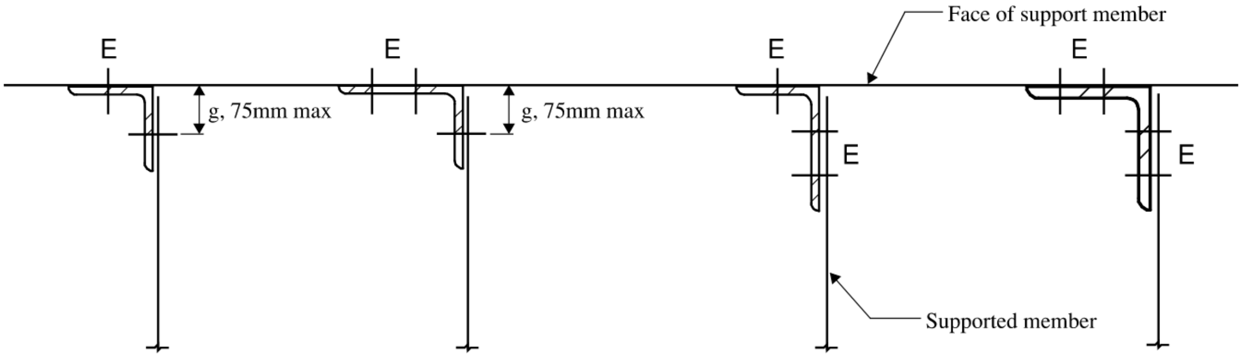


Figure 4-3: Eccentricity in single angle bolted connections

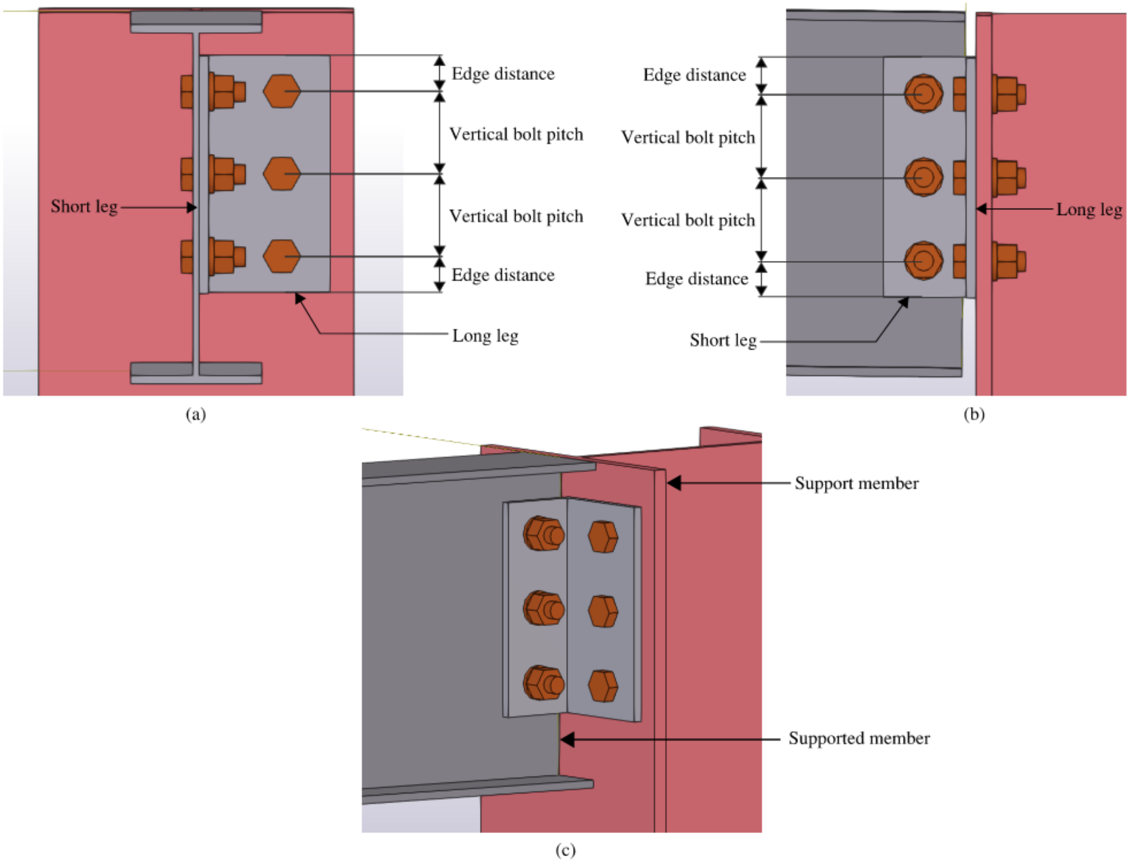


Figure 4-4: Bolted–bolted single angle connection of a L127x89x9.5 angle: (a) front view, (b) side view, (c) isometric view

The connection design module for bolted–bolted angle connections also ensures the adherence to design integrity by verifying various geometric constraints. These constraints, essential for meeting detailing requirements, include:

1. Support spacing: Verifies if the specified angle size and orientation fit within the available space provided by the support member.
2. Bolt spacing: Checks if the bolt pitch adheres to the minimum pitch specified in CSA S16 for the given bolt size.
3. Bolt clashing: Examines whether the bolts would collide, following the detailing requirements outlined in the CISC Handbook of Steel Construction for the given bolt size.
4. Edge distance: Verifies if the minimum edge distance requirement, as specified in CSA S16 for the given bolt size, is met.
5. Skew angle: Determines if the supported member is skewed relative to the support member, as a bolted–bolted angle connection cannot be used in such cases.

6. HSS support member: Determines if the support member is an HSS column since the optimization tool does not currently support thru-bolts, making a bolted–bolted angle connection unsuitable in this particular scenario.
7. Depth of connection: Determines if the depth of the connection satisfies the connection stability requirement.
8. Connection space: Verifies the availability of sufficient space for the given connection detail.

The optimization procedure for bolted–bolted single- and double- angle connections follow a similar approach, with the only distinction lying in the evaluated limit states between these connection types. The procedure can be divided into three steps. First, the design parameters are initialized, and incompatible angle sizes are filtered out. This ensures that only appropriate angle sizes are considered for the optimization process. Next, the minimum number of bolts and smallest angle size that satisfy the strength requirements are determined through an iterative process. Finally, in cases where a coped supported member is involved, the limit states related to it are evaluated. Concurrently, the material usage is optimized by reducing the bolt pitch, while maintaining the necessary strength requirements.

Figure 4-5 illustrates the initial step of the procedure. In this step, the list of angles provided by the user is sorted in ascending order based on leg size and thickness. This sorting approach ensures that the angle sizes are iterated from smallest to largest, facilitating the selection of the optimal angle size in subsequent stages. Figure 4-5 also indicates the initial design parameters. The number of bolt rows is set to a minimum of 2, with the option to add more bolts if necessary. The number of bolt columns is set to 1. The vertical bolt pitch is initially set as the maximum bolt pitch, the lesser of 14 times the thickness of the thinner connected part or 180 mm as per AISC Specification J3.5 (AISC, 2016). This allows for gradual adjustments and fine-tuning of the bolt pitch by decreasing it before considering the addition of more bolts.

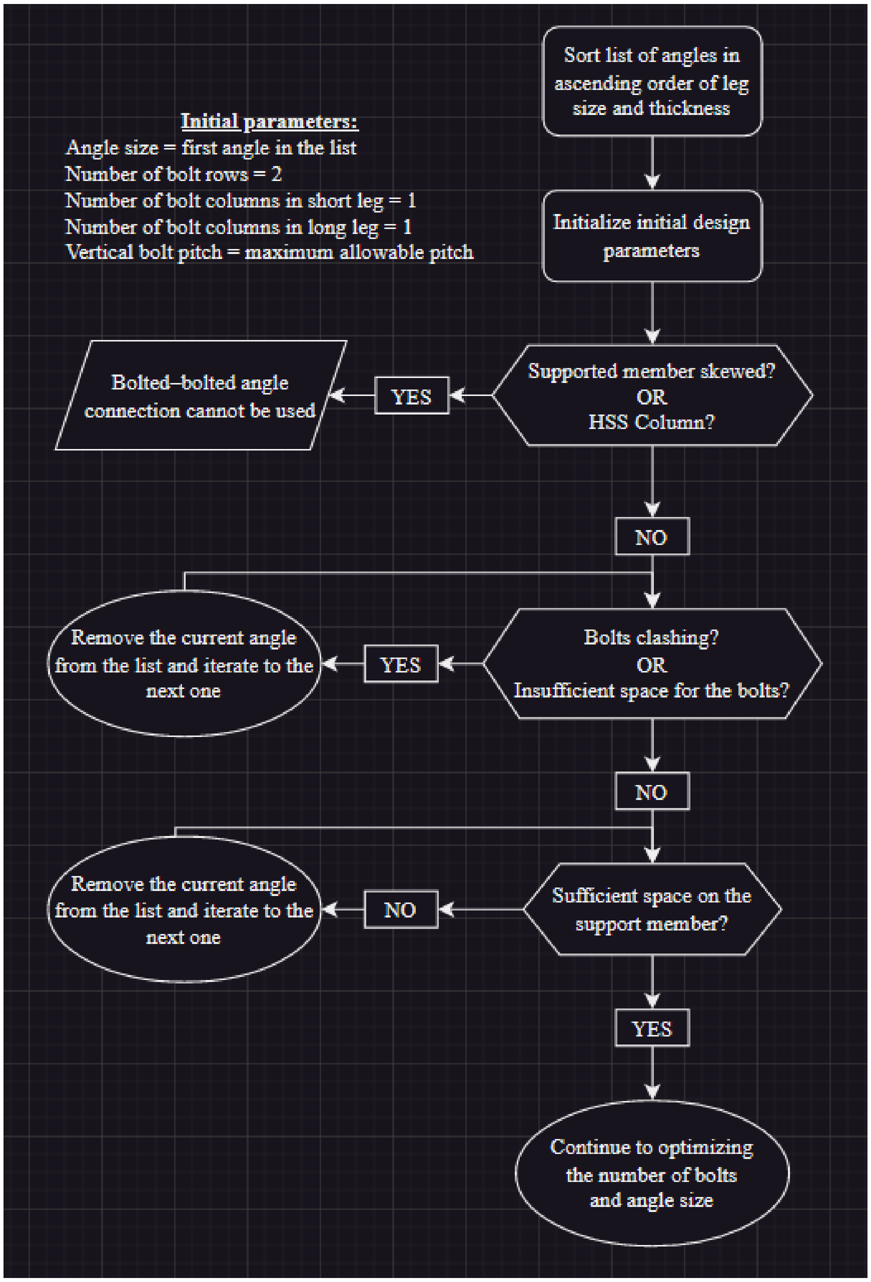

Sort list of angles in ascending order of leg size and thickness
Initial parameters:
Angle size = first angle in the list
Number of bolt rows = 2
Number of bolt columns in short leg = 1
Number of bolt columns in long leg = 1
Vertical bolt pitch = maximum allowable pitch
Initialize initial design parameters
Supported member skewed? OR HSS Column?
YES
Bolted–bolted angle connection cannot be used
NO
Bolts clashing? OR Insufficient space for the bolts?
YES
Remove the current angle from the list and iterate to the next one
NO
Sufficient space on the support member?
NO
Remove the current angle from the list and iterate to the next one
YES
Continue to optimizing the number of bolts and angle size

Figure 4-5: Flowchart of initial step of optimization procedure for bolted–bolted angle connections

Figure 4-6 provides an illustration of the second step of the procedure, which is carried out separately for both the web framing leg and the outstanding framing leg. This section follows an iterative approach to ensure the availability of connection space for adding additional bolts to satisfy the limit states associated with the bolt group. Simultaneously, it iterates through the list of angles to identify the smallest angle size that meets the required strength criteria. It is important to note that certain cases may arise where a satisfactory connection design cannot be achieved, such as when the number of required bolts exceeds the available connection space.

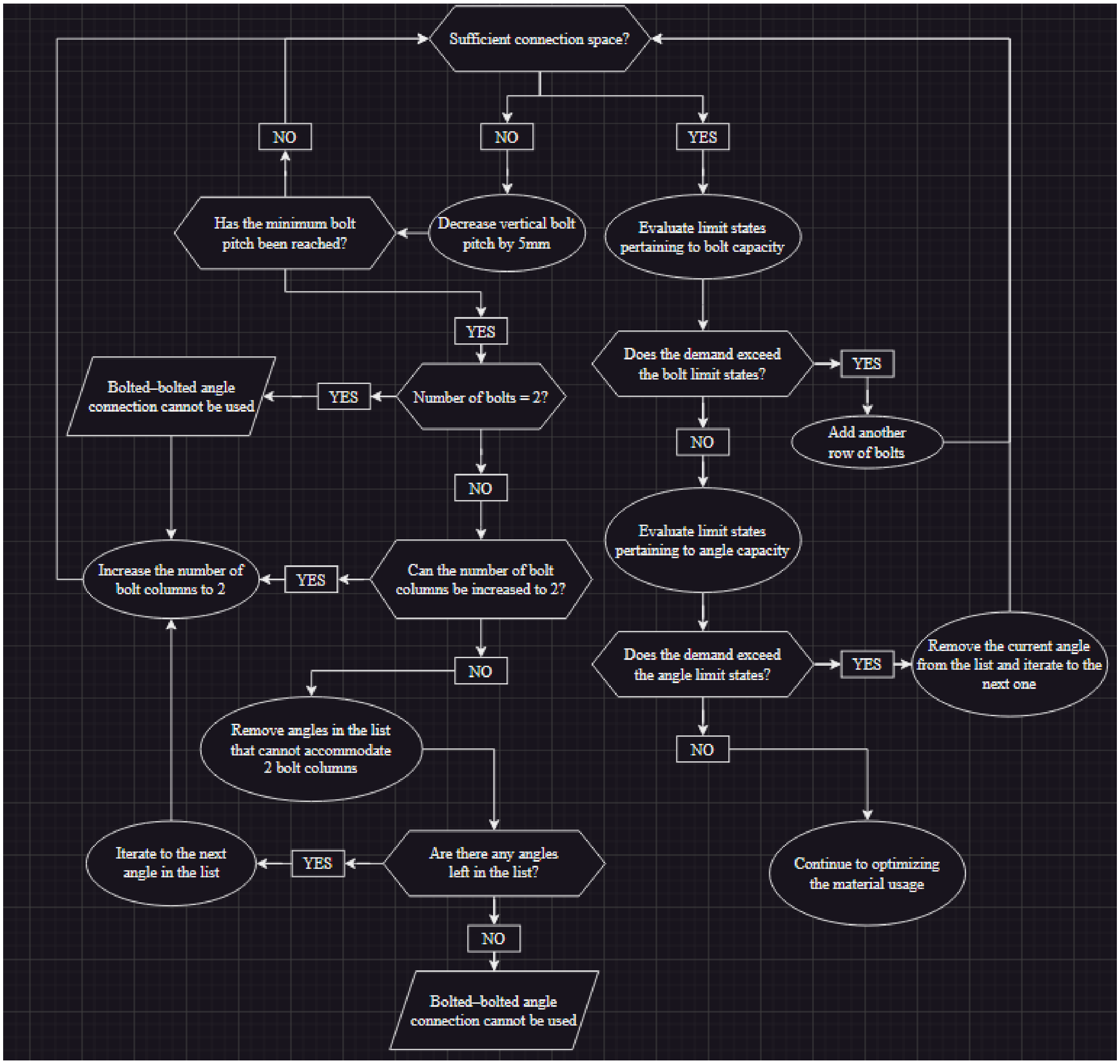

Sufficient connection space?
NO
NO
YES
Has the minimum bolt pitch been reached?
Decrease vertical bolt pitch by 5mm
Evaluate limit states pertaining to bolt capacity
YES
Does the demand exceed the bolt limit states?
YES
Bolted–bolted angle connection cannot be used
YES
Number of bolts = 2?
NO
Add another row of bolts
NO
Evaluate limit states pertaining to angle capacity
Increase the number of bolt columns to 2
YES
Can the number of bolt columns be increased to 2?
Does the demand exceed the angle limit states?
YES
Remove the current angle from the list and iterate to the next one
NO
Remove angles in the list that cannot accommodate 2 bolt columns
NO
Iterate to the next angle in the list
YES
Are there any angles left in the list?
Continue to optimizing the material usage
NO
Bolted–bolted angle connection cannot be used

Figure 4-6: Flowchart of second step of optimization procedure for bolted–bolted angle connections

Figure 4-7 provides an overview of the final step of the optimization procedure. Once the bolt group and angle size have been determined, the optimization script proceeds to evaluate whether the supported member is coped. In the case of a coped member, additional limit states specific to the coping are assessed. It is crucial to note that if the limit states associated with the coped member fail to meet the required strength criteria, the connection type will be deemed unsuitable at this stage. This is because any further adjustments to the bolt group would lead to an unsatisfactory connection according to the bolt limit states. However, if the limit states for the coped member are met or if the supported member is not coped, the bolt pitch is then further reduced, aiming to minimize material usage while still satisfying the required strength criteria. The final connection design results in a bolted–bolted angle connection with satisfied geometric constraints and a demand-to-capacity ratio that is maximized while remaining below 1.0.

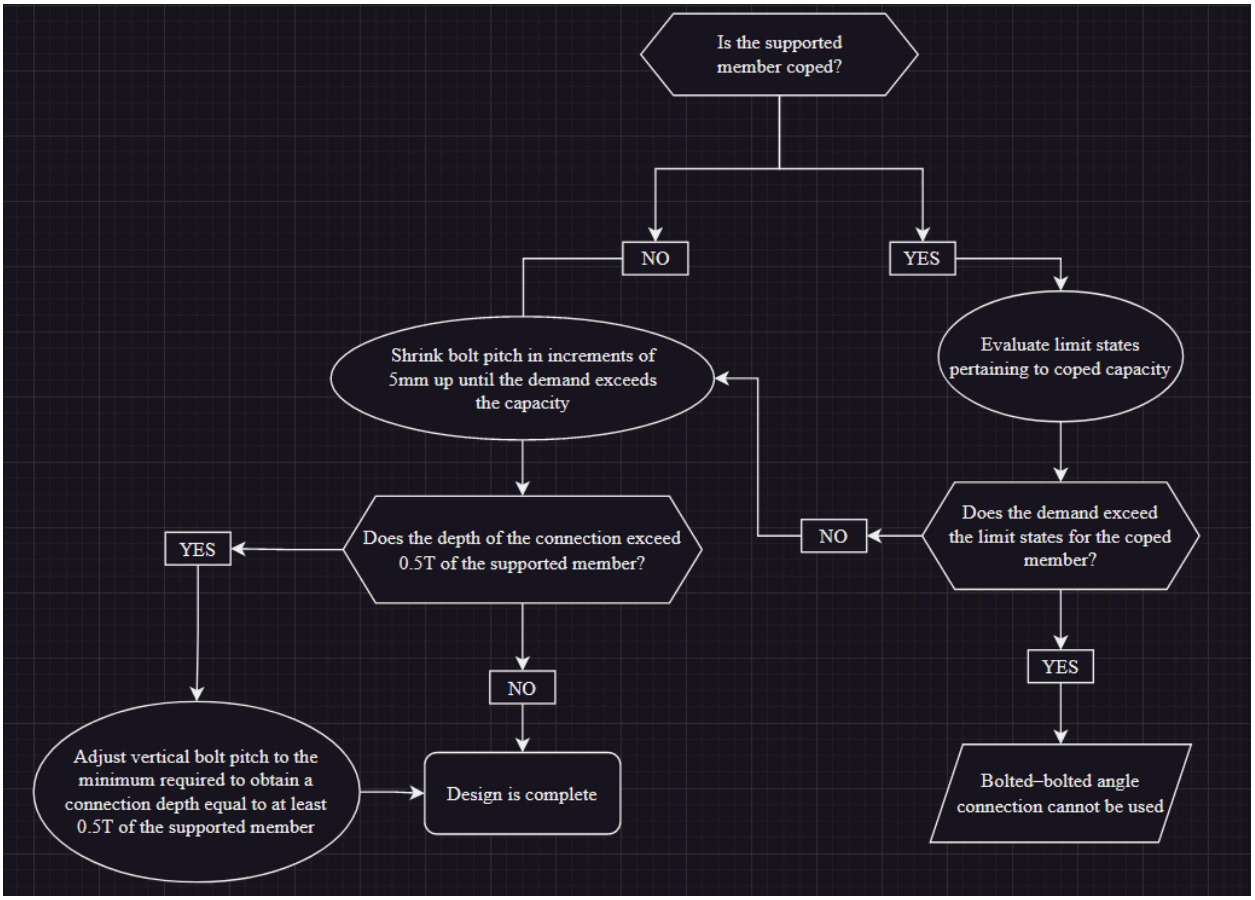

Is the supported member coped?
NO
YES
Shrink bolt pitch in increments of 5mm up until the demand exceeds the capacity
Evaluate limit states pertaining to coped capacity
YES
Does the depth of the connection exceed 0.5T of the supported member?
NO
Does the demand exceed the limit states for the coped member?
NO
YES
Adjust vertical bolt pitch to the minimum required to obtain a connection depth equal to at least 0.5T of the supported member
Design is complete
Bolted–bolted angle connection cannot be used

Figure 4-7: Flowchart of final step of optimization procedure for bolted–bolted angle connections

4.2.2 Welded–Bolted Angle Connections

The design and optimization of a welded–bolted angle connection incorporates several key parameters:

1. Angle designation: This parameter refers to the size of the angle that will be used. It is selected from a list of angles specified by the user before the optimization process.
2. Number of bolt rows: This parameter determines the number of rows of bolts that will be used in the connection.

3. Number of bolt columns: This parameter indicates the number of columns of bolts in the angle, with a maximum limit of 2.
4. Fillet weld leg size: This parameter refers to the size of the fillet weld that will be used in the connection.
5. Vertical bolt pitch: This parameter denotes the vertical distance between the bolt rows.
6. End distance: This parameter measures the distance between the outermost bolts and the top and bottom edges of the angle.
7. Orientation of the legs: This parameter determines how the short and long legs of the angle are connected to the supporting and supported members.
8. Location of the weld: This parameter indicates whether the angle is welded to the support or the supported member.
9. Wrapping of the angle: This parameter determines if the angle is wrapped around the web of the supported member, that is, the web is on the inside face of the angle.

These parameters are adjusted during the optimization of the connection such that the demand-to-capacity ratio is maximized while maintaining a ratio equal to or below 1.0. Similar to the bolted–bolted angle connections, the edge distance is also taken as the minimum allowable edge distance specified by S16-19. The gauges for the specific angle sizes are taken from the CISC Handbook of Steel Construction. An illustration of this type of connection is shown in Figure 4-8.

According to AISC (2019), it is more convenient to calculate the minimum base metal thickness required to align the available shear rupture strength of the base metal with that of the welds than the available strength of the connecting element. For fillet welds, the minimum base metal thickness requirement to match the shear rupture strength of the weld is

$$t_{min} = \frac{F_{EXX}\frac{\sqrt{2}}{2}D}{F_u}$$

where

D = leg dimension of the fillet weld

F_{EXX} = strength of the filler metal

F_u = specified minimum tensile strength of the base metal

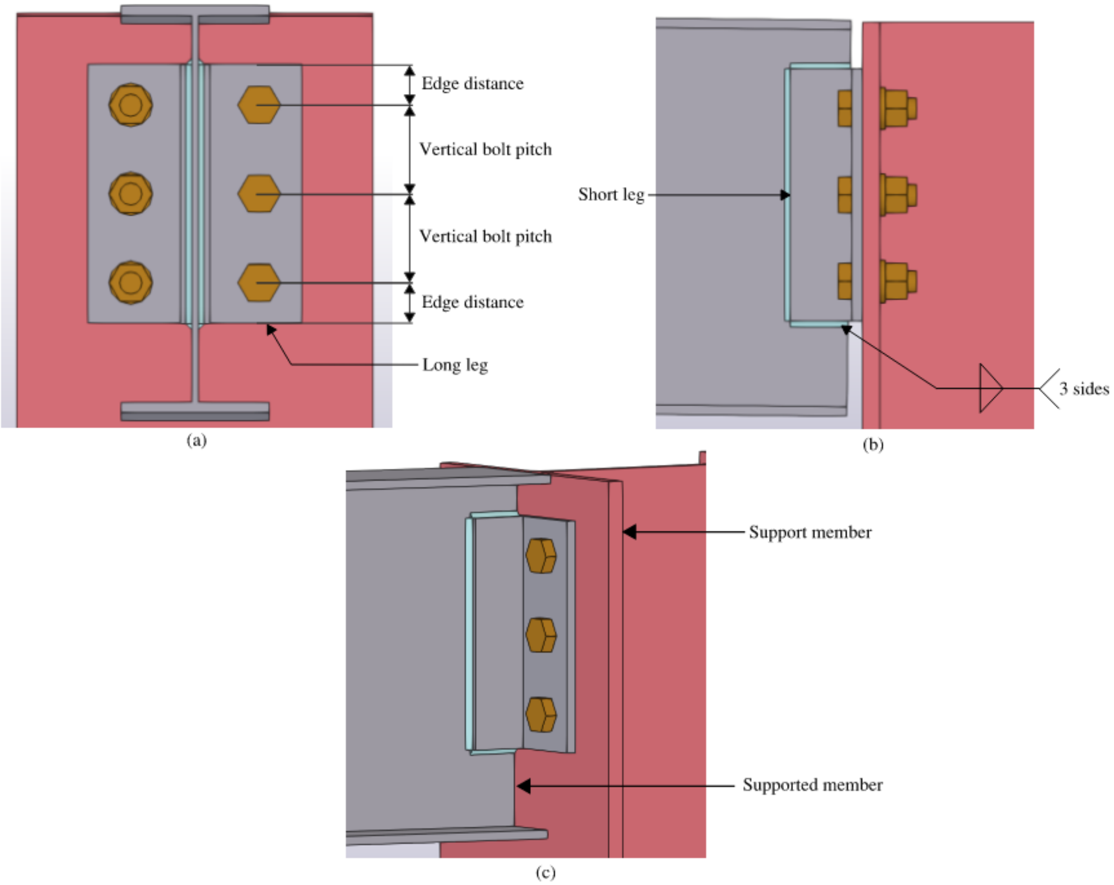


Figure 4-8: Welded–bolted angle connection of a 2L89x64x9.5 angle: (a) front view, (b) side view, (c) isometric view

In the case of double welded–bolted angle connections where fillet welds are applied on both sides of an element, particularly when welded to the web of the supported member, the minimum base metal thickness needed to align with the shear rupture strength of the weld is doubled.

For ease of erection, it is assumed that welded–bolted angle connections are welded to the support during the optimization procedure. While this assumption may not always hold true, welding the connection to the support also mitigates challenges related to the minimum base metal thickness requirement for welded–bolted double angle connections. For welded–bolted single angle connections, the weld is applied to the heel of the angle as opposed to the toe, and the angle wraps around the web of the supported member, as depicted in Figure 4-9. This arrangement serves to reduce the eccentricity from the centre of gravity of the weld group relative to the applied load and allows for additional spacing at the support member. Additionally, the short leg of the angle is welded to the support to further reduce the eccentricity of the weld group.

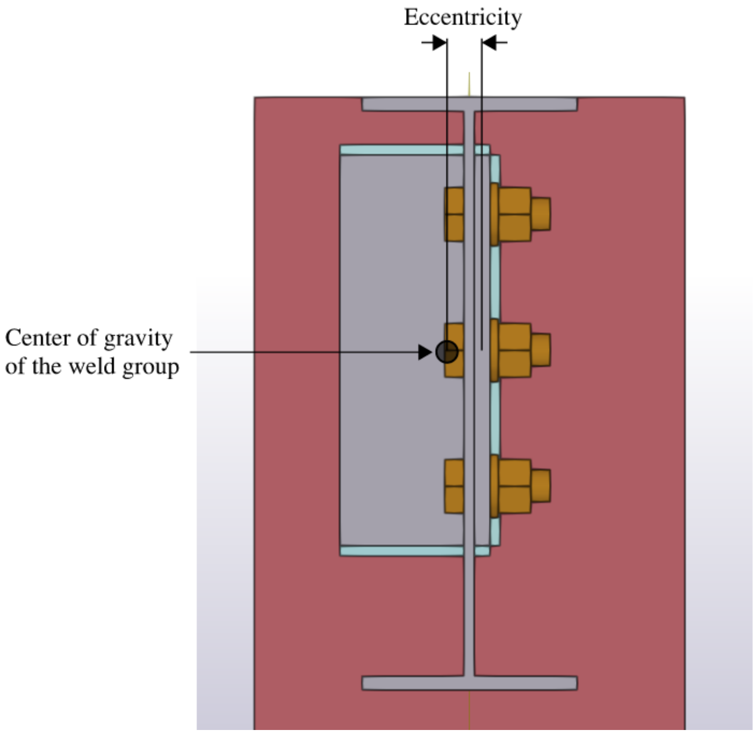


Figure 4-9: Eccentricity of weld group for a single welded–bolted angle connection

In order to maintain design integrity, the connection design module verifies compliance with various geometric constraints. These constraints encompass the following:

1. Support spacing: Verifies if the specified angle size and orientation fit within the available space provided by the support member.
2. Bolt spacing: Checks if the bolt pitch falls within the allowable range specified in CSA S16 for the given bolt size.
3. Bolt clashing: Examines whether the bolts would collide, following the detailing requirements outlined in the Canadian Handbook of Steel Construction for the given bolt size.
4. Edge distance: Verifies if the minimum edge distance requirement, as specified in CSA S16 for the given bolt size, is met.
5. Skew angle: Determines if the supported member is skewed relative to the support member, as a bolted–bolted angle connection cannot be used in such cases.
6. HSS support member: In the case where the support member is identified as an HSS column, welding the angle to the support member becomes necessary due to the limitation of the optimization tool, which does not currently support the utilization of through-bolts.

7. Weld size: Verifies that the fillet weld size adheres to the guidelines specified in the CISC handbook.
8. Depth of connection: Determines if the depth of the connection satisfies the connection stability requirement.
9. Connection space: Verifies the availability of sufficient space for the given connection detail.

The optimization procedures for welded–bolted single- and double- angle connections follow a similar approach, with the main distinction lying in the evaluation of limit states that are specific to each connection type. The optimization procedure can be divided into four steps. It begins with the initialization of design parameters and the filtering out of incompatible angle sizes, ensuring that only suitable angle sizes are considered for the optimization process. Next, an iterative process determines the minimum number of bolts and the corresponding angle size that satisfy the strength requirements. Subsequently, the minimum weld size is determined, and a final evaluation is conducted to identify the smallest suitable angle size that meets the strength requirements. In the case where the supported member is coped, the associated limit states are assessed and the material usage is minimized through the reduction of bolt pitch, all while ensuring the required strength requirements are met.

Figure 4-10 illustrates the initial step of the optimization procedure. In this step, the list of angles provided by the user is sorted in ascending order based on leg size and thickness. Simultaneously, the list of weld sizes provided by the user is also sorted in ascending order based on the size of the fillet welds. This sorting approach ensures that the angle and weld sizes are iterated from smallest to largest, facilitating the selection of the most optimal angle and weld size in subsequent steps. The initial design parameters are also shown in Figure 4-10. The number of bolt rows is set to a minimum of 2, with the option to add more bolts if necessary. Similarly, the number of bolt columns is also initially set to the minimum requirement of 1. The vertical bolt pitch is initially set as the maximum bolt pitch, the lesser of 14 times the thickness of the thinner connected part or 180mm as per AISC Specification J3.5 (AISC, 2019). This allows for gradual adjustments and fine-tuning of the bolt pitch by decreasing it before considering the addition of more bolts.

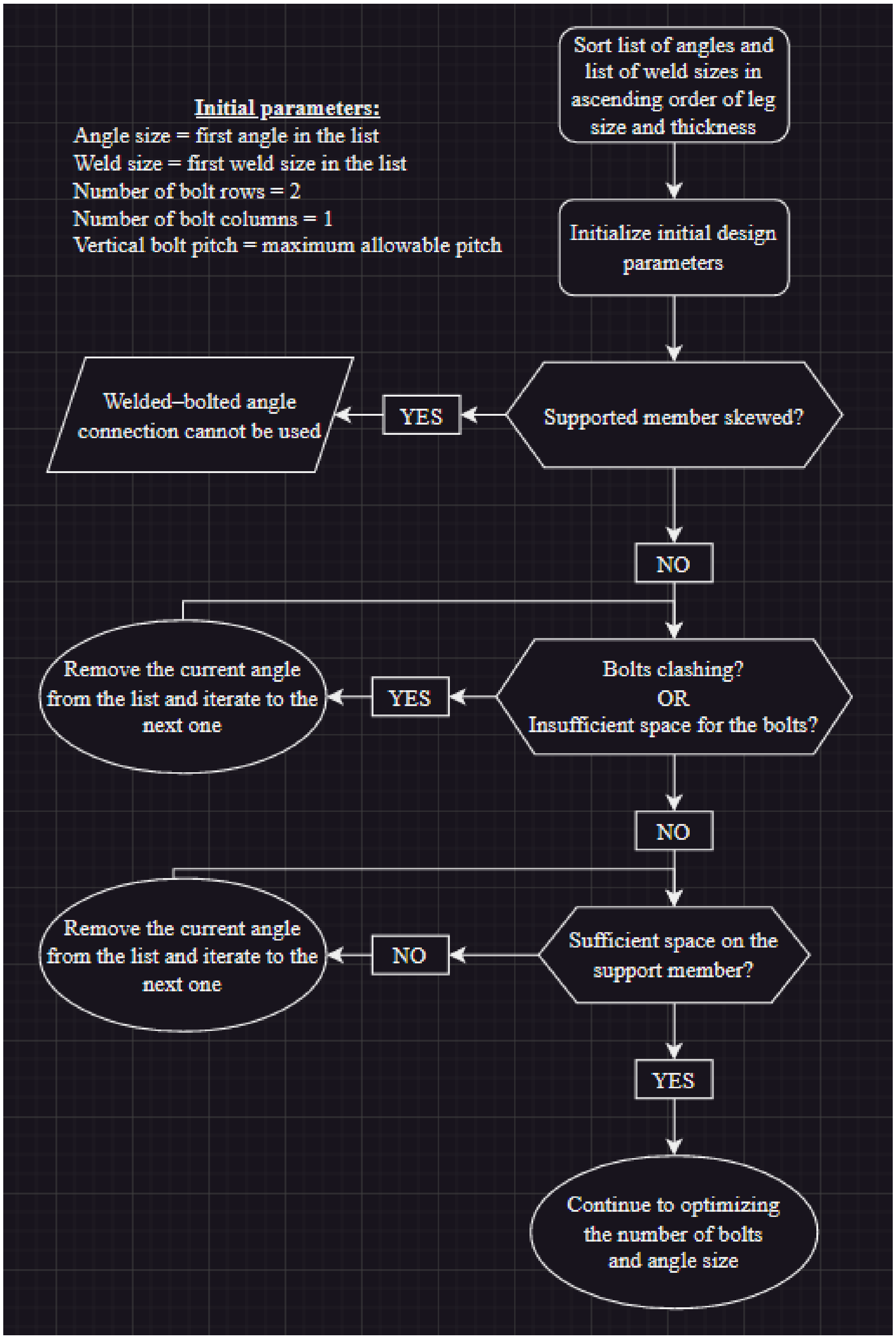

Sort list of angles and list of weld sizes in ascending order of leg size and thickness
Initial parameters:
Angle size = first angle in the list
Weld size = first weld size in the list
Number of bolt rows = 2
Number of bolt columns = 1
Vertical bolt pitch = maximum allowable pitch
Initialize initial design parameters
Supported member skewed?
YES
Welded–bolted angle connection cannot be used
NO
Bolts clashing? OR Insufficient space for the bolts?
YES
Remove the current angle from the list and iterate to the next one
NO
Sufficient space on the support member?
NO
Remove the current angle from the list and iterate to the next one
YES
Continue to optimizing the number of bolts and angle size

Figure 4-10: Flowchart of initial step of optimization procedure for welded–bolted angle connections

Figure 4-11 depicts the second step of the optimization procedure, which employs an iterative approach to ensure adequate connection space for accommodating additional bolts to satisfy the limit states pertaining to the bolt group. Concurrently, it iterates through the list of angles to identify the smallest angle size that meets the requirements imposed by the bolt group. Just like in bolted–bolted angle connections, there may be instances where it is not possible to achieve a suitable connection.

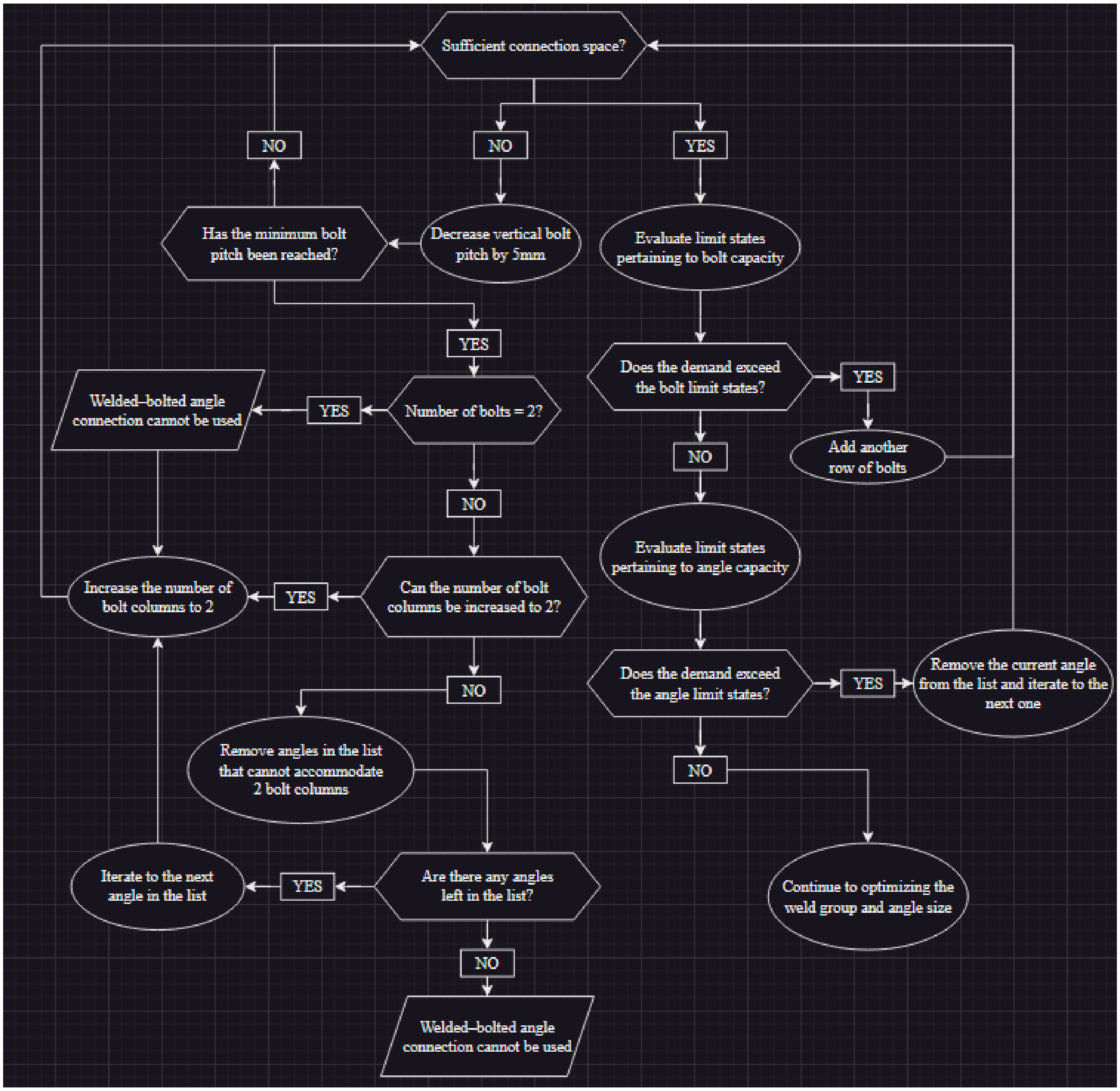

Sufficient connection space?
NO
NO
YES
Has the minimum bolt pitch been reached?
Decrease vertical bolt pitch by 5mm
Evaluate limit states pertaining to bolt capacity
YES
Does the demand exceed the bolt limit states?
YES
Welded–bolted angle connection cannot be used
YES
Number of bolts = 2?
Add another row of bolts
NO
NO
Evaluate limit states pertaining to angle capacity
Increase the number of bolt columns to 2
YES
Can the number of bolt columns be increased to 2?
Does the demand exceed the angle limit states?
YES
Remove the current angle from the list and iterate to the next one
NO
Remove angles in the list that cannot accommodate 2 bolt columns
NO
Iterate to the next angle in the list
YES
Are there any angles left in the list?
Continue to optimizing the weld group and angle size
NO
Welded–bolted angle connection cannot be used

Figure 4-11: Flowchart of second step of the optimization procedure of welded–bolted angle connections

Figure 4-12 provides an overview of the third step of the optimization procedure. In this section, an iterative process is used to determine the minimum weld size that will satisfy the limit states pertaining to the weld group. Unsuitable weld and angle sizes are iteratively removed from their respective lists until a suitable weld size is determined.

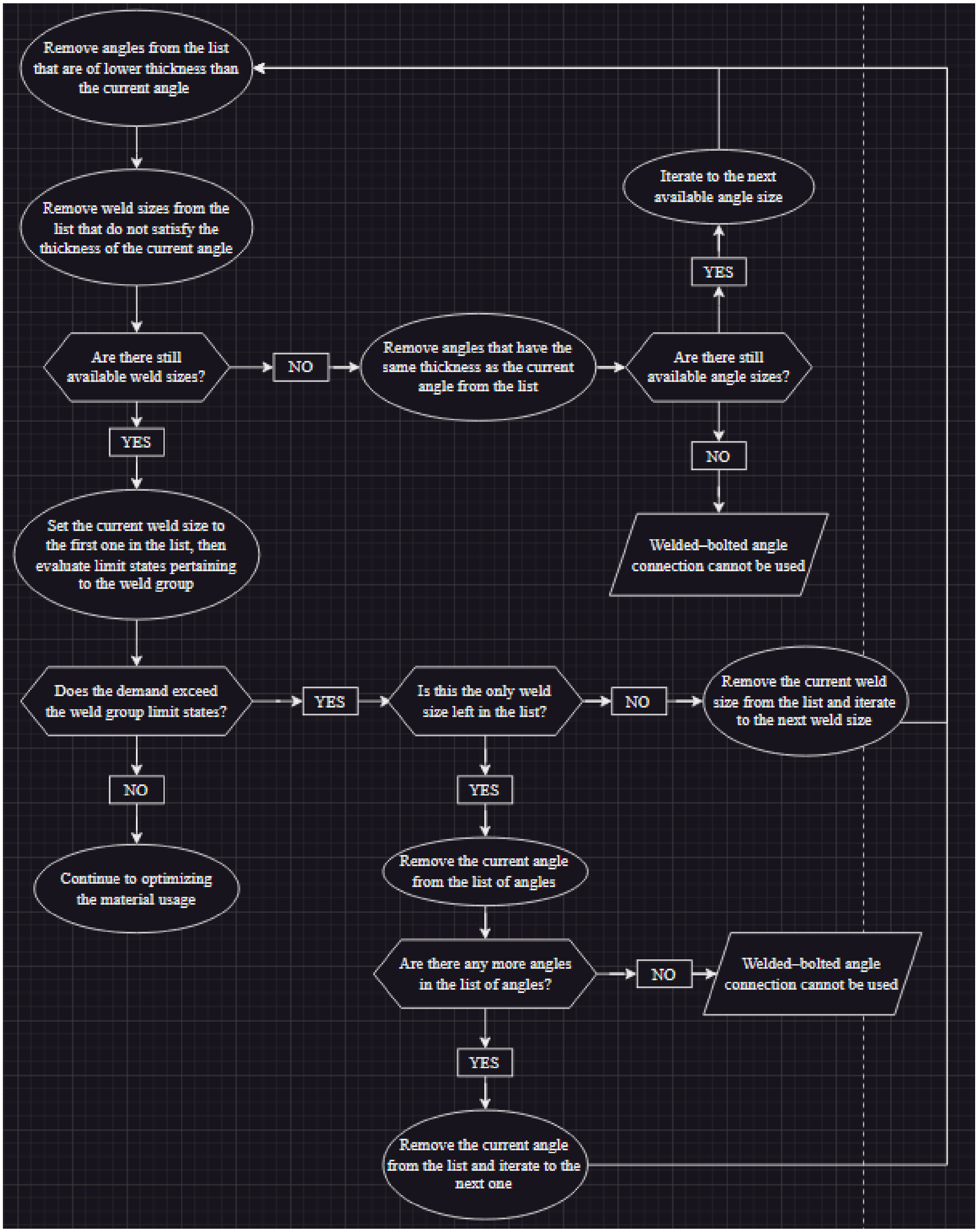
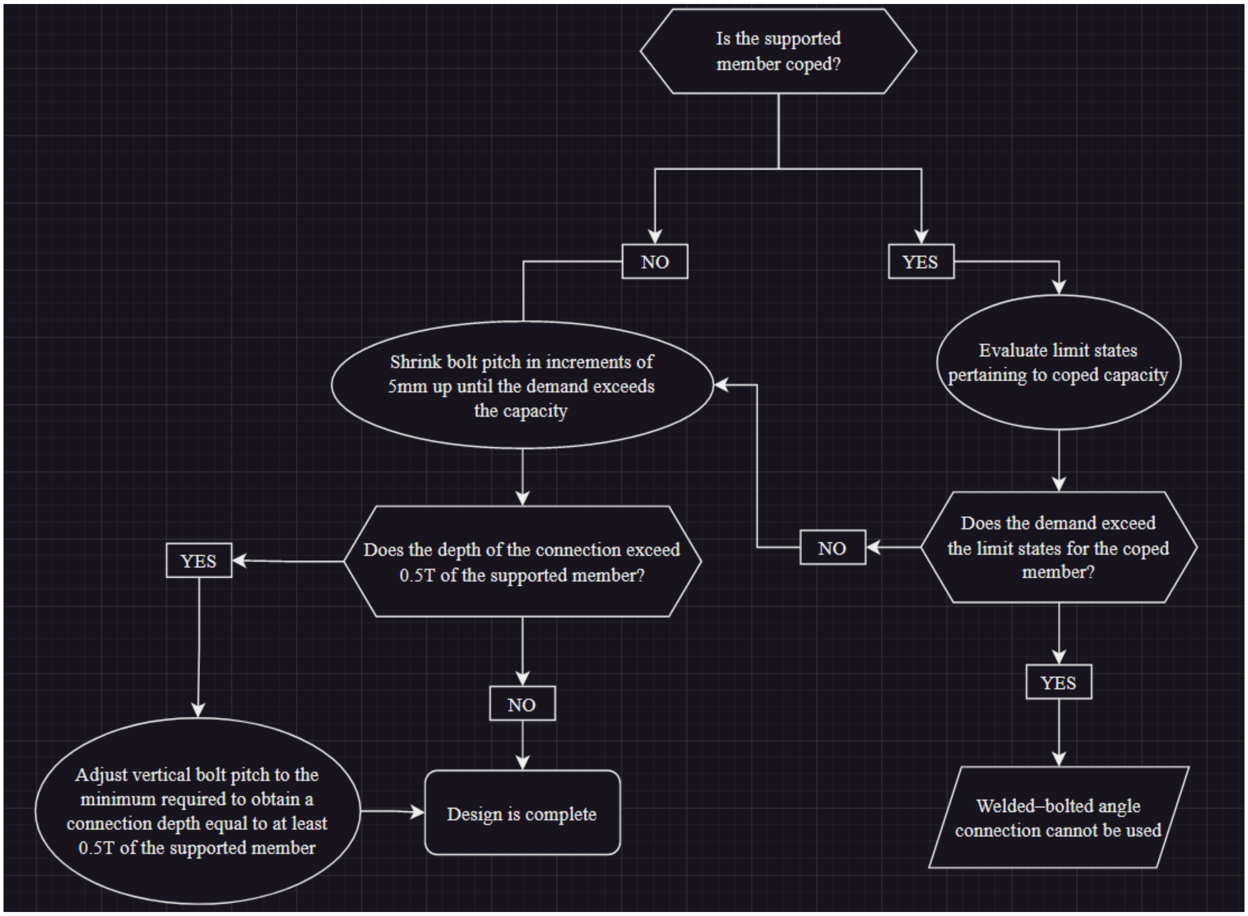

Remove angles from the list that are of lower thickness than the current angle
Remove weld sizes from the list that do not satisfy the thickness of the current angle
Are there still available weld sizes?
NO
Remove angles that have the same thickness as the current angle from the list
Are there still available angle sizes?
YES
Iterate to the next available angle size
NO
Welded–bolted angle connection cannot be used
YES
Set the current weld size to the first one in the list, then evaluate limit states pertaining to the weld group
Does the demand exceed the weld group limit states?
YES
Is this the only weld size left in the list?
NO
Remove the current weld size from the list and iterate to the next weld size
NO
Continue to optimizing the material usage
YES
Remove the current angle from the list of angles
Are there any more angles in the list of angles?
NO
Welded–bolted angle connection cannot be used
YES
Remove the current angle from the list and iterate to the next one

Figure 4-12: Flowchart of third step of optimization procedure for welded–bolted angle connections

Figure 4-13 provides an overview of the final section of the optimization procedure. Once the bolt group, weld group and angle size have been determined, the optimization script proceeds to evaluate whether the supported member is coped. In the case of a coped member, additional limit states specific to the coping are assessed. If the limit states for the coped member are met or if the supported member is not coped, the bolt pitch is then further reduced, aiming to minimize material usage while still satisfying the required strength criteria. The final connection design results in a welded–bolted angle connection with a maximized demand-to-capacity ratio while maintaining a ratio below 1.0.



Is the supported member coped?
NO
YES
Shrink bolt pitch in increments of 5mm up until the demand exceeds the capacity
Evaluate limit states pertaining to coped capacity
YES
Does the depth of the connection exceed 0.5T of the supported member?
NO
Does the demand exceed the limit states for the coped member?
NO
YES
Adjust vertical bolt pitch to the minimum required to obtain a connection depth equal to at least 0.5T of the supported member
Design is complete
Welded–bolted angle connection cannot be used

Figure 4-13: Flowchart of final step of optimization procedure for welded–bolted angle connections

4.2.3 Shear Tab Connections

The design and optimization of a shear tab connection incorporates several key parameters:

1. Plate thickness: This parameter denotes the thickness of the shear tab plate in mm.
2. Number of bolt rows: This parameter determines the number of rows of bolts that will be used in the connection.

3. Number of bolt columns: This parameter indicates the number of columns of bolts in the angle, with a maximum of 2.
4. Vertical bolt pitch: This parameter denotes the vertical distance between the bolt rows.
5. Horizontal bolt pitch: This parameter denotes the horizontal distance between the bolt columns.
6. End distance: This parameter measures the distance between the outermost bolts and the top and bottom edges of the plate.
7. Edge distance: This parameter measures the distance between the outermost bolts to the edge of the supported member and the plate.

Similar to the other types of connections, these parameters are adjusted during the optimization of the connection such that the demand-to-capacity ratio is maximized while still maintaining a ratio below 1.0. It is important to mention that the dimensions of the plate, including its length, width, and depth, are determined by the number of bolt rows and columns, the bolt pitch, and the edge and end distances. Both the vertical and horizontal edge distances are taken as the minimum allowable edge distance specified by S16-19. Likewise, the horizontal bolt pitch is established based on the minimum allowable bolt pitch specified in S16-19.

The weld size is determined in such a way that the plate yields before the weld fractures. Muir and Hewitt (2019) derived a ratio between the weld size and plate thickness based on the equations for weld rupture and plate yield strength, resulting in a closed-form solution. The equation is as follows:

$$w \geq \frac{t_p F_y \sqrt{3}}{2F_{EXX}}$$

where

w = fillet weld size

t_p = plate thickness

F_y = yield strength of the plate

F_{EXX} = strength of the filler metal

In their work, the Muir and Hewitt (2019) assumed F_y = 50 ksi and F_{EXX} = 70 ksi, leading to a required weld thickness equal to or greater than 5/8 of the plate thickness (Muir and Hewitt, 2009).

However, in the context of the optimization tool, F_{EXX} is assumed to be the strength of an E49XX electrode (490 MPa), and F_y is based on the user's selection, typically 300W steel for plates.

AISC outlines two distinct configurations for shear tab connections: the conventional configuration, which necessitates adherence to specific dimensional and other limitations outlined in the AISC Steel Construction Manual (AISC, 2019) and research conducted by Muir and Thornton (2011), and the extended configuration. The extended configuration comes into play when the dimensional and other limitations of the conventional method cannot be met. Both configurations are employed in the optimization tool and are considered in their respective circumstances. An example of a shear tab connection is illustrated in Figure 4-14.

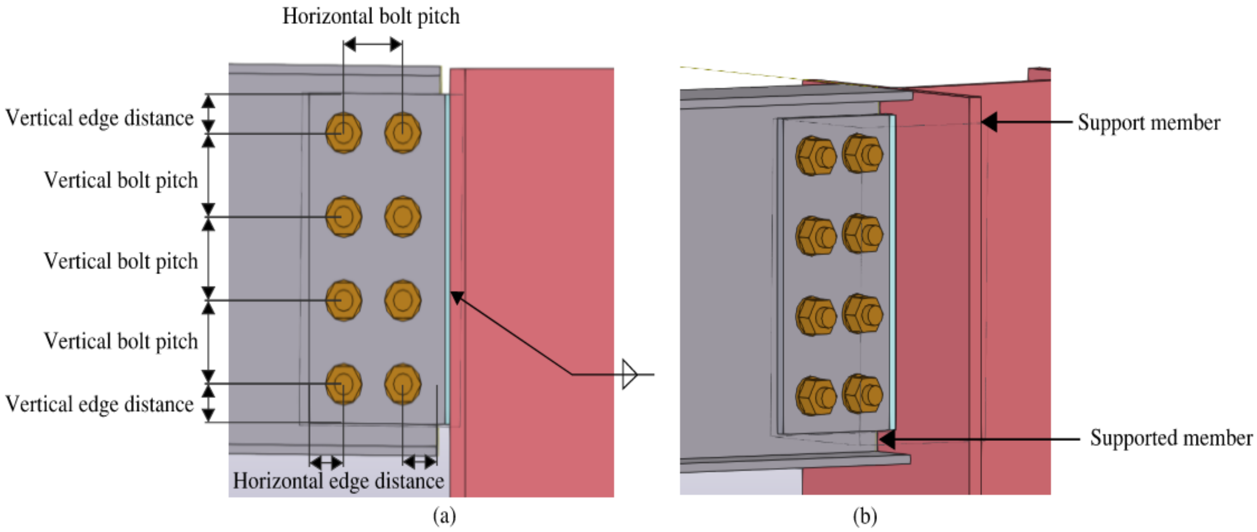


Figure 4-14: Shear tab connection: a) side view, b) isometric view

The various geometric and ductility constraints associated with shear tab connections are evaluated within the connection design module. These constraints are as indicated:

1. Maximum plate thickness: This criterion validates whether the plate thickness is sufficient to protect the bolt group by yielding before bolt shear occurs (Muir and Hewitt, 2009). The maximum plate thickness is determined based on the bolt group strength specified in the AISC Steel Construction Manual.
2. Bolt spacing: This check ensures that the bolt pitch falls within the allowable range specified in CSA S16 for the given bolt size.

3. Edge distance: This criterion verifies if the minimum edge distance requirement, as specified in CSA S16 for the given bolt size, is met.
4. Skew angle: This assessment determines whether the supported member is skewed more than 60° relative to the support member, as shear tab connections cannot be used in such cases.
5. Depth of connection: Determines if the depth of the connection satisfies the connection stability requirement.
6. Connection space: Verifies the availability of sufficient space for the given connection detail.

The optimization process for a shear tab connection comprises four key steps. It commences by initializing design parameters and identifying incompatible joints, specifically those where supported members have skew angles exceeding 60° relative to the line perpendicular to the support. Following that, an iterative approach is employed to establish the minimum number of bolts necessary to meet the strength criteria. A minimum plate thickness is then chosen to ensure it satisfies the strength requirements. In cases where the supported member is coped, the associated limit states are evaluated while simultaneously optimizing material usage through the reduction of bolt pitch.

Figure 4-15 visually presents the first step of the optimization procedure, which involves sorting the user-provided list of plate thicknesses in ascending order. This sorting approach ensures that plate sizes are iterated in an ascending order of thickness, ensuring the selection of the most optimal plate thickness when sizing the plate. The flowchart in Figure 4-15 also depicts the initial design parameters. The number of bolt rows is initially set to a minimum of 2, with the flexibility to add more bolts, if necessary, while the number of bolt columns is set to 1. To facilitate gradual adjustments and fine-tuning of the bolt pitch, the vertical bolt pitch is initially established as the maximum allowable value. This approach enables subsequent reductions in the bolt pitch before considering the addition of more bolts. Furthermore, the skew angle of the supported member relative to the support is verified to ensure it does not exceed 60°. This verification step is crucial, as a shear tab connection would not be suitable in cases where the skew angle exceeds the specified limit.

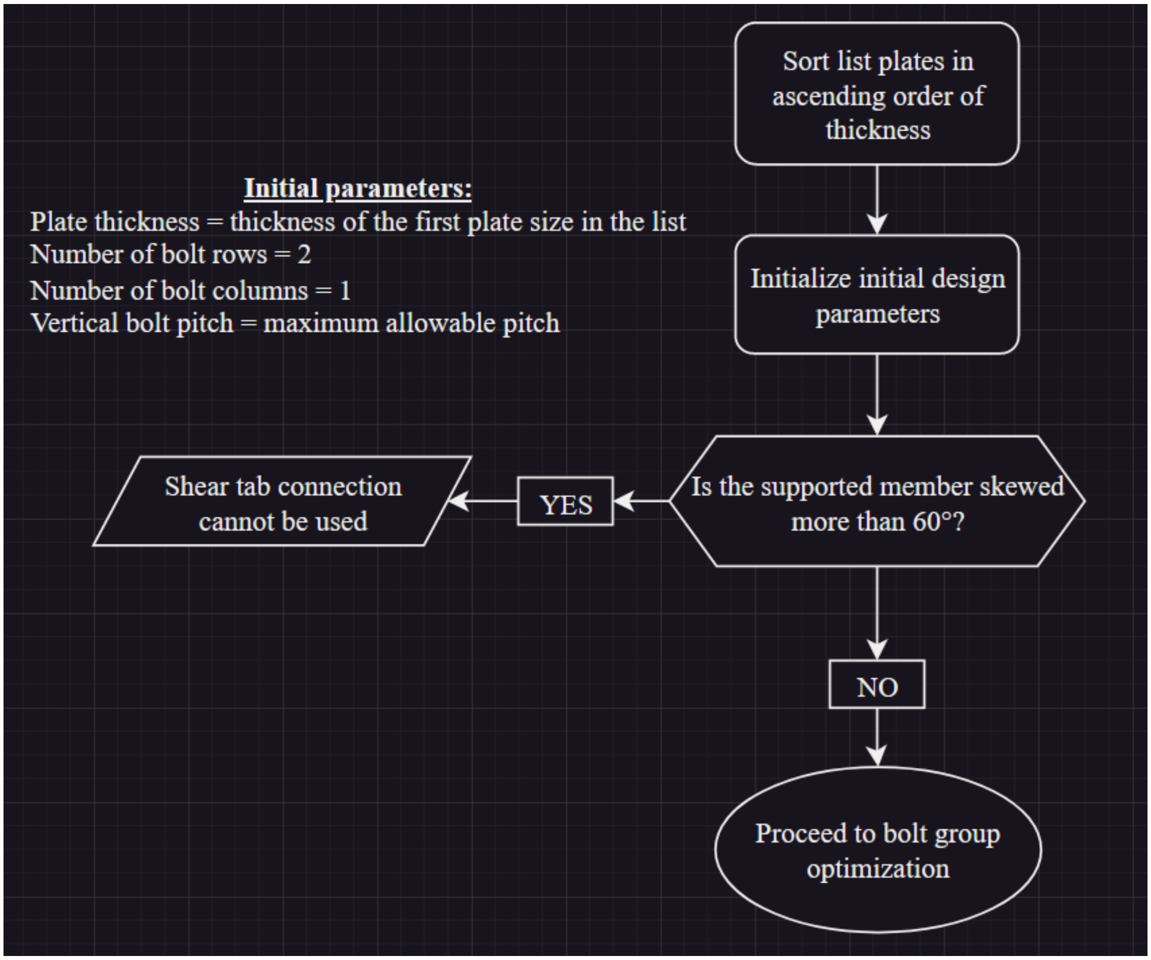

Sort list plates in ascending order of thickness
Initial parameters:
Plate thickness = thickness of the first plate size in the list
Number of bolt rows = 2
Number of bolt columns = 1
Vertical bolt pitch = maximum allowable pitch
Initialize initial design parameters
Shear tab connection cannot be used
YES
Is the supported member skewed more than 60°?
NO
Proceed to bolt group optimization

Figure 4-15: Flowchart of initial step of optimization procedure for shear tab connections

Figure 4-16 illustrates the second step of the optimization procedure, which utilizes an iterative method to ensure sufficient connection space for accommodating additional bolts that meet the limit states associated with the bolt group. In essence, this step involves analyzing the limit states related to the bolt group and determining if the demand-to-capacity ratio exceeds 1.0. If it does, additional bolt rows are added, provided there is adequate connection space available. Once the maximum connection space has been attained, the optimization process transitions to adding an extra bolt column instead, with a maximum of two bolt columns. Once the bolt group is optimized, the process proceeds to optimizing the plate of the shear tab connection. If it is not possible to achieve a bolt group configuration that satisfies the strength requirements, the shear tab connection will be deemed unsuitable in such cases.

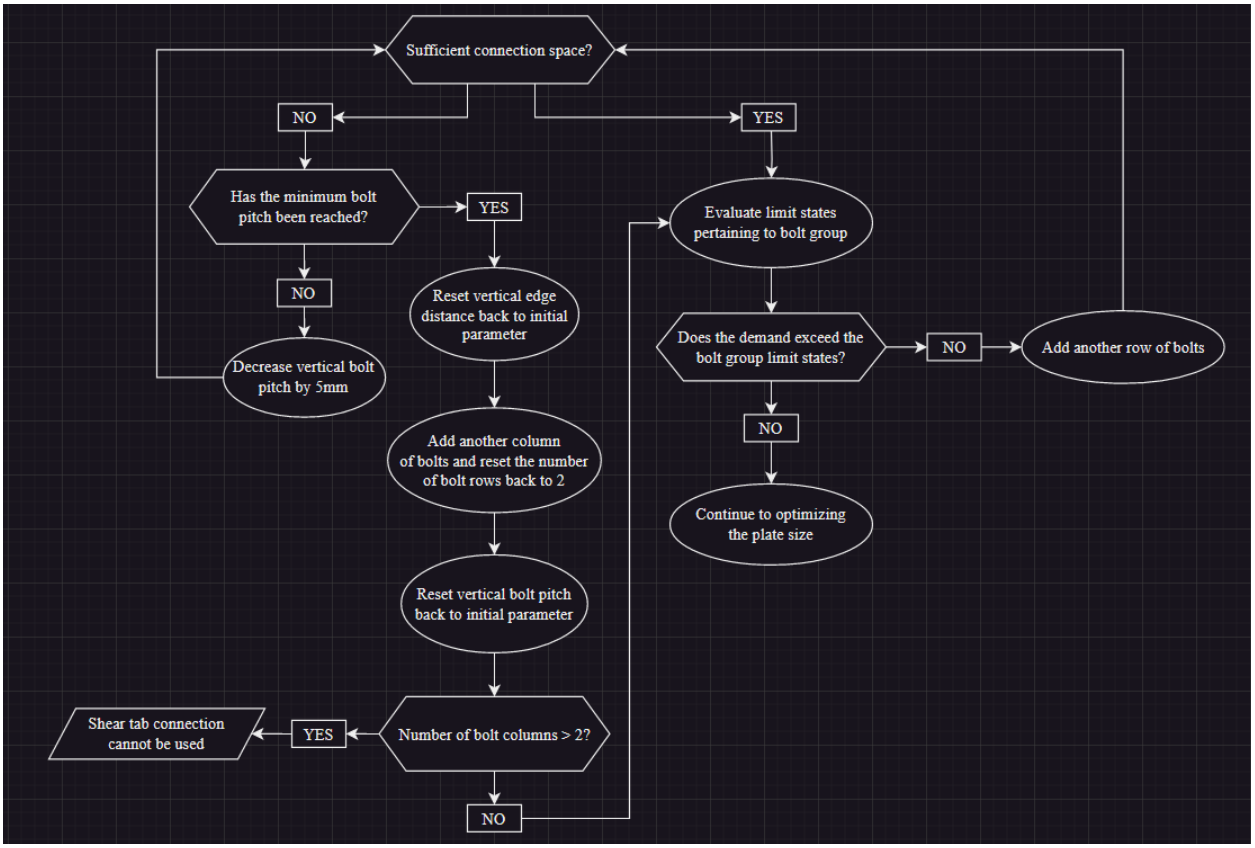

Sufficient connection space?
NO
YES
Has the minimum bolt pitch been reached?
YES
NO
Decrease vertical bolt pitch by 5mm
Reset vertical edge distance back to initial parameter
Add another column of bolts and reset the number of bolt rows back to 2
Reset vertical bolt pitch back to initial parameter
Number of bolt columns > 2?
YES
Shear tab connection cannot be used
NO
Evaluate limit states pertaining to bolt group
Does the demand exceed the bolt group limit states?
NO
Add another row of bolts
NO
Continue to optimizing the plate size

Figure 4-16: Flowchart of second step of optimization procedure for shear tab connections

Figure 4-17 illustrates the third step of the optimization procedure, which utilizes an iterative method to determine the minimum plate thickness required to satisfy strength requirements. Following this step, the procedure reaches its final step which aims to optimize the material usage depicted in Figure 4-18. The optimization script proceeds to evaluate whether the supported member is coped. In the case of a coped member, additional limit states specific to the coping are assessed. Similar to the other type of connections, if the limit states associated with the coped member fail to meet the required strength criteria, the connection type will be deemed unsuitable at this stage. This is because any further adjustments to the bolt group would lead to an unsatisfactory connection according to the bolt limit states. However, if the limit states for the coped member are met or if the supported member is not coped, the bolt pitch is then further reduced, aiming to minimize material usage while still satisfying the required strength criteria. The final connection design results in a shear tab connection with a maximized demand-to-capacity ratio while maintaining a ratio below 1.0.

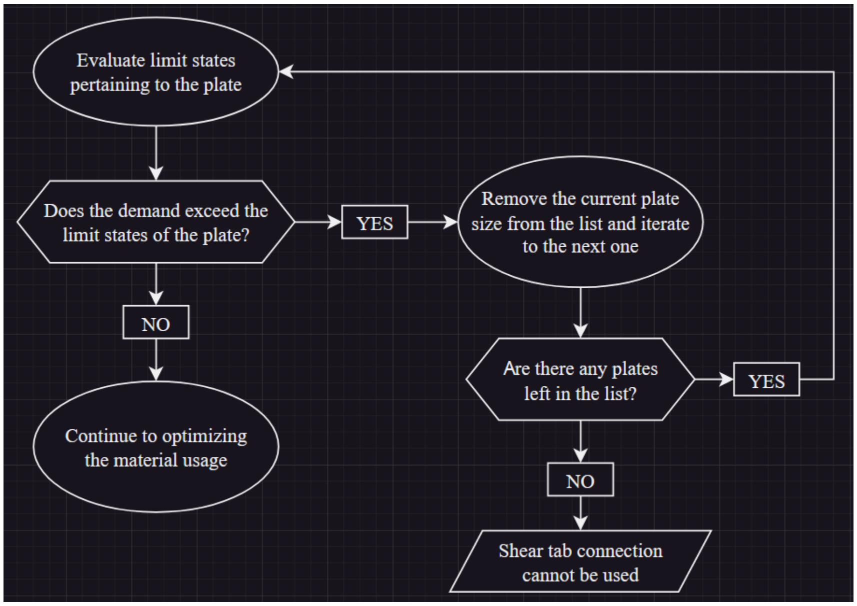

Evaluate limit states pertaining to the plate
Does the demand exceed the limit states of the plate?
YES
Remove the current plate size from the list and iterate to the next one
NO
Are there any plates left in the list?
YES
Continue to optimizing the material usage
NO
Shear tab connection cannot be used

Figure 4-17: Flowchart of third step of optimization procedure for shear tab connections

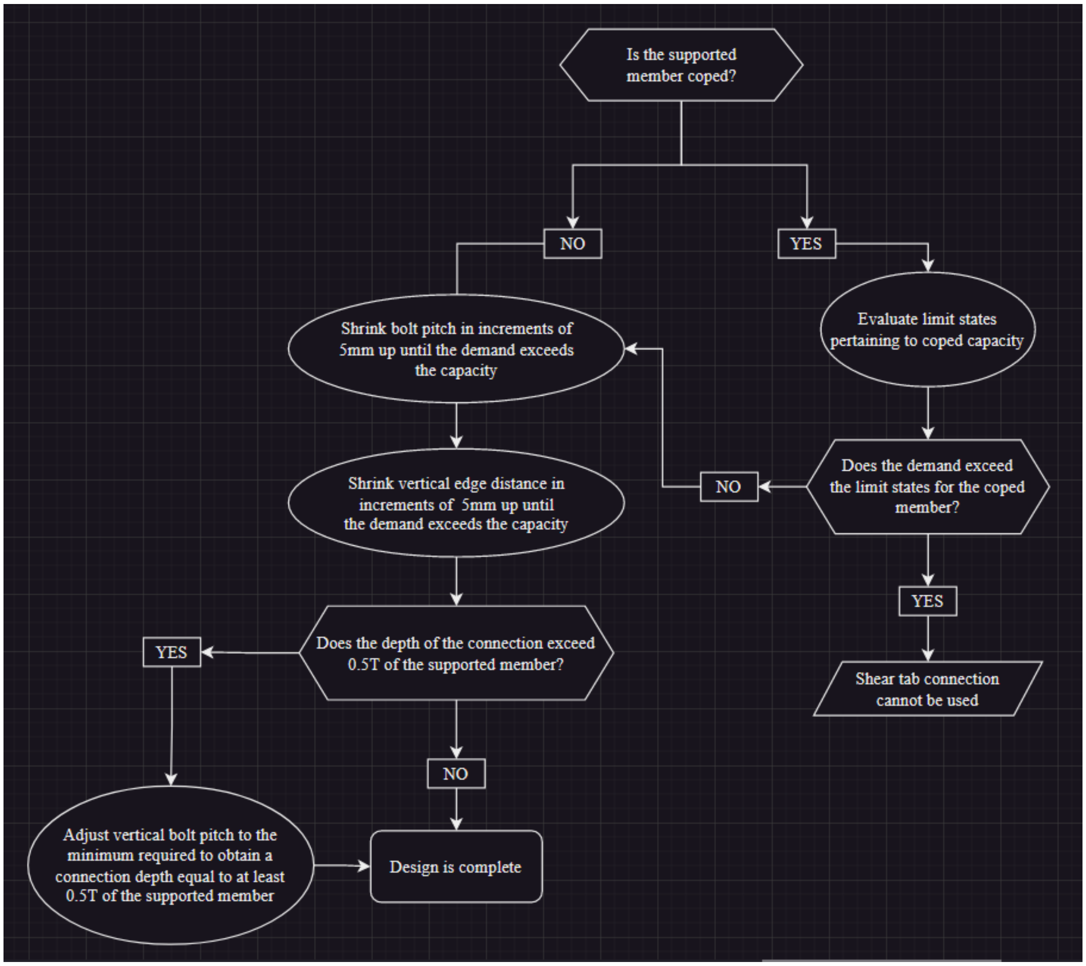

Is the supported member coped?
NO
YES
Shrink bolt pitch in increments of 5mm up until the demand exceeds the capacity
Evaluate limit states pertaining to coped capacity
Shrink vertical edge distance in increments of 5mm up until the demand exceeds the capacity
NO
Does the demand exceed the limit states for the coped member?
YES
Does the depth of the connection exceed 0.5T of the supported member?
YES
Shear tab connection cannot be used
NO
Adjust vertical bolt pitch to the minimum required to obtain a connection depth equal to at least 0.5T of the supported member
Design is complete

Figure 4-18: Flowchart of final step of optimization procedure for shear tab connections

4.3 Optimization Procedure

In order to reduce both the cost and weight of the structure, a two-tiered optimization strategy is implemented as illustrated in Figure 4-1. In the first optimization phase, multiple connections at each joint are designed and optimized using limit states design. In the subsequent phase, metaheuristics are employed to identify the most optimal combination across the entire structure. While the cost function inherently incorporates the weight of the structure, it is important to explicitly consider weight as an additional objective in the optimization process to evaluate its impact on the overall cost. This approach not only yields valuable insights into the effectiveness of the optimization procedure for multi-objective problems, but also provides insight for future research involving other objectives. Since this optimization problem involves minimizing both cost and weight, the process generates a range of solutions for the user to further refine or utilize immediately. Finally, the optimization tool automatically details the preferred solution and its corresponding connection details directly into the user's Tekla model. The rationale behind this approach is the immense scope of the search space, which would have posed a significant computational burden had a single optimization procedure been employed. The utilization of metaheuristics under such circumstances would have rendered the task highly computationally expensive.

To illustrate this point, it is important to note that each joint requires its own set of variables, representing the parameters for designing the specified shear connection type. As a result, buildings with thousands of connections would necessitate tens, if not hundreds, of thousands of variables to be evaluated before a single solution could be obtained. Considering that metaheuristics typically require thousands of evaluations to approach the global optimum, the scale of this task becomes prohibitively large. By dividing the optimization procedure into two distinct layers, the design of connections can be accomplished using scripts outlined in Section 4.2, rather than relying on metaheuristics. This approach considerably enhances the speed of the optimization process. Subsequently, each joint will have multiple potential options, which can serve as variables to search for the optimal combination of connections throughout the entire structure to minimize both the overall cost and weight. Moreover, since the connections have already been designed, evaluating an individual solution—comprising the combination of all connections in the structure—and determining its total cost and weight becomes a straightforward mathematical operation executed by the computer.

To initiate the optimization process, the necessary shear connections at the joint are extracted from the Tekla model. Along with this, design information is also retrieved, including the elevation difference between the supporting and supported members, the skew and slope angles between them, as well as the framing configuration, among other relevant details. Subsequently, the initial optimization phase commences. At each joint necessitating a shear connection, every specified type of shear connection designated by the user undergoes design and optimization, adhering to the procedures outlined in Section 4.2. Consequently, a joint may present up to five potential shear connections that can be utilized. It is worth highlighting that for a building comprising approximately 400 connections, the entire process can be completed within five seconds. Given that the existing library of connections comprises only five types of shear connections, there may arise situations where none of these pre-defined connections can adequately satisfy the given design force and connection space requirements. In such instances, the joint is excluded from the optimization process, and a flag is raised to alert the user. This flag signifies that separate evaluation and consideration must be undertaken for these joints outside the scope of the optimization process.

Afterwards, metaheuristics is employed for the second optimization phase. In this phase, each joint is treated as a variable within the search process, with each potential connection option for that joint considered as a distinct variable value. A solution is defined as the combination of all connections within the structure, and its overall cost and weight are evaluated using the cost function specified in Section 3.5.

As all connections at each joint already fulfill strength and geometric requirements, there is no need for constraints related to these aspects during the evaluation of solutions using metaheuristics. However, in the case of W-shape column or girder web configurations involving bolted–bolted angle connections, erection issues may arise. These erection issues, referred to as double connections, occur during field-bolted construction when beams or girders frame opposite each other. Double connections pose safety concerns when they arise in the web of a column or the web of a beam that continuously spans over the top of a column, with all field bolts utilizing the same open holes. Ensuring a secure connection becomes crucial when erecting the first member, while simultaneously bringing the second member into its final position (AISC, 2019). As the current optimization tool lacks the ability to consider framing intricacies like staggered angles, clipped angle connections, or temporary erection seats, which are crucial for establishing a secure connection while maneuvering the second member into its final position, a constraint function is implemented to effectively tackle this challenge. This constraint function penalizes solutions that involve such connection configurations, steering the optimization process away from favoring them. Consequently, this approach eliminates the potential issues associated with double connections.

Upon completion of the optimization process, the top hundred solutions are exported to a spreadsheet, providing a comprehensive summary of potential solutions. This spreadsheet includes a concise overview of the minimal cost and minimal weight options. In addition, another sheet quantifies data pertaining to all the solutions, presenting information such as the total count of connection types employed, cumulative lengths of various angle sizes, collective area of different plate sizes, overall weight, weight of the connection elements, total cost, and combined weight of different fillet weld sizes, among other relevant metrics. Following the generation of potential solutions, the user can analyze the different options and select the most suitable one according to their needs. Upon making a decision, the user can specify their chosen option, prompting the optimization tool to automatically incorporate the detailed connections directly into their Tekla model. This streamlined process allows users to readily visualize the outputs to gain a better understanding of the designs.

In the second layer of optimization, three distinct metaheuristic algorithms were utilized: NSGA-II, SPEA2, and SMPSO, as detailed in Section 3.6. To facilitate a fair comparison of their effectiveness, the parameters of each algorithm were adjusted to be identical. For instance, the NSGA-II algorithm had a population size of 100, the SMPSO algorithm had a swarm size of 100, and the SPEA2 algorithm had a population size of 80 with an archive size of 20, resulting in an overall population of 100. Furthermore, a maximum evaluation limit of 500,000 was imposed on all three algorithms, which was calibrated based on the results of the case studies discussed in Chapter 5. Consistent parameters were also established for the corresponding crossover, mutation, and selection operators required by the algorithms. By maintaining uniformity in these settings, a consistent basis for evaluating and contrasting the performance of each algorithm was established.

To address the optimization problem at hand, an integer encoding approach was employed. This encoding method was chosen as each variable represents a joint, encompassing all potential connection options specific to that joint. To illustrate this, consider a 2-storey planar frame depicted in Figure 4-19. The red circles in the figure represent the joints where shear connections are required, and a total of six connections are required. Assuming that each joint can be any of the five potential shear connection types and have been optimized in the initial layer of optimization, each variable can range from 0 to 4, representing a distinct shear connection. An individual solution to this problem would then be represented by a string of integers, such as 020341, which comprises the collection of all these variables. Since an integer encoding scheme is employed, the single-point crossover operator and bit flip mutation operator were utilized as they are compatible with integer encodings. Further details on these operators are provided below.

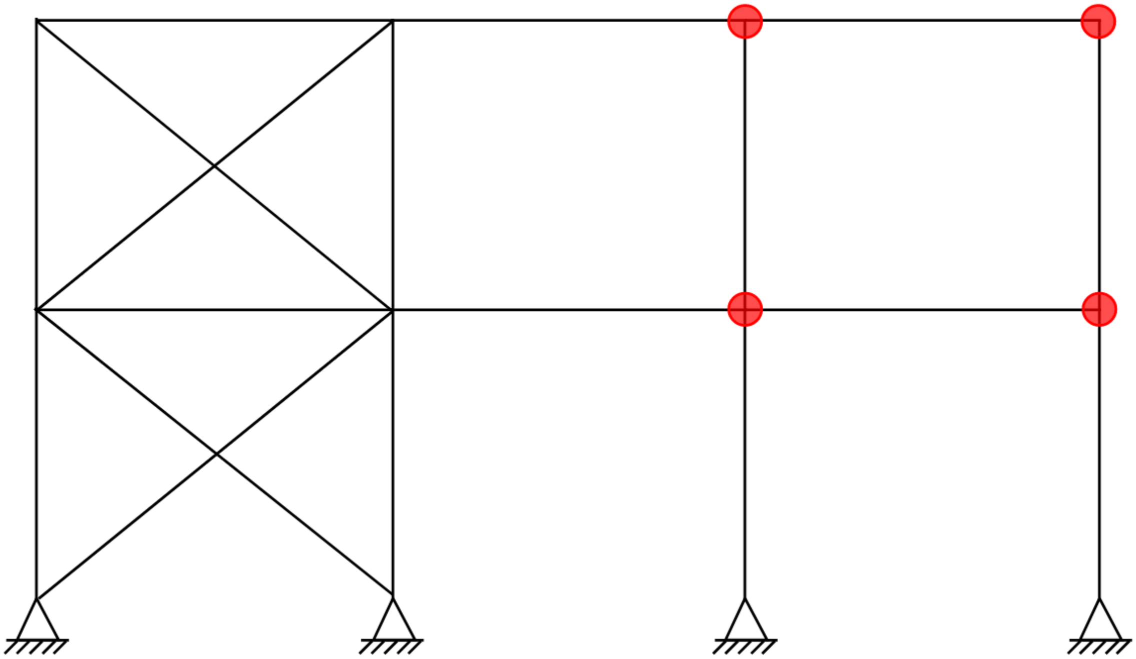


Figure 4-19: 2-storey planar frame with red circles indicating where shear connections are required

The NSGA-II and SPEA2 algorithms utilized the single-point crossover operator, which involves selecting a crossover point on the parent organism string and swapping all data beyond that point between the two parent organisms (Deb and Agrawal, 1995), as illustrated in Figure 4-20. The probability and distribution index for this crossover operator were both set to 0.9. Additionally, all three algorithms employed the bit flip mutation operator, which randomly selects one or more bits and flips them to introduce changes in the solution (Chicano et al., 2015), as depicted in Figure 4-21. The distribution index for the bit flip mutation was set to 0.9, and the probability of mutation was determined as 1 divided by the number of variables in the optimization problem, corresponding to the number of joints. For the NSGA-II and SPEA2 algorithms, a binary tournament selection process was employed as the selection operator. This process involves randomly selecting individuals from a population and passing those with the best fitness to fill the mating pool (Miller and Goldberg, 1995), as depicted in Figure 4-22.

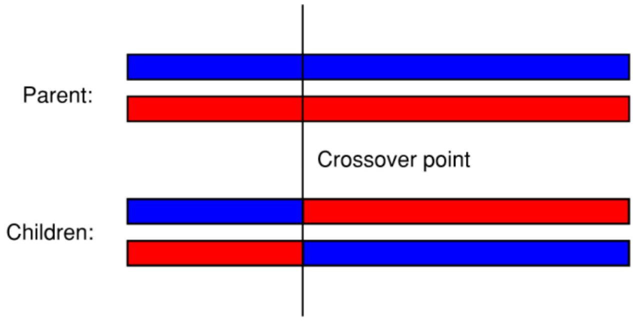


Figure 4-20: Single-point crossover in genetic algorithms

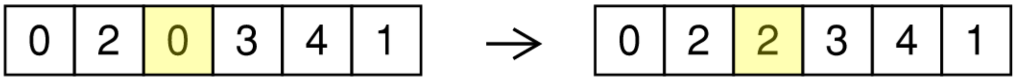


Figure 4-21: Bit flip mutation in genetic algorithms

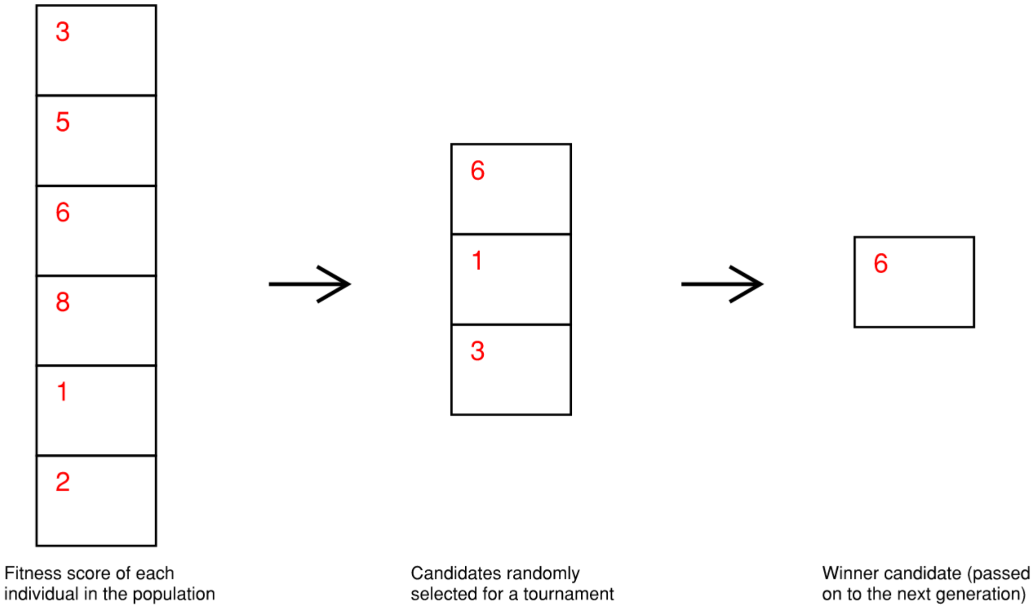

3
5
6
8
1
2
6
1
3
6
Fitness score of each individual in the population
Candidates randomly selected for a tournament
Winner candidate (passed on to the next generation)

Figure 4-22: Binary tournament selection in genetic algorithms

The chosen values for the probability and distribution index for the crossover and mutation operators align with the default settings provided in jMetal and have been widely used in test problems across various literature (Chicano et al., 2015; Deb et al., 2002; Deb and Agrawal, 1995; Zitzler et al., 2001). Although it is important to acknowledge that optimal parameter settings may differ for various optimization problems (Hamdan, 2014), it is not a primary concern in the context of the current research problem. Given the individual preferences and diverse characteristics of building projects among users, it is unrealistic to seek a universally applicable set of optimal parameters for the optimization problem.

The performance of these algorithms in the context of this research problem is evaluated and compared through case studies conducted and discussed in Chapter 5.

5 RESULTS AND DISCUSSION

5.1 Introduction

In this chapter, we examine two case studies that employed the optimization tool to evaluate its real-world effectiveness. The first case study involved a 2-storey steel building for a car dealership in Grand Prairie, Alberta, while the second case study focused on a 5-storey steel storage building. The steel fabricator responsible for both buildings shared the projects in the form of a Tekla model that included their connection designs along with the building's corresponding structural drawings.

As discussed before, the optimization tool is currently only capable of designing and optimizing shear connections. Therefore, in line with this functionality, the shear connections from both fully designed buildings were filtered and extracted, followed by the application of the cost function detailed in Section 3.5 to estimate the total material and fabrication cost of both projects based on the actual connections that were designed. Subsequently, the Tekla models were stripped of their connections, and the design loads were taken from the structural drawings. The optimization tool was then deployed, allowing for an examination and comparison of the resulting shear connections against the actual connections utilized in the project. Furthermore, a comprehensive analysis was conducted to compare and evaluate the overall cost implications.

To further explore the capabilities of the optimization tool, additional tests were carried out using varying design loads to assess the effects of higher and lower design forces. Additionally, an extensive comparison was made among the NSGA-II, SPEA2, and SMPSO algorithms to identify the algorithm that yielded the most optimal results and assess its corresponding execution time.

The Grand Prairie Honda and 5-Storey Affordable Storage case studies are explored in Sections 5.2 and 5.3, respectively. In Section 5.4, the findings of these case studies are examined, providing an in-depth discussion on the outcome and feasibility of the optimization tool.

5.2 Grand Prairie Honda Case Study

The Grand Prairie Honda project is a 2-storey steel structure designed in 2021. The structure comprises 381 shear connections that are designed for 60% of the plastic moment capacity of the supported member, assuming the member is laterally braced. The unit rates specified for the steel, bolt, and labour costs are tabulated in Table 5-1. The unit rates used in this study were recommendations provided by a local steel fabricator (Collins Steel, personal communication, 2023) and may not represent a reliable estimate for other steel fabricators.

Table 5-1: Material and labour unit rates utilized in the case studies

Description	**Unit Rate**
Structural Steel	$2.31/kg
Bolts	$2.00/bolt
Shop Operator Cost	$100/hr
Shop Fitter Cost	$100/hr
Shop Welder Cost	$150/hr

To determine the overall material and fabrication cost of the structure, connections that did not fall under the category of shear connections were eliminated from the Tekla model. Afterwards, the existing shear connections were extracted and categorized based on the type of connection that was used and its details are summarized under the “Fabricator” column in Table 5-2. In the table, the number in brackets represents the total quantity of stock material required, based on a stock plate size of 60” x 144” and a stock angle length of 40’. To obtain the overall weight of the structure, information regarding the structural steel members, such as their length and profile, was also extracted from the Tekla model. Subsequently, using the overall weight of the structure and the details of the shear connections, the cost function was applied to calculate both the total material and fabrication cost of the entire structure, as well as the total cost specifically attributed to the fabrication of the shear connections. This allowed for an assessment of the cost implications associated with both the structure as a whole and its shear connections in isolation.

Afterwards, the Tekla model was stripped of its existing connections, while the user settings were modified to ensure that the optimization process would adhere to the fabricator’s preferences in terms of the angle designations. To accurately assess the tool's performance in line with the fabricator's intentions, we opted to maintain the same angle designations and specified a no-coping preference. The optimization tool was then employed three times using the three metaheuristics discussed in Section 3.6 and 4.3: NSGA-II, SPEA2, and SMPSO. Each metaheuristic was set such that 500,000 total evaluations were conducted before terminating. The lowest cost option obtained from each metaheuristic is summarized in Table 5-2.

Table 5-2: Summary of lowest cost option for Grand Prairie Honda determined from the optimization tool using different metaheuristics

Description	Quantity			
	Fabricator	NSGA-II	SPEA2	SMPSO
Bolted-Bolted Single Angle	0	0	1	22
Welded–Bolted Single Angle	123	92	102	51
Bolted–Bolted Double Angle	21	0	1	23
Welded–Bolted Double Angle	0	13	12	36
Shear Tab	237	276	265	249
Area of PL10 (m^2)	8.91 (2)	7.03 (2)	6.87 (2)	5.99 (2)
Area of PL13 (m^2)	0.48 (1)	0.19 (1)	0.19 (1)	0.15 (1)
Area of PL16 (m^2)	0.84 (1)	0	0	0
Length of L89x64x9.5 (m)	33.94 (3)	23.11 (2)	24.69 (3)	12.20 (2)
Length of L89x89x9.5 (m)	7.52 (1)	0	0.56 (1)	20.47 (2)
Length of L102x76x9.5 (m)	0.84 (1)	0	0	0
Number of Bolts	1658	1158	1165	1261
Total Weight (kg)	92,386	92,399	92,414	92,486
Cost of Shear Connections	$29,267	$27,695	$27,545	$37,664
Material and Steel Fabrication Cost	$276,459	$274,887	$274,736	$284,856
Execution Time	N/A	6m 26s	7m 9s	5m 57s

Additional supplementary tests were carried out using the NSGA-II algorithm, encompassing various design loads and coping preferences. These parameters encompassed 60% of the plastic moment of the supported member with a coping preference, 50% of the plastic moment of the supported member with and without a coping preference, and 50% of the shear resistance of the supported member with and without a coping preference. These tests were conducted to assess the effectiveness of the tool across varying design loads and to observe how the results differ under different parameters. Figure 5-1 showcases the total material and steel fabrication cost associated with each design load and Figure 5-2 focuses only on the cost associated with the shear connections.

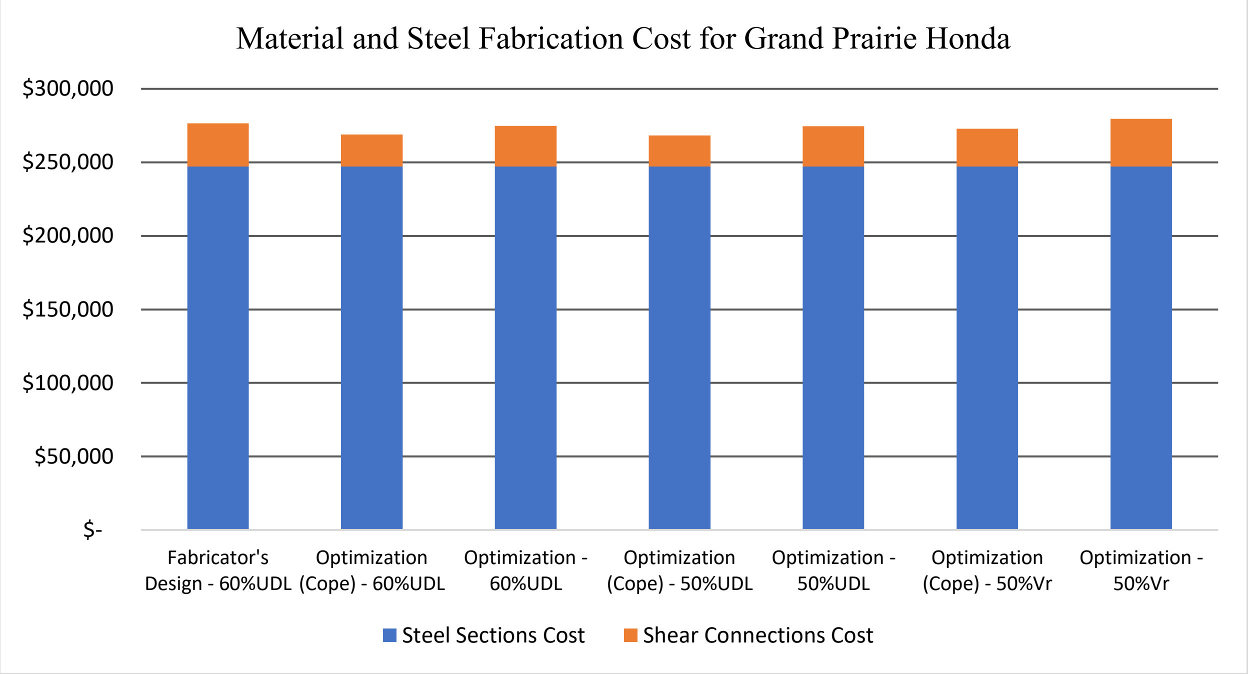


Figure 5-1: Material and steel fabrication cost for Grand Prairie Honda under various design loads using the optimization tool using NSGA-II algorithm

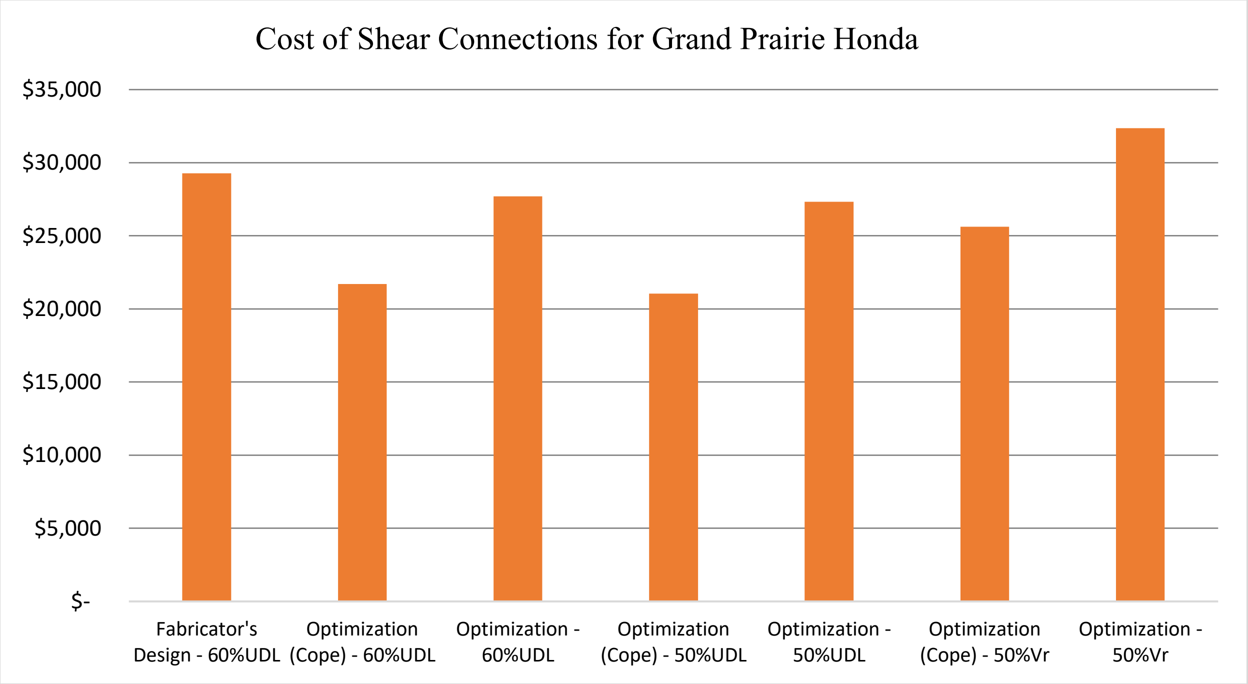


Figure 5-2: Cost of shear connections for Grand Prairie Honda under various design loads using the optimization tool using NSGA-II algorithm

5.3 5-Storey Affordable Storage Case Study

The 5-Storey Affordable Storage project is a 5-storey steel structure designed in 2022. The structure comprises 1415 shear connections that are designed for 60% of the plastic moment capacity of the supported member, assuming the member is laterally braced. The unit rates specified for the steel, bolt, and labour cost are tabulated in Table 5-1. Similar to the other case study, the unit rates used in this study were recommendations provided by a local steel fabricator and may not represent a reliable estimate for other steel fabricators.

Similar to the Grand Prairie Honda case study, connections that did not fall under the category of shear connections were eliminated from the Tekla model. Afterwards, the existing shear connections were extracted and categorized based on the type of connection that was used. The existing connections in the Tekla model were then removed, while the user settings were modified to ensure that the optimization process would adhere to the fabricator's preferences in terms of the angle designations. The optimization tool was then employed three times using three different metaheuristics: NSGA-II, SPEA2, and SMPSO. The lowest cost option obtained from each metaheuristic is summarized in Table 5-3. In the table, the number in brackets represents the total quantity of stock material required, based on a stock plate size of 60" x 144" and a stock angle length of 40'.

Table 5-3: Summary of lowest cost option for 5-Storey Affordable Storage determined from the optimization tool using different metaheuristics

Description	Quantity			
	Fabricator	NSGA-II	SPEA2	SMPSO
Bolted-Bolted Single Angle	303	0	20	48
Welded–Bolted Single Angle	0	69	86	66
Bolted–Bolted Double Angle	188	6	30	150
Welded–Bolted Double Angle	0	149	165	105
Shear Tab	924	1191	1114	1046
Area of PL10 (m^2)	28.14 (6)	33.63 (7)	32.63 (7)	26.18 (5)
Area of PL13 (m^2)	15.17 (3)	0.05 (1)	0.05 (1)	0.41 (1)

Area of PL16 (m^2)	5.97 (2)	0	0	0.96 (1)
Area of PL19 (m^2)	0	0	0	1.25 (1)
Length of L89x64x9.5 (m)	33.94 (3)	93.13 (8)	99.26 (9)	54.94 (5)
Length of L89x89x9.5 (m)	190.03 (16)	2.50 (1)	15.76 (2)	65.82 (6)
Number of Bolts	9571	3692	4099	4984
Total Weight (kg)	302,956	301,465	301,682	302,083
Cost of Shear Connections	$118,301	$111,808	$112,999	$143,562
Material and Steel Fabrication Cost	$866,931	$860,438	$861,629	$892,192
Execution Time	N/A	18m 20s	18m 59s	20m 4s

Additional supplementary tests identical to the ones conducted for the Grand Prairie Honda case study were carried out for this case study as well to further assess the effectiveness of the tool across differing design loads. Figure 5-3 showcases the total material and steel fabrication cost associated with each design load and Figure 5-4 focuses only on the cost associated with the shear connections.

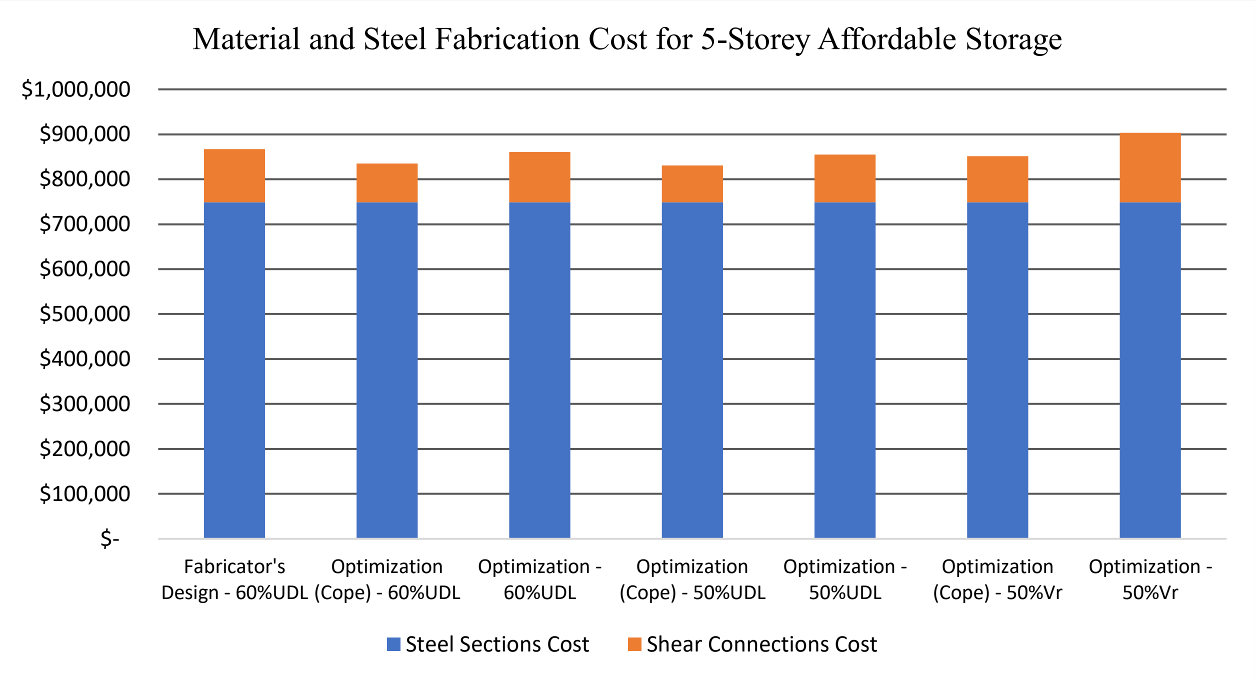


Figure 5-3: Material and steel fabrication cost for 5-Storey Affordable Storage under various design loads using the optimization tool using NSGA-II algorithm

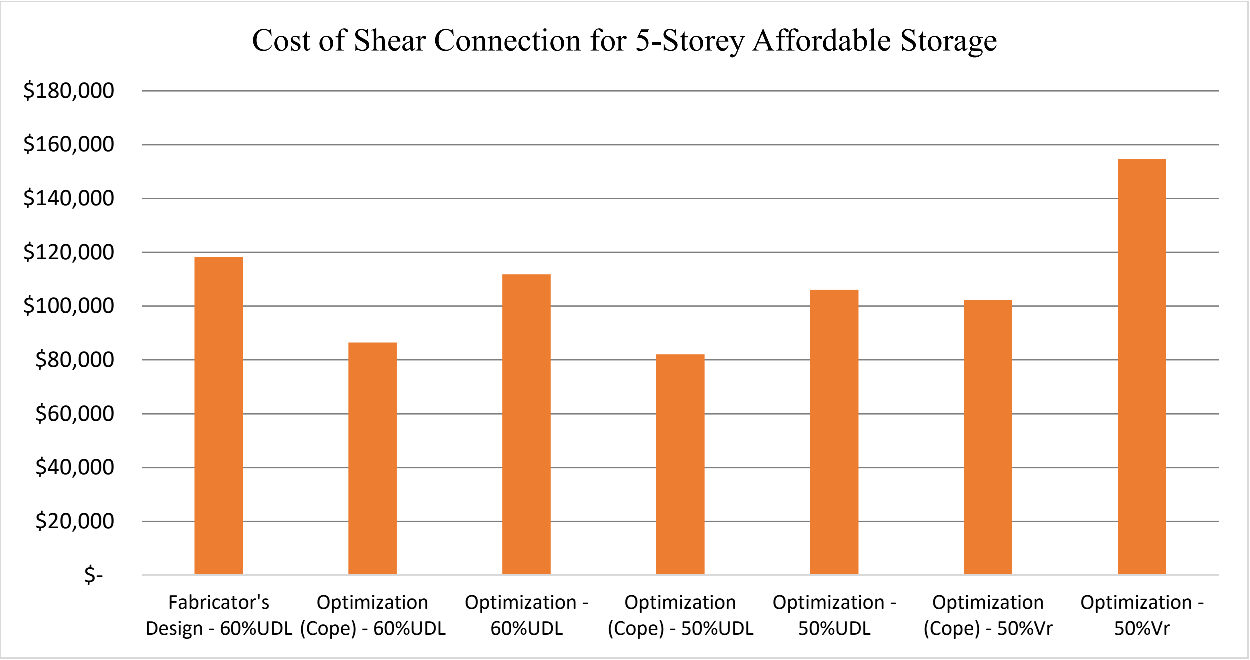


Figure 5-4: Cost of shear connections for 5-Storey Affordable Storage under various design loads using the optimization tool using NSGA-II algorithm

5.4 Summary of Results

In the Grand Prairie Honda case study, the optimization process exhibited consistent execution speed across all three metaheuristics, with an average runtime of approximately 6 minutes and 39 seconds. Similarly, in the 5-Storey Affordable Storage case study, the optimization process also maintained consistent execution speed across all three metaheuristics, with an average runtime of approximately 19 minutes and 8 seconds. It's important to note that the actual runtime may vary based on the computational resources available on the machine executing the process.

In the Grand Prairie Honda case study, the SPEA2 algorithm produced the most cost-effective solution, achieving a 5.9% lower material and fabrication cost of shear connections than the original design. Likewise, the NSGA-II algorithm yielded a cost reduction of 5.4%. In contrast, the SMPSO algorithm resulted in a higher cost of 28.7% compared to the original design.

Similarly, in the 5-Storey Affordable Storage case study, the NSGA-II algorithm emerged as the most effective in reducing material and fabrication costs of shear connections achieving a 5.5% reduction in comparison to the original design. The SPEA2 algorithm also proved beneficial, yielding a cost reduction of 4.5%. On the other hand, the SMPSO algorithm again resulted in a higher cost compared to the original design resulting in an increase of 21.4%.

The comparable performance of the NSGA-II and SPEA2 algorithms can be attributed to their similar genetic algorithm framework in their optimization approach. However, the underperformance of the SMPSO algorithm remains unclear. It is worth noting that different metaheuristics may yield varying results depending on the specific optimization problem at hand, as discussed in relevant literature in Chapter 2. Hence, it is possible that the SMPSO algorithm is not well-suited for addressing this particular problem.

The results of the optimization tool showcase that the NSGA-II and SPEA2 algorithms yield cost-effective solutions that require significantly less material compared to the original designs. This reduction in material usage may be attributed to the connection designer's utilization of similar, albeit suboptimal, designs across various joints to enhance efficiency during the fabrication process. The use of standardized connections can streamline fabrication procedures by facilitating repetition, resulting in cost savings. However, this approach may lead to over-designed connections. Currently, the optimization tool does not incorporate the cost savings associated with repetitive connections, as quantifying them is challenging. Nevertheless, future research should aim to explore this aspect as a potential objective function for optimization.

The optimization tool resulted in minimal cost savings for the overall steel fabrication for both buildings. The most cost-effective solution only reduced the total cost by 0.6% for the Grand Prairie Honda and by 0.7% for the 5-Storey Affordable Storage. This outcome can be attributed to several factors. First, the majority of the cost in steel fabrication is determined by the weight of the steel sections, which is predetermined by the engineer prior to the connection design phase. Consequently, the optimization tool's impact on cost, in its current state, is limited. Furthermore, shear connections represent a relatively small portion of the overall cost compared to other types of connections. Finally, due to the simplicity of shear connections, connection designers often achieve efficient design solutions, leading to a considerable number of connections already being cost-optimal.

Although the optimization tool resulted in minimal cost savings, the primary advantage lies in its speed for designing and detailing connections. The tool has shown the ability to efficiently optimize and detail over 1400 connections in approximately 19 minutes. This significant acceleration in the process stands in clear contrast to the traditional approach, which often demands hundreds of hours from engineers and detailers. Furthermore, the optimization tool generates multiple alternative options that users can directly work with or select from, adding further flexibility and choice to the design process. It also has the potential to reveal solutions or geometries that may not even have been considered by the fabricator *a priori*, especially when the tool is extended to more complex fabrication problems.

Additional supplementary tests were also performed for both case studies that analyzed the results of the optimization tool under different design loads. For both case studies, the test results demonstrate that the cost of shear connections tends to increase with the design load, aligning with expectations. Notably, the Grand Prairie Honda case study revealed an approximate 20% reduction in shear connection costs, while the 5-Storey Affordable Storage case study demonstrated an impressive cost reduction of approximately 50% when utilizing 50% of the plastic moment of the supported member for the design load, as opposed to 50% of its shear resistance. This finding underscores the significant impact that the design load has on the overall cost of connections. Moreover, it is worth highlighting the substantial cost savings that can be achieved in fabrication shops equipped with a beam line capable of automating the coping process. This observation is evident when comparing the cost differences between the preferred and non-preferred coping options under similar design loads.

In summary, the current state of the optimization tool demonstrates its capability to automate and optimize shear connections in building design, resulting in cost reductions compared to traditional design methods. While the cost reductions may not be substantial, the tool's remarkable speed in generating numerous design options, allowing for user adjustments or direct implementation, signifies its potential for significant cost savings.

Moreover, the optimization tool presents opportunities for even greater cost savings by incorporating additional connection types and objective functions beyond cost and weight optimization. Expanding the tool's capabilities in these aspects could unlock further cost-saving potential. Furthermore, the findings emphasize that while optimizing connections alone can yield cost savings, simultaneous optimization of framing configurations and section sizes could lead to even greater cost reductions, necessitating further research in this area.

The case studies also highlight the significant influence of design loads on material and fabrication costs for connections. This prompts the question of whether utilizing actual design loads, as opposed to using generic assumptions, can result in substantially lower costs for fabricators.

It's also important to highlight some key considerations that the case studies did not fully address. First, there was limited evaluation of the cost function, with only the material and fabrication costs of shear connections and the material and handling expenses of steel members being considered. Second, the studies did not incorporate additional strengthening requirements, such as the need for stiffeners. Finally, while the two case studies covered a wide range of joint types that required all five shear connection types, they did not include skewed or sloped joints, making it impossible to assess their impact.

6 CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

Steel fabricators and connection designers in the steel construction industry encounter difficulties in accurately evaluating the efficiency and cost implications of their designs when relying on traditional connection design methods. The complexity of the process, involving numerous parameters and the need for recalculations due to project changes, leads to time-consuming and inefficient workflows. To overcome these challenges, this research proposed the development of an optimization tool that integrates emerging technologies like parametric and generative design. This tool aimed to automate and streamline the connection design workflow, provide instant feedback on cost and constructability, and promote better integration across disciplines. By leveraging this tool, steel fabricators and connection designers can achieve greater efficiency and productivity in their design processes.

The optimization tool, built using Grasshopper3D and C# programming, utilizes user preferences and their corresponding Tekla model to design, optimize, and detail steel connections. The core objective of the tool is to achieve cost and weight minimization for buildings by employing a cost function that quantifies material and fabrication expenses associated with steel connections. The tool generates a wide array of potential design options for the user, providing them with the flexibility to either make direct selections or explore a diverse range of alternatives through the use of metaheuristics and generative design.

To assess the tool's effectiveness against real-world designs, two case studies were conducted on a 2-storey and a 5-storey steel building using the optimization tool. These studies served to evaluate the tool's performance and its ability to meet the requirements and standards of actual design scenarios. The case studies showcased the tool’s potential to generate connection designs that yielded lower costs compared to conventional methods. The tool’s standout advantage lies in its exceptional speed, enabling swift design, optimization, and detailing of connections. It produced one hundred potential design options for two buildings featuring 381 and 1415 shear connections, accomplishing this task in 6 and 19 minutes, respectively. In contrast, completing this task would typically require hundreds of engineering and detailing hours.

While the current optimization tool shows promise, it does have limitations primarily stemming from the time constraints of the research project. Specifically, only five types of shear connections have been implemented and investigated, while other types of connections are not currently compatible. Furthermore, additional strengthening elements like doubler plates or stiffeners are not considered in the tool. The existing cost function only accounts for material and fabrication costs of the five implemented shear connections, necessitating expansion if more connections are added in the future. Additionally, cost implications associated with certain erection constraints are not factored into the tool. Lastly, the current objective functions used for optimization do not fully capture a truly optimized design, as fabrication and erection considerations like the use of repetitive designs are not accounted for in the tool.

6.2 Recommendations for Future Research

With the recent utilization of emerging technologies in the steel construction industry, this research holds significant potential for future research projects to be pursued. Notably, the integration of additional connection design modules, such as moment and bracing connections, along with other forms of connections inherent in typical building structures, could result in enhanced optimization by considering the entirety of the structure. Further exploration is necessary to develop a more sophisticated cost function capable of accurately estimating the material, fabrication, and erection costs of the structure, taking into account real-time data. Additionally, it is essential to investigate alternative objective functions that can more effectively minimize the overall cost of the structure.

The outcomes of the case studies revealed that optimizing connections alone does not have a substantial impact on the overall cost of the structure. Consequently, there is a need for further research to expand the optimization tool's scope, encompassing the optimization of framing layout and steel sections as well. This extension would significantly enhance the tool's effectiveness and efficiency in steel building design. Moreover, incorporating actual design loads instead of relying on generalized statements like using 50% of the supported members shear capacity, would yield additional cost savings and further streamline the design process, fostering increased efficiency in steel building design practices.

REFERENCES

Abdallah, H., Ezzedine, F., Haddad, A., Salami, G., Sanboskani, H., Dabaghi, M., & Hamzeh, F. (2019). Employing Generative Design for Sustainable Construction. *Proceedings of the Creative Construction Conference 2019*, 692–698. https://doi.org/10.3311/CCC2019-095

AISC. (2010). *Specification for structural steel buildings* (ANSI/AISC 360-10). American Institute of Steel Construction.

AISC. (2016). *Specification for structural steel buildings* (ANSI/AISC 360-16). American Institute of Steel Construction.

AISC. (2018). *Structural Steel: An Industry Overview* [White paper]. American Institute of Steel Construction. https://www.aisc.org/globalassets/aisc/publications/white-papers/structural_steel_industry_overview_2018.pdf

AISC. (2019). *Steel Construction Manual* (15th. Edition). American Institute of Steel Construction.

AISC. (2022). *Specification for structural steel buildings* (ANSI/AISC 360-22). American Institute of Steel Construction.

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

Alkhadashi, A., Mohammad, F., Zubayr, R. O., Aoun Klalib, H., & Balik, P. (2022). Multi-objective design optimisation of steel framed structures using three different methods. *International Journal of Structural Integrity*, *13*(1), 92–111. https://doi.org/10.1108/IJSI-07-2021-0080

Aram, S., Eastman, C., & Beetz, J. (2014). Qualitative and Quantitative Cost Estimation: A Methodology Analysis. *Computing in Civil and Building Engineering (2014)*, 381–389. https://doi.org/10.1061/9780784413616.048

Autodesk. (2023). *Revit* (Version 2023) [Computer software]. Autodesk Inc. https://www.autodesk.ca/en

Barg, S. (2020). *Integrated Steel Design Through Automation* [Doctoral dissertation]. Stanford University.

Barg, S., Flager, F., & Fischer, M. (2018). An analytical method to estimate the total installed cost of structural steel building frames during early design. *Journal of Building Engineering*, *15*, 41–50. https://doi.org/10.1016/j.jobe.2017.10.010

Barraza, M., Bojórquez, E., Fernández-González, E., & Reyes-Salazar, A. (2017). Multi-objective optimization of structural steel buildings under earthquake loads using NSGA-II and PSO. *KSCE Journal of Civil Engineering*, *21*(2), 488–500. https://doi.org/10.1007/s12205-017-1488-7

Canadian Industry Program for Energy Conservation, Canadian Steel Producers Association, & Canada (Eds.). (2007). *Benchmarking energy intensity in the Canadian steel industry*. Natural Resources Canada.

Carter, C. J., & Schlafly, T. J. (2008). *$ave More Money*. AISC Modern Steel Construction.

Chicano, F., Sutton, A. M., Whitley, L. D., & Alba, E. (2015). Fitness Probability Distribution of Bit-Flip Mutation. *Evolutionary Computation*, *23*(2), 217–248. https://doi.org/10.1162/EVCO_a_00130

CISC. (2021). *Handbook of steel construction* (12th Edition). Canadian Institute of Steel Construction (CISC).

Collins Steel. (2023). *Copy of FabricationErection Unit Costs—ED.xlsx* [Personal communication].

Computers and Structures Inc. (2022). *CSiBridge* (Version 2022) [Computer software]. Computers and Structures Inc. https://www.csiamerica.com/

CSA. (2019). *Design of steel structures* (CAN/CSA Standard No. S16-19). Canadian Standards Association.

Deb, K. (1999). An introduction to genetic algorithms. *Sādhanā*, *24*, 293–315. https://doi.org/10.1007/BF02823145

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Ltd.

Deb, K. (2011). Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction. *KanGAL Report No. 2011003*.

Deb, K., & Agrawal, R. B. (1995). Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, *9*, 115–148.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

Dorst, K. (2019). Co-evolution and emergence in design. *Design Studies*, *65*, 60–77. https://doi.org/10.1016/j.destud.2019.10.005

Dorst, K., & Cross, N. (2001). Creativity in the design process: Co-evolution of problem–solution. *Design Studies*, *22*(5), 425–437. https://doi.org/10.1016/S0142-694X(01)00009-6

Dubois, A., & Gadde, L.-E. (2002). The construction industry as a loosely coupled system: Implications for productivity and innovation. *Construction Management and Economics*, *20*(7), 621–631. https://doi.org/10.1080/01446190210163543

Duncan, A., Kingi, V., & Brunsdon, N. (2018). *Adopting new ways in the building and construction industry* (Study Report SR406).

Duong, E., Darras, A., Driver, R. G., Essa, M., & Imanpour, A. (2022). Applications of Artificial Intelligence Techniques for Optimization of Structural Steel Connections. In S. Walbridge, M. Nik-Bakht, K. T. W. Ng, M. Shome, M. S. Alam, A. el Damatty, & G. Lovegrove (Eds.), *Proceedings of the Canadian Society of Civil Engineering Annual Conference 2021* (Vol. 244, pp. 273–285). Springer Singapore. https://doi.org/10.1007/978-981-19-0656-5_23

Durillo, J. J., García-Nieto, J., Nebro, A. J., Coello, C. A. C., Luna, F., & Alba, E. (2009). Multi-Objective Particle Swarm Optimizers: An Experimental Comparison. In M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, & M. Sevaux (Eds.), *Evolutionary Multi-Criterion Optimization* (Vol. 5467, pp. 495–509). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-01020-0_39

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760–771. https://doi.org/10.1016/j.advengsoft.2011.05.014

Durillo, J. J., Nebro, A. J., & Alba, E. (2010). The jMetal framework for multi-objective optimization: Design and architecture. *IEEE Congress on Evolutionary Computation*, 1–8. https://doi.org/10.1109/CEC.2010.5586354

Evers, H. G. A., & Maatje, I. F. (2000). *COST BASED ENGINEERING AND PRODUCTION OF STEEL CONSTRUCTIONS. Proceedings of the Steel Design Codes-Fourth International Workshop on Connections in Steel Structures*, 14–22.

García-Segura, T., & Yepes, V. (2016). Multiobjective optimization of post-tensioned concrete box-girder road bridges considering cost, CO2 emissions, and safety. *Engineering Structures*, *125*, 325–336. https://doi.org/10.1016/j.engstruct.2016.07.012

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company Inc.

Gonçalves, M. S., Lopez, R. H., & Miguel, L. F. F. (2015). Search group algorithm: A new metaheuristic method for the optimization of truss structures. *Computers & Structures*, *153*, 165–184. https://doi.org/10.1016/j.compstruc.2015.03.003

Grand View Research. (2018). *Strutural Steel Market Size, Share & Trends Analysis Report By Application (Non-residential (Industrial, Commerical, Offices, Institutional), Residential), By Region And Segment Forecasts, 2019-2025*. https://www.grandviewresearch.com/industry-analysis/structural-steel-market#

Greiner, D., Galván, B., Emperador, J. M., Méndez, M., & Winter, G. (2011). Introducing Reference Point Using g-Dominance in Optimum Design Considering Uncertainties: An Application in Structural Engineering. In R. H. C. Takahashi, K. Deb, E. F. Wanner, & S. Greco (Eds.), *Evolutionary Multi-Criterion Optimization* (Vol. 6576, pp. 389–403). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-19893-9_27

Haapio, J. (2012). *Feature-Based Costing Method for Skeletal Steel Structures based on the Process Approach* [Doctoral dissertation]. Tampere University of Technology.

Håkansson, H., & Ingemansson, M. (2013). Industrial renewal within the construction network. *Construction Management and Economics*, *31*(1), 40–61. https://doi.org/10.1080/01446193.2012.737470

Hamdan, M. M. (2014). Revisiting the distribution index in simulated binary crossover operator for evolutionary multiobjective optimisation algorithms. *2014 Fourth International Conference on Digital Information and Communication Technology and Its Applications (DICTAP)*, 37–41. https://doi.org/10.1109/DICTAP.2014.6821653

Hasançebi, O., Çarbaş, S., Doğan, E., Erdal, F., & Saka, M. P. (2009). Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. *Computers & Structures*, *87*(5–6), 284–302. https://doi.org/10.1016/j.compstruc.2009.01.002

H'mida, F., Martin, P., & Vernadat, F. (2006). Cost estimation in mechanical production: The Cost Entity approach applied to integrated product engineering. *International Journal of Production Economics*, *103*(1), 17–35. https://doi.org/10.1016/j.ijpe.2005.02.016

Hofmeyer, H., & Davila Delgado, J. M. (2015). Coevolutionary and genetic algorithm based building spatial and structural design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *29*(4), 351–370. https://doi.org/10.1017/S0890060415000384

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press.

Hui Wang & Feng Qian. (2008). Improved PSO-based Multi-Objective Optimization using inertia weight and acceleration coefficients dynamic changing, crowding and mutation. *2008 7th*

World Congress on Intelligent Control and Automation, 4479–4484. https://doi.org/10.1109/WCICA.2008.4593644

Jabi, W. (2013). *Parametric design for architecture*. Laurence King Publishing.

Johan, R., Chernyavsky, M., Fabbri, A., Gardner, N., Haeusler, H., & Zavoleas, Y. (2019). *Building Intelligence Through Generative Design—Structural analysis and optimisation informed by material performance*. 371–380. https://doi.org/10.52842/conf.caadria.2019.1.371

Kannimuthu, M., Ekambaram, P., Raphael, B., & Kuppuswamy, A. (2018). Resource Unconstrained and Constrained Project Scheduling Problems and Practices in a Multiproject Environment. *Advances in Civil Engineering*, *2018*, 1–13. https://doi.org/10.1155/2018/9579273

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, *4*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Khanduja, N., & Bhushan, B. (2021). Recent Advances and Application of Metaheuristic Algorithms: A Survey (2014–2020). In H. Malik, A. Iqbal, P. Joshi, S. Agrawal, & F. I. Bakhsh (Eds.), *Metaheuristic and Evolutionary Computation: Algorithms and Applications* (Vol. 916, pp. 207–228). Springer Singapore. https://doi.org/10.1007/978-981-15-7571-6_10

Kicinger, R., Obayashi, S., & Arciszewski, T. (2007). Evolutionary Multiobjective Optimization of Steel Structural Systems in Tall Buildings. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, & T. Murata (Eds.), *Evolutionary Multi-Criterion Optimization* (Vol. 4403, pp. 604–618). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-70928-2_46

Kripakaran, P., Hall, B., & Gupta, A. (2011). A genetic algorithm for design of moment-resisting steel frames. *Structural and Multidisciplinary Optimization*, *44*(4), 559–574. https://doi.org/10.1007/s00158-011-0654-7

Kulak, G. L., Fisher, J. W., Struik, J. H. A., & Fisher, J. W. (1987). *Guide to design criteria for bolted and riveted joints* (2nd ed). John Wiley & Sons, Inc.

Kunakote, T., & Bureerat, S. (2011). Multi-objective topology optimization using evolutionary algorithms. *Engineering Optimization*, *43*(5), 541–557. https://doi.org/10.1080/0305215X.2010.502935

Lawson, B. (2010). *How designers think: The design process demystified* (Reprint). Elsevier Architectural Press.

Lesik, D. F., & Kennedy, D. J. L. (1990). Ultimate strength of fillet welded connections loaded in plane. *Canadian Journal of Civil Engineering*, *17*(1), 55–67. https://doi.org/10.1139/l90-008

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, *18*(3), 300–315. https://doi.org/10.1016/0022-5193(68)90080-5

Loosemore, M. (2014). Improving construction productivity: A subcontractor's perspective. *Engineering, Construction and Architectural Management*, *21*(3), 245–260. https://doi.org/10.1108/ECAM-05-2013-0043

Luke, S. (2013). *Essentials of metaheuristics: A set of undergraduate lecture notes; Online Version 2.0* (2. ed). Lulu.

Madeira, J. A., Rodrigues, H., & Pina, H. (2005). Multi-objective optimization of structures topology by genetic algorithms. *Advances in Engineering Software*, *36*(1), 21–28. https://doi.org/10.1016/j.advengsoft.2003.07.001

Makki, M., Showkatbakhsh, M., & Song, Y. (2022). *Wallacei* [Computer software]. https://www.wallacei.com/

Martínez-Rocamora, A., García-Alvarado, R., Casanova-Medina, E., González-Böhme, L. F., & Auat-Cheein, F. (2020). Parametric Programming of 3D Printed Curved Walls for Cost-Efficient Building Design. *Journal of Construction Engineering and Management*, *146*(5), 04020039. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001811

McKinsey Global Institute. (2017). *Reinventing Construction: A Route To Higher Productivity*. McKinsey & Company.

McNeel, R., & Associates. (2023a). *Grasshopper 3D* (Version 7) [Computer software]. Robert McNeel & Associates.

McNeel, R., & Associates. (2023b). *Rhinocerous 3D* (Version 7) [Computer software]. Robert McNeel & Associates.

Mei, L., & Wang, Q. (2021). Structural Optimization in Civil Engineering: A Literature Review. *Buildings*, *11*(2), 66. https://doi.org/10.3390/buildings11020066

Microsoft. (2023). *C# Programming Language* [C#]. Microsoft.

Miettinen, K. (1998). *Nonlinear Multiobjective Optimization* (Vol. 12). Springer US. https://doi.org/10.1007/978-1-4615-5563-6

Miller, B. L., & Goldberg, D. E. (1995). Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Systems*, *9*(3), 193–212.

Muir, L. S., & Hewitt, C. M. (2009). Design of Unstiffened Extended Single-Plate Shear Connections. *Engineering Journal, American Institute of Steel Construction*, *46*, 67–80.

Muir, L. S., & Thornton, W. A. (2011). The Development of a New Design Procedure for Conventional Single-Plate Shear Connections. *Engineering Journal, American Institute of Steel Construction*, *48*, 141–152.

Muñoz-La Rivera, F., Mora-Serrano, J., Valero, I., & Oñate, E. (2021). Methodological-Technological Framework for Construction 4.0. *Archives of Computational Methods in Engineering*, *28*(2), 689–711. https://doi.org/10.1007/s11831-020-09455-9

Nebro, A. J., Durillo, J. J., Garcia-Nieto, J., Coello Coello, C. A., Luna, F., & Alba, E. (2009). SMPSO: A new PSO-based metaheuristic for multi-objective optimization. *2009 IEEE Symposium on Computational Intelligence in Milti-Criteria Decision-Making*, 66–73. https://doi.org/10.1109/MCDM.2009.4938830

Neumann, J. V. (1951). The General and Logical Theory of Automata. In *Cerebral Mechanisms in Behavior: The Hixon Symposium* (pp. 1–41). John Wiley & Sons, Inc.

Neves, M., Basaglia, C., & Camotim, D. (2022). Stiffening optimisation of conventional cold-formed steel cross-sections based on a multi-objective Genetic Algorithm and using Generalised Beam Theory. *Thin-Walled Structures*, *179*, 109713. https://doi.org/10.1016/j.tws.2022.109713

Noilublao, C., & Bureerat, S. (2009). Simultaneous Topology, Shape and Sizing Optimisation of Skeletal Structures Using Multiobjective Evolutionary Algorithms. In W. P. Dos Santos (Ed.), *Evolutionary Computation*. InTech. https://doi.org/10.5772/9613

Oosterhof, S. A. (2014). *Behaviour of Steel Shear Connections for Assessing Structural Vulnerability to Disproportionate Collapse* [Doctoral dissertation]. University of Alberta.

Oxman, R. (2006). Theory and design in the first digital age. *Design Studies*, *27*(3), 229–265. https://doi.org/10.1016/j.destud.2005.11.002

Pavlovčič, L., Krajnc, A., & Beg, D. (2004). Cost function analysis in the structural optimization of steel frames. *Structural and Multidisciplinary Optimization*, *28*(4), 286–295. https://doi.org/10.1007/s00158-004-0430-z

Purshouse, R. C., Deb, K., Mansor, M. M., Mostaghim, S., & Wang, R. (2014). A review of hybrid evolutionary multiple criteria decision making methods. *2014 IEEE Congress on Evolutionary Computation (CEC)*, 1147–1154. https://doi.org/10.1109/CEC.2014.6900368

Rahman, M. M. (2014). Barriers of Implementing Modern Methods of Construction. *Journal of Management in Engineering*, *30*(1), 69–77. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000173

Rodrigues, E., Amaral, A. R., Gaspar, A. R., & Gomes, Á. (2015). An Approach to Urban Quarter Design Using Building Generative Design and Thermal Performance Optimization. *Energy Procedia*, *78*, 2899–2904. https://doi.org/10.1016/j.egypro.2015.11.662

Rodrigues, E., Gaspar, A. R., & Gomes, Á. (2013a). An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology. *Computer-Aided Design*, *45*(5), 887–897. https://doi.org/10.1016/j.cad.2013.01.001

Rodrigues, E., Gaspar, A. R., & Gomes, Á. (2013b). An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 2: Validation and performance tests. *Computer-Aided Design*, *45*(5), 898–910. https://doi.org/10.1016/j.cad.2013.01.003

Ruby, D. I. (2008). *Steel Design Guide 23: Constructability of Structural Steel Buildings*. American Institute of Steel Construction.

Rutten, D. (2019). *Galapagos* [Computer software].

Saka, M. P., & Geem, Z. W. (2013). Mathematical and Metaheuristic Applications in Design Optimization of Steel Frame Structures: An Extensive Review. *Mathematical Problems in Engineering*, *2013*, 1–33. https://doi.org/10.1155/2013/271031

Saka, M. P., Hasançebi, O., & Geem, Z. W. (2016). Metaheuristics in structural optimization and discussions on harmony search algorithm. *Swarm and Evolutionary Computation*, *28*, 88–97. https://doi.org/10.1016/j.swevo.2016.01.005

Salehi, H., & Burgueño, R. (2018). Emerging artificial intelligence methods in structural engineering. *Engineering Structures*, *171*, 170–189. https://doi.org/10.1016/j.engstruct.2018.05.084

Sanchis, J., Martinez, M., & Blasco, X. (2008). Multi-objective engineering design using preferences. *Engineering Optimization*, *40*(3), 253–269. https://doi.org/10.1080/03052150701693057

Schon, D. A. (1992). Designing as reflective conversation with the materials of a design situation. *Knowledge-Based Systems*, *5*(1), 3–14. https://doi.org/10.1016/0950-7051(92)90020-G

Sepasgozar, S., & Davis, S. (2018). Construction Technology Adoption Cube: An Investigation on Process, Factors, Barriers, Drivers and Decision Makers Using NVivo and AHP Analysis. *Buildings*, *8*(6), 74. https://doi.org/10.3390/buildings8060074

Sharma, D., Deb, K., & Kishore, N. N. (2011). Domain-specific initial population strategy for compliant mechanisms using customized genetic algorithm. *Structural and*

Multidisciplinary Optimization, *43*(4), 541–554. https://doi.org/10.1007/s00158-010-0575-x

Shen, L., & Lin, Y. X. (2014). Strategies in Using Building Information Modeling (BIM) to Solve Problems in Project Management of Chinese Construction Enterprises. *Applied Mechanics and Materials*, *501–504*, 2700–2705. https://doi.org/10.4028/www.scientific.net/AMM.501-504.2700

Silverman, B. W. (1986). Density Estimation for Statistics and Data Analysis. *Chapman & Hall*.

Singh, P., & Choudhary, S. K. (2021). Introduction: Optimization and Metaheuristics Algorithms. In H. Malik, A. Iqbal, P. Joshi, S. Agrawal, & F. I. Bakhsh (Eds.), *Metaheuristic and Evolutionary Computation: Algorithms and Applications* (Vol. 916, pp. 3–33). Springer Singapore. https://doi.org/10.1007/978-981-15-7571-6_1

Singh, V., & Gu, N. (2012). Towards an integrated generative design framework. *Design Studies*, *33*(2), 185–207. https://doi.org/10.1016/j.destud.2011.06.001

Sivanandam, S. N., & Deepa, S. N. (2007). *Introduction to genetic algorithms*. Springer.

Skitmore, M., & Ashworth, A. (1983). *The Effectiveness of Estimating in the Construction Industry*. The Chartered Institute of Building, Englemere, Berkshire, England.

Stanković, T., Štorga, M., & Marjanović, D. (2012). Synthesis of Truss Structure Designs by NSGA-II and NodeSort Algorithm. *Strojniški Vestnik – Journal of Mechanical Engineering*, *58*(3), 203–212. https://doi.org/10.5545/sv-jme.2011.042

Stankovic, T., Ziha, K., & Marjanovic, D. (2011). Tracing Engineering Evolution with Evolutionary Algorithms. In E. Kita (Ed.), *Evolutionary Algorithms*. InTech. https://doi.org/10.5772/15934

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, *7*(3), 343–351. https://doi.org/10.1068/b070343

Tang, X., Bassir, D. H., & Zhang, W. (2011). Shape, sizing optimization and material selection based on mixed variables and genetic algorithm. *Optimization and Engineering*, *12*(1–2), 111–128. https://doi.org/10.1007/s11081-010-9125-z

Tekla. (2022). *Tekla Structures* (Version 2022) [Computer software]. Tekla.

The MathWorks Inc. (2022). *MATLAB* (9.13.0 (R2022b)) [MATLAB]. The MathWorks Inc. https://www.mathworks.com

Tian, Y., Wang, H., Zhang, X., & Jin, Y. (2017). Effectiveness and efficiency of non-dominated sorting for evolutionary multi- and many-objective optimization. *Complex & Intelligent Systems*, *3*(4), 247–263. https://doi.org/10.1007/s40747-017-0057-5

Touloupaki, E., & Theodosiou, T. (2017). Energy Performance Optimization as a Generative Design Tool for Nearly Zero Energy Buildings. *Procedia Engineering*, *180*, 1178–1185. https://doi.org/10.1016/j.proeng.2017.04.278

Voß, S. (2001). Meta-heuristics: The State of the Art. In A. Nareyek (Ed.), *Local Search for Planning and Scheduling* (Vol. 2148, pp. 1–23). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45612-0_1

Winslow, P., Pellegrino, S., & Sharma, S. B. (2010). Multi-objective optimization of free-form grid structures. *Structural and Multidisciplinary Optimization*, *40*(1–6), 257–269. https://doi.org/10.1007/s00158-009-0358-4

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media.

Yang, J., Qu, L., Shen, Y., Shi, Y., Cheng, S., Zhao, J., & Shen, X. (2020). Swarm Intelligence in Data Science: Applications, Opportunities and Challenges. In Y. Tan, Y. Shi, & M. Tuba

(Eds.), *Advances in Swarm Intelligence* (Vol. 12145, pp. 3–14). Springer International Publishing. https://doi.org/10.1007/978-3-030-53956-6_1

Yang, X. (2010). *Engineering Optimization: An Introduction with Metaheuristic Applications* (1st ed.). Wiley. https://doi.org/10.1002/9780470640425

Yang, X.-S. (2010). *Nature-inspired metaheuristic algorithms* (2. ed). Luniver Press.

Yap, J. B. H., Chow, I. N., & Shavarebi, K. (2019). Criticality of Construction Industry Problems in Developing Countries: Analyzing Malaysian Projects. *Journal of Management in Engineering*, *35*(5), 04019020. https://doi.org/10.1061/(ASCE)ME.1943-5479.0000709

Yeganeh, A. A., Azizi, M., & Falsafi, R. (2019). Root Causes of Design-Construction Interface Problems in Iranian Design-Build Projects. *Journal of Construction Engineering and Management*, *145*(12), 05019014. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001727

Zavala, G. R., Nebro, A. J., Luna, F., & Coello Coello, C. A. (2014). A survey of multi-objective metaheuristics applied to structural optimization. *Structural and Multidisciplinary Optimization*, *49*(4), 537–558. https://doi.org/10.1007/s00158-013-0996-4

Zidane, Y. J.-T., & Andersen, B. (2018). The top 10 universal delay factors in construction projects. *International Journal of Managing Projects in Business*, *11*(3), 650–672. https://doi.org/10.1108/IJMPB-05-2017-0052

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *TIK Report*, *103*. https://doi.org/10.3929/ethz-a-004284029

Appendix A – Isometric views of Grand Prairie Honda Structure

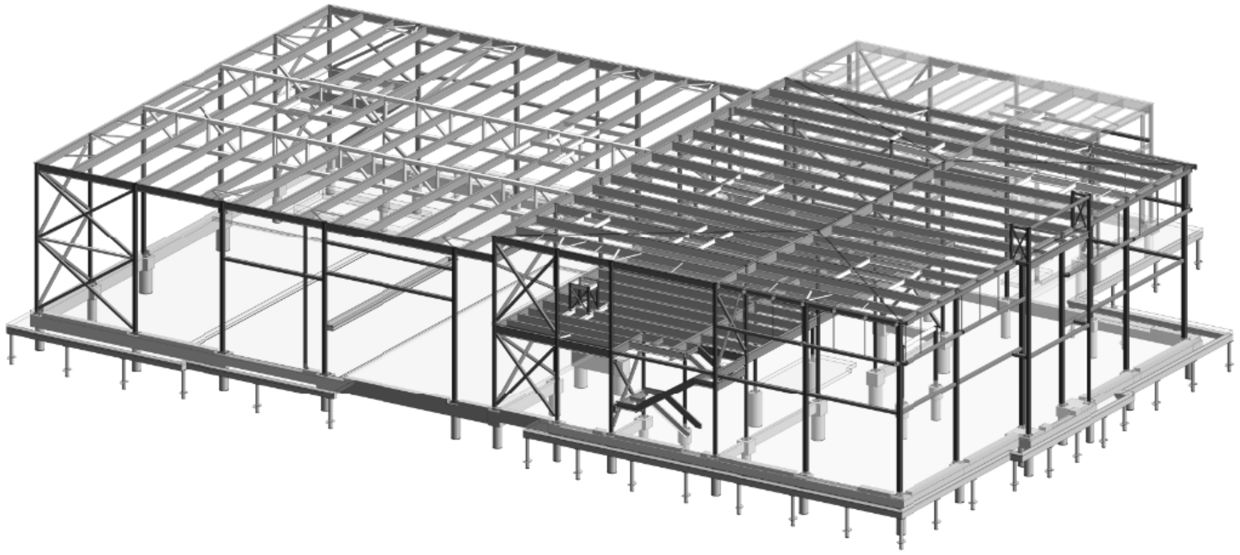


Figure A-1: SE isometric view of Grand Prairie Honda structure

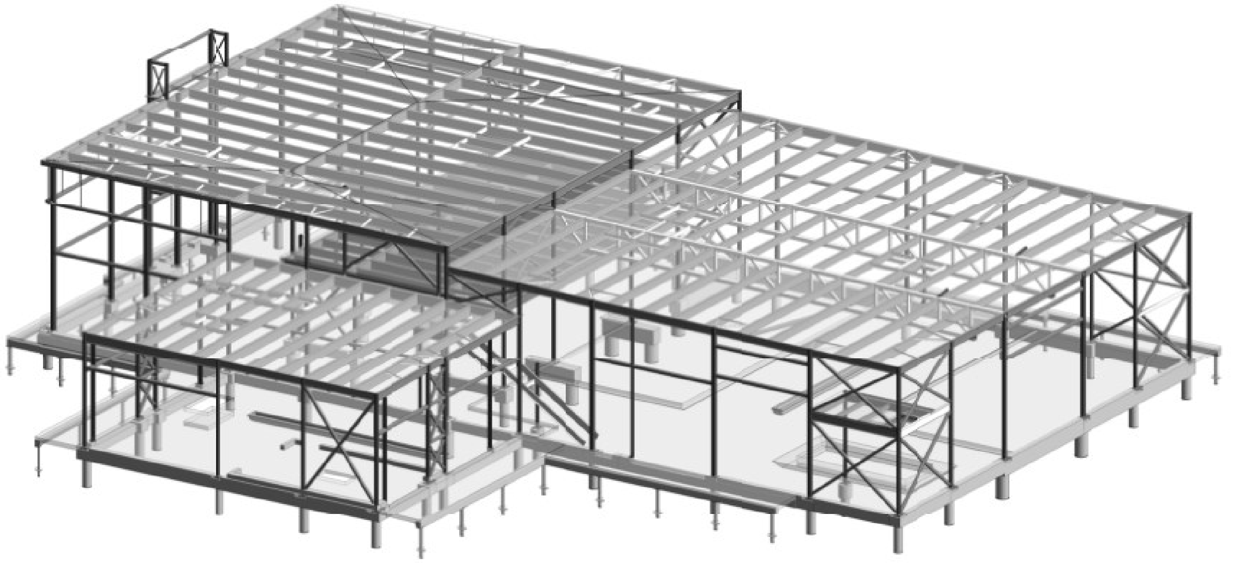


Figure A-2: NW isometric view of Grand Prairie Honda structure

Appendix B – Isometric views of 5-Storey Affordable Storage Structure

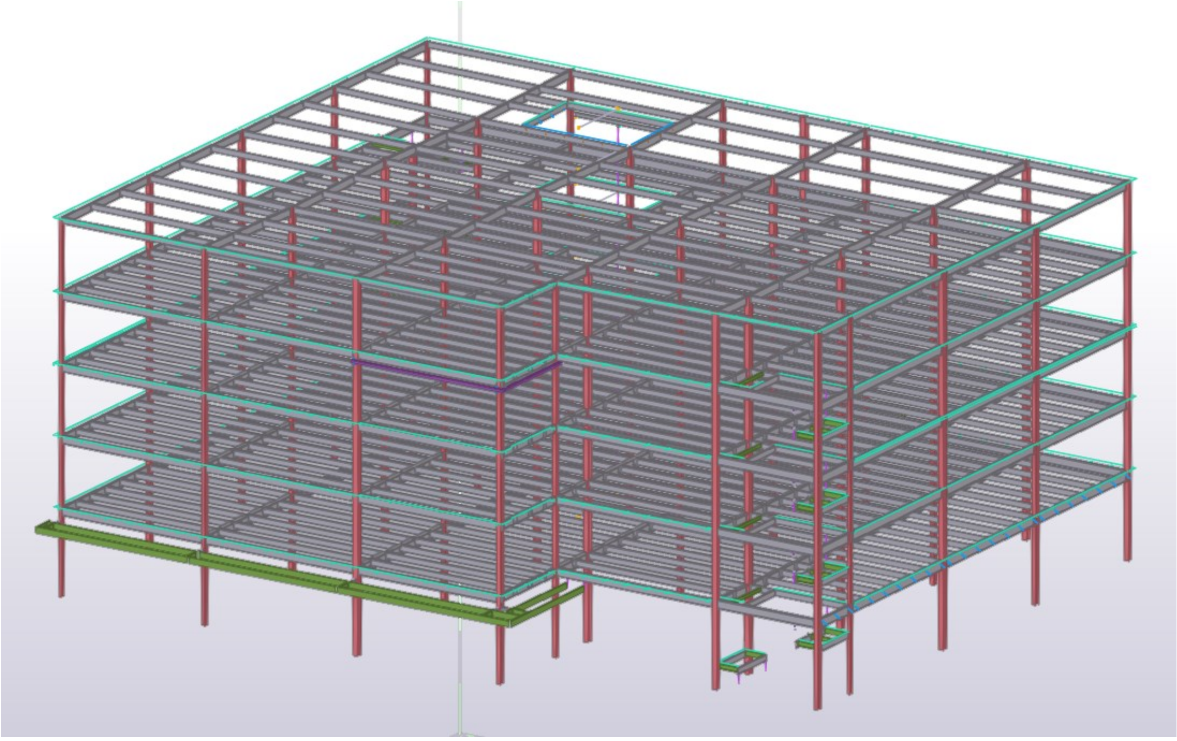


Figure B-1: SW isometric view of 5-Storey Affordable Storage structure

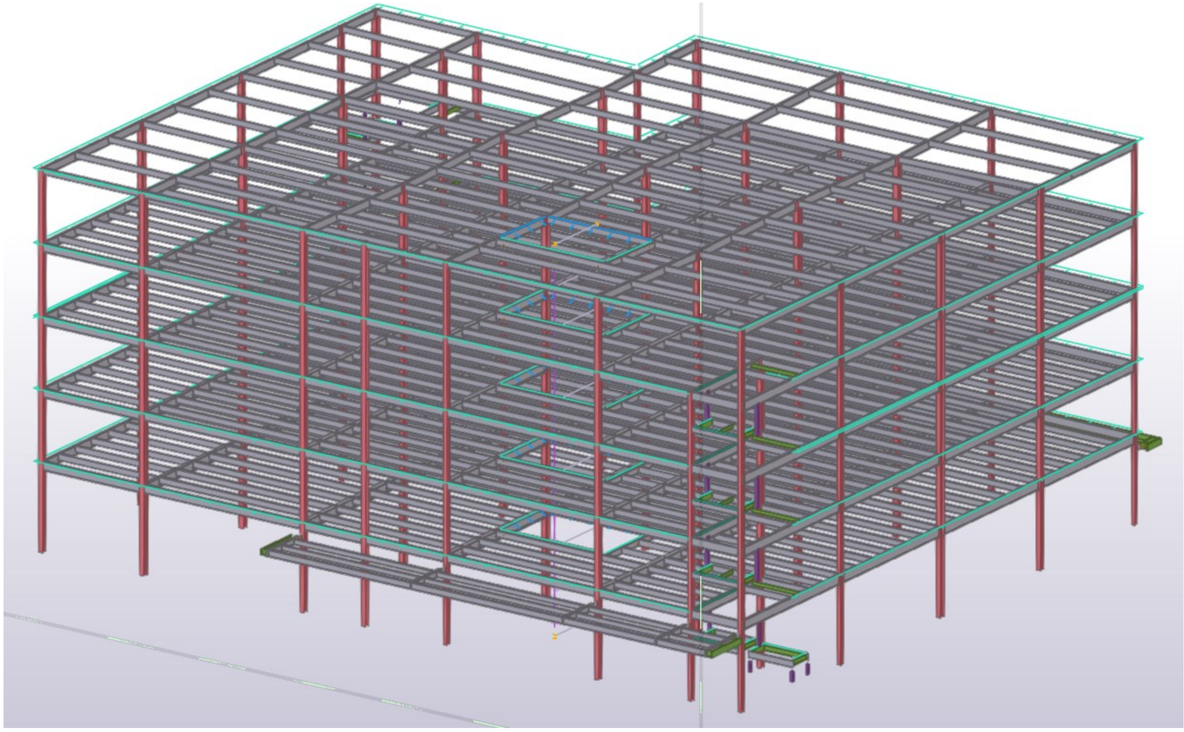


Figure B-2: NE isometric view of 5-Storey Affordable Storage structure

Appendix C – Connection Optimization Tool User Manual

CONNECTION OPTIMIZATION TOOL USER MANUAL

Eric Duong (Steel Centre)

Eric Duong
eduong1@ualberta.ca

Introduction

The intent of this document is to outline the methodology behind the C# code and Grasshopper script that I have created for my MSc research. The document serves to explain the Classes, Structures, and Methods that are in the C# code beyond what is captured in the comments. Reviewing this document should be done in conjunction with reading the code if further clarifications are required.

Additionally, there is a brief section in the beginning outlining how to install and use the optimization tool.

Any questions or inquiries about the optimization tool can be directed to my email eduong1@ualberta.ca.

Connection Optimization Tool

Installation

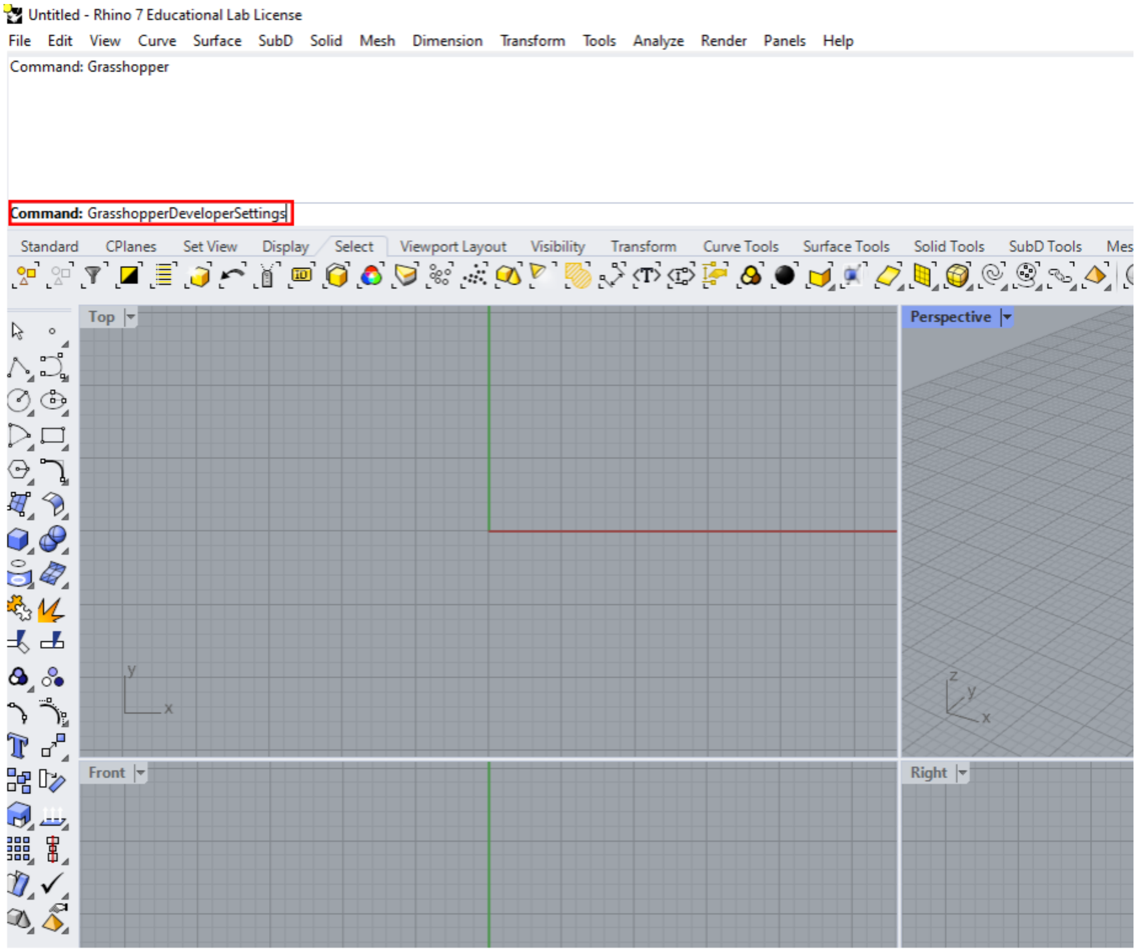
The optimization tool is built around a Grasshopper script. The current script is named ConnectionOptimization_ED.gh (as of 04/04/2023). The script can be downloaded at Food4Rhino at this link: https://www.food4rhino.com/en/app/steel-centre-0. The files are not public available, so the user will have to "Log in with Rhino Accounts" using the Steel Centre Emerging Technology's Google account which is as follows:

Email: SteelCentreEmergTech@gmail.com
Password: Buildwithsteel123!

To run the optimization tool in Grasshopper, the Grasshopper-Tekla Live Link is required. This can be downloaded in the following link: https://warehouse.tekla.com/#/catalog/details/b901f77d-cfe8-4a97-894b-f4053829c297. If the user does not have access to Tekla Warehouse (i.e., is a student), they can download it here: https://warehouse.tekla.com/#/catalog/details/74297db8-8ff1-4442-804a-baa82c86e1eb?showVersions=true&sortBy=-modifiedAt.

In addition to Grasshopper-Tekla Live Link, the "SteelCentre" Library needs to also be added into your Grasshopper. This can be downloaded at Food4Rhino at this link: https://www.food4rhino.com/en/app/steel-centre-0. Once the file is downloaded it will contain a folder named "ConnectionOptimization" that will need to be pathed to in the Grasshopper developer settings in Rhino3D. The instructions below outline how to do this (in addition to a few photos):

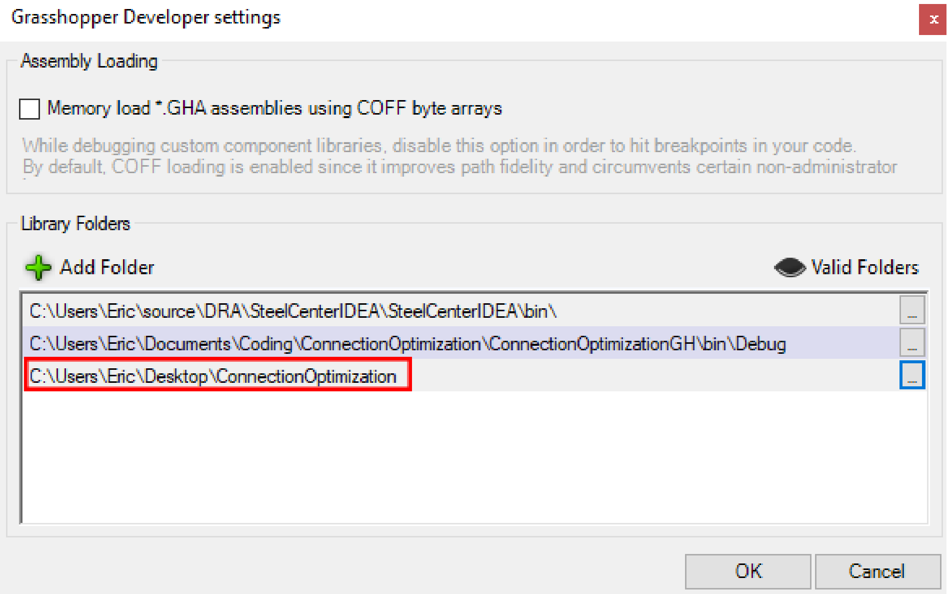
1. Launch Rhino3D. Type GrasshopperDeveloperSettings into the Command box to open up the developer settings.
2. In the developer settings, click "Add Folder" and select the ConnectionOptimization folder that you would have downloaded from food4Rhino.
3. Restart Rhino3D and Grasshopper. Once it restarts, the SteelCentre tab should be added into your Grasshopper.



Untitled - Rhino 7 Educational Lab License

File Edit View Curve Surface SubD Solid Mesh Dimension Transform Tools Analyze Render Panels Help
Standard CPlanes Set View Display Select Viewport Layout Visibility Transform Curve Tools Surface Tools Solid Tools SubD Tools Mes
Top
Perspective
Front
Right

```
Command: Grasshopper
Command: GrasshopperDeveloperSettings
```



Grasshopper Developer settings

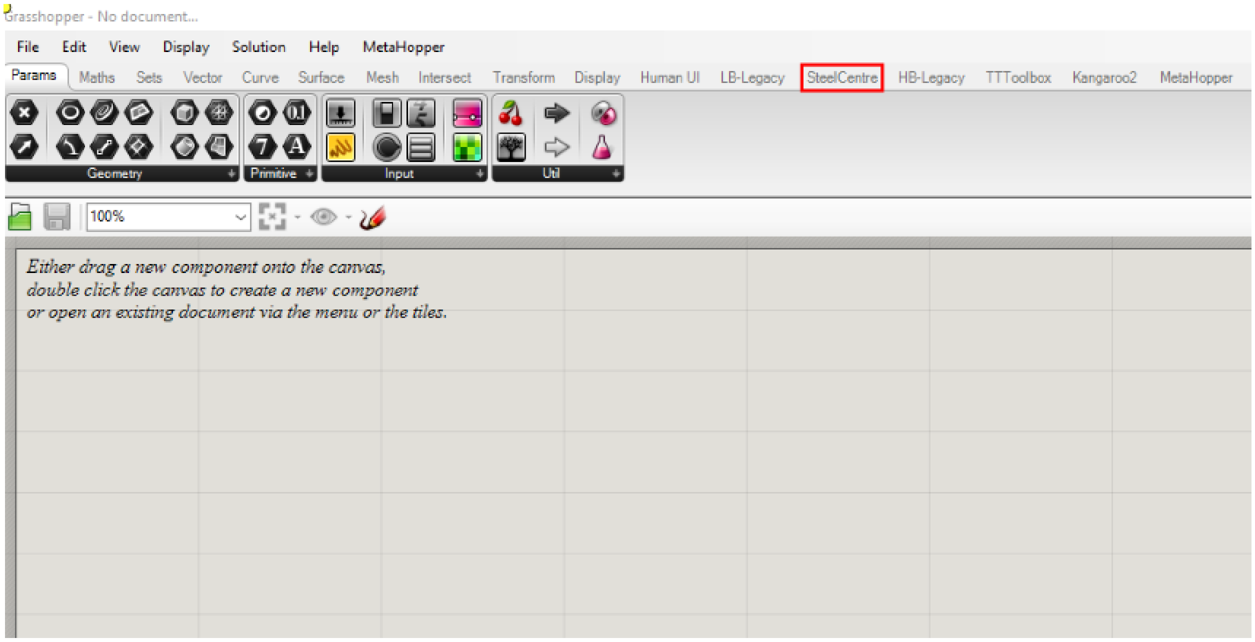
Assembly Loading

☐ Memory load *.GHA assemblies using COFF byte arrays

While debugging custom component libraries, disable this option in order to hit breakpoints in your code.
By default, COFF loading is enabled since it improves path fidelity and circumvents certain non-administrator

Library Folders

C:\Users\Eric\source\DRA\SteelCenterIDEA\SteelCenterIDEA\bin\
C:\Users\Eric\Documents\Coding\ConnectionOptimization\ConnectionOptimizationGH\bin\Debug
C:\Users\Eric\Desktop\ConnectionOptimization



Either drag a new component onto the canvas,
double click the canvas to create a new component
or open an existing document via the menu or the tiles.

Future Development

The code for the project can be found here: https://github.com/EricDuongED/ConnectionOptimization. Any future development can be done based on the source code. If the user has not created custom components in Grasshopper in the past, I recommend these tutorials to get started:

https://developer.rhino3d.com/guides/grasshopper/your-first-component-windows/

https://developer.rhino3d.com/guides/grasshopper/simple-component/

https://developer.rhino3d.com/guides/grasshopper/what-is-a-grasshopper-component/

How to Use

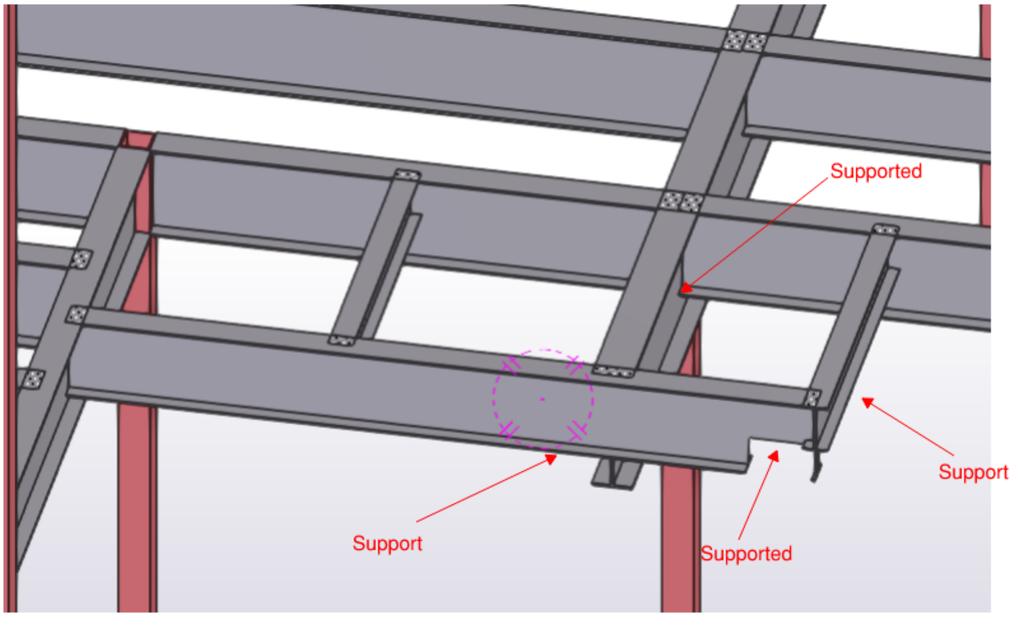
Once the necessary libraries are added to Grasshopper, the tool is ready for use. Before launching Rhino/Grasshopper, the user must first have their Tekla Model open. This is important because the Grasshopper-Tekla Live Link attaches to an open Tekla Structures 2022 application as it launches and cannot attach to a newly opened model. It should be noted that the current optimization tool only works for Tekla Structures 2022. This can be changed in the future (see C# Solution - ConnectionOptimization).

Before using the tool, the Tekla Model must not have any connections modelled. Additionally, the user should create a few different views to properly run the optimization tool.

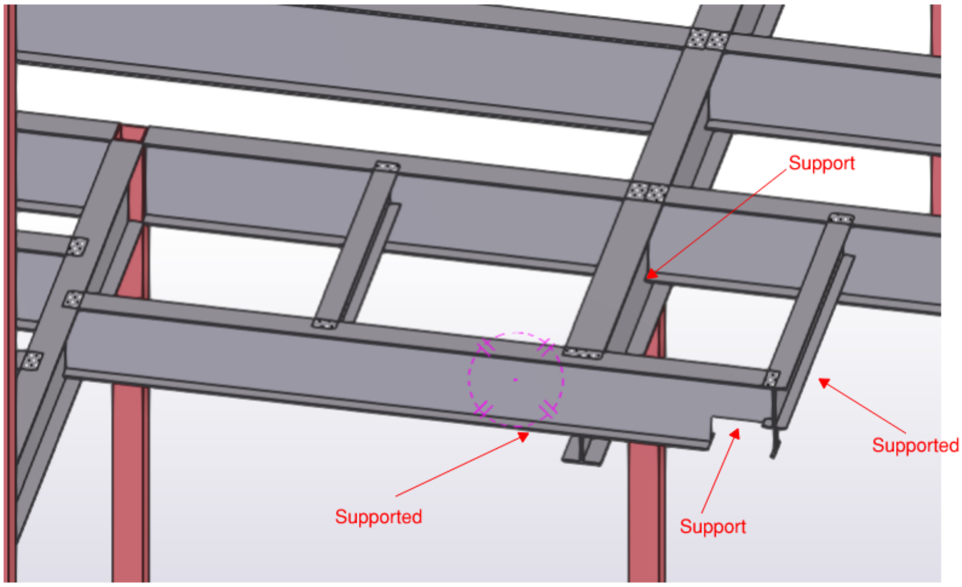
- View 1: Standard view with all the miscellaneous steel.
- View 2: A view with all the miscellaneous steel removed.
- View 3. A view with all the miscellaneous steel removed as well as any members or parts that are not beams, columns, or brace members.

These views are important when extracting the proper volume of material, number of parts, and the shear connections that need to be designed during the optimization process. This is briefly showcased in the video below.

Before running the optimization tool, the user should specify which connections are moment connections by changing the corresponding supported member's UDA. See the RemovedMomentConnection method below for detailed steps on how to do so. For any non-conventional framing, that is, a framing configuration where the tool might have issues figuring out what's the support and supported member. For example, in the figure below, the tool will assume the following:



However, the actual framing should be represented like the following:



This issue arises from the fact that the tool only has the geometry of the Tekla model to determine these framing configurations. Without the structural drawings or structural analysis of the structure, it's basically impossible to tell in certain scenarios. Because of this, I have offloaded this responsibility to the designer who will have access to this information to input the correct framing before running the tool. To do this, the user must adjust the member's UDAs. Detailed information on how to do this is shown below in the DeterminePrimaryMemberUDA and DetermineSupportMember methods.

Once this is complete, the user shall be able to input their preferences and select the members in the model to run the optimization tool. The video below showcases how to run the tool.

https://youtu.be/vvKCZgaeAqI

C# Solution – ConnectionOptimization

This C# solution contains all the code necessary to run the Grasshopper script to utilize the connection optimization tool. The NuGet packages that are used in the solution are as follows:

- **Grasshopper** *by Robert McNeel and Associates*
- **Log4net** *by The Apache Software Foundation*
- **Microsoft.Office.Interop.Excel** *by Microsoft*
- **Newtonsoft.Json** *by James Newton-King*
- **RhinoCommon** *by Robert McNeel and Associates*
- **Tekla.Structures** (2022.0.10715) *by Trimble*
- **Tekla.Structures.Catalogs** (2022.0.10715) *by Trimble*
- **Tekla.Structures.Datatype** (2022.0.10715) *by Trimble*
- **Tekla.Structures.Model** (2022.0.10715) *by Trimble*
- **Tekla.Structures.Plugins** (2022.0.10715) *by Trimble*

The Tekla NuGet packages are configured for Tekla Structures 2022. The optimization tool compiled by this code will **ONLY** work for Tekla Structures 2022. In the future, if the user wishes to use other versions of Tekla Structures, the NuGet Packages must be updated to the corresponding Tekla Structures version and the library will have to be reloaded into Grasshopper. This issue arose when I updated the tool from Tekla Structures 2021 to 2022. The process is straight forward as it only required updating the NuGet Packages; however, if any methods used in the TeklaOpen API Documentation is changed in future versions, then the user must update the affected code.

Ensure that the Grasshopper-Tekla Live Link version matches the Tekla Structures version that the user wishes to use.

ConnectionOptimization

This C# project is primarily used for testing through a console application as opposed to using Grasshopper. The only file of importance is Program.cs which is where the test code is written and tested.

ConnectionOptimmizationGH

This C# project contains the majority of the code used in the optimization code. It contains the code required to install the "SteelCentre" library into Grasshopper.

0. Classes

This folder contains all the classes that are used throughout the project.

Angle.cs

Angle(*string* name) – An Angle object can be created by passing the proper name (CISC handbook). The properties of the angle are determined from a JSON file containing all the properties of the angles. This JSON file was created based off the Sections Database.xlsx file. In the future, if any other sections are to be added, the spreadsheet should be updated and subsequently the JSON file.

There are additional properties for the Angle object that pertain to the angle connection. These properties are not populated until the connections are actually designed and is referenced later in this document.

An additional AngleShape class is defined in this .cs file which is used to pull information from the previously mentioned JSON file.

Bolt.cs

Bolt(*string* name, *string* grade) – A Bolt object can be created by passing the name of the bolt and its corresponding bolt grade. Based on the bolt size, the metric diameter, minimum edge distances (S16-19), area, and required hole diameter will be calculated and stored as properties in the object. Based on the bolt grade, the ultimate and shear strengths are calculated.

ConnectionData.cs

There are a few classes that are stored in this file. All of which include the classes that store the user preferences and cost functions required to determine the cost of a connection.

ConnectionData – Contains the values that are used to determine the total weight and cost of a connection.

UserPreferences – Contains the user preferences such as the grade of steel, bolt size, bolt grade, etc. that the user specifies for the optimization tool. The information will be stored as a List<string> and the object will be created through this list.

CostData – Contains the cost for steel, shop welding, bolts, and shop labour. The CostData object is created based off the user preferences and is where the cost function is determined. The current values for the cost function are determined based off the Fabrication&Erection Unit Costs.xlsx.

DBB_Data_GH.cs

This class is used to create an object that designs and optimizes a double angle bolted-bolted (DBB) connection. The file is heavily commented therefore not much description is required in this document.

DWB_Data_GH.cs

This class is used to create an object that designs and optimizes a double angle welded-bolted (DWB) connection. The file is heavily commented therefore not much description is required in this document.

Functions.cs

This class is used to determine the strength of bolt group or weld group using the Instantaneous Centre of Rotation (IC) method. There are two classes that are present in this file, Bolt and Weld which contains properties that store values necessary to perform the IC method. This was one of the first things I coded in the project which is why it exists in its own file, Functions.cs file, as opposed to Methods.cs. I also could not be bothered to combine them a year later.

Material.cs

This class is used to store the yield and ultimate stress of the steel based on the grade given by the user.

Methods.cs

Arguably the most important class that I have in the entire project. It contains all the methods that is used to develop the entire optimization tool.

Clone<T>(*T* source) – Used to perform a deep copy of an object via serialization. It's used to ensure that the pass-by-reference issue is resolved for certain instances during the optimization process. In the future, if additional classes are created, they must be serializable for this method to work.

CalculateCenterPoint(*Point* point1, *Point* point2) – Returns the point that lies directly in the middle of two given points.

CheckForBearingConnection(*Beam* col, *Beam* beam) – Returns TRUE if the beam and the column forms a bearing connection or if the column is a hanging column and FALSE if not. First it checks to see if the col is actually a Column based on the Tekla Model. If the angle between the beam and the column (based on its vectors) is less than 85° it assumes it is not a bearing connection. It then determines the plane of the beam that extends in the direction of the flanges and calculates the point at which the column and the beam intersects. Afterwards, the distance between the intersection point and end point of the column that is closest to the beam is calculated. If the ratio of this distance and the depth of the beam is greater than 0.75 (arbitrarily assigned ratio), it assumes that the column is a bearing/hanging column and returns TRUE. However, if this distance is greater than the depth of the beam, it cannot be a bearing/hanging column and the method returns FALSE. For example, if a column is modelled such that a beam sits on the top end of a column, the ratio would be equal to 1 thus will be a bearing connection. However, if the beam connects to the mid-height of a column, then the ratio would far exceed 1 thus cannot be a bearing connection.

CheckForBraceConnection(*Beam* mainMem, *Beam* conMem) – Returns TRUE if the two members make up a braced frame and FALSE if not. Assumes that it's a brace connection if at least one of the two members is either an HSS or angle AND if the angle is between 10° and 80° between the two members. Note that this isn't a foolproof method to determine brace connections because the braces could be other structural shapes as well.

CheckForEndPointConnection(*Beam* mem1, *Beam* mem2) – Returns TRUE if mem1 connects at an end point of mem2. Returns FALSE if mem1 connects at least a member depth (mem1) away from an end point of mem2.

CheckForMomentConnection(*Beam* secBeam, *Beam* primBeam) – Returns TRUE if the connection formed by the two members is a moment connection. Returns FALSE otherwise. The user has to specify "YES" in the UDA "START_MOMENT_CONN" or "END_MOMENT_CONN" in order for this method to work.

CheckForSpliceConnection(*Beam* mainMem, *Beam* conMem) – Returns TRUE if the two members make up a spliced connection and FALSE if not. Determine the GeometricPlane of both members if they are parallel then the method returns TRUE.

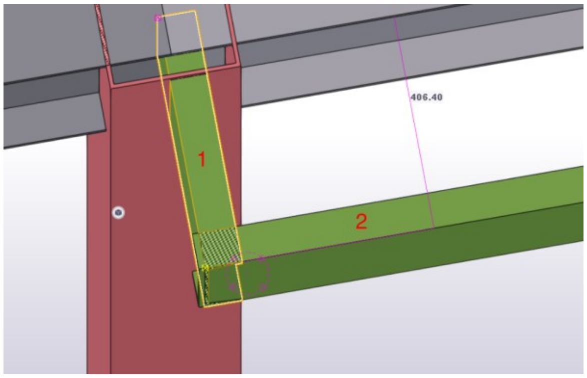
CheckParallelMembers(*Beam* beam1, *Beam* beam2, *Model* model) – Returns TRUE if the two beams that are passed are parallel to each other. Returns FALSE if not.

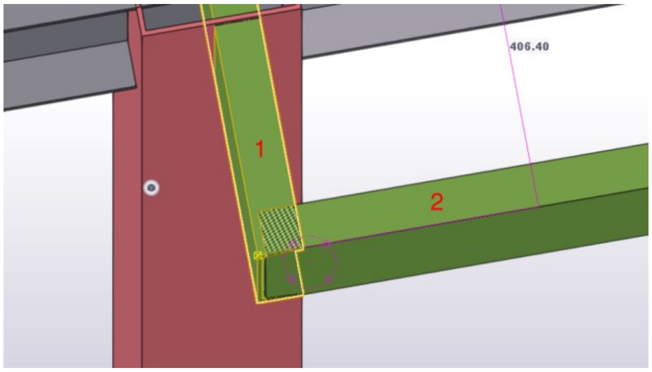
CreateAxisAlignedBoundingBox(*Beam* beam) – Creates an axis-aligned bounding box of a Tekla Beam. This method isn't used as oriented bounding boxes are much better at determining collisions in model space. See next method.

CreateOrientedBoundingBox(*Beam* beam, *Model* model) – Creates an oriented bounding box of a Tekla Beam. This method creates a rectangular box around the member and is used to determine which members collide with each other in model space.

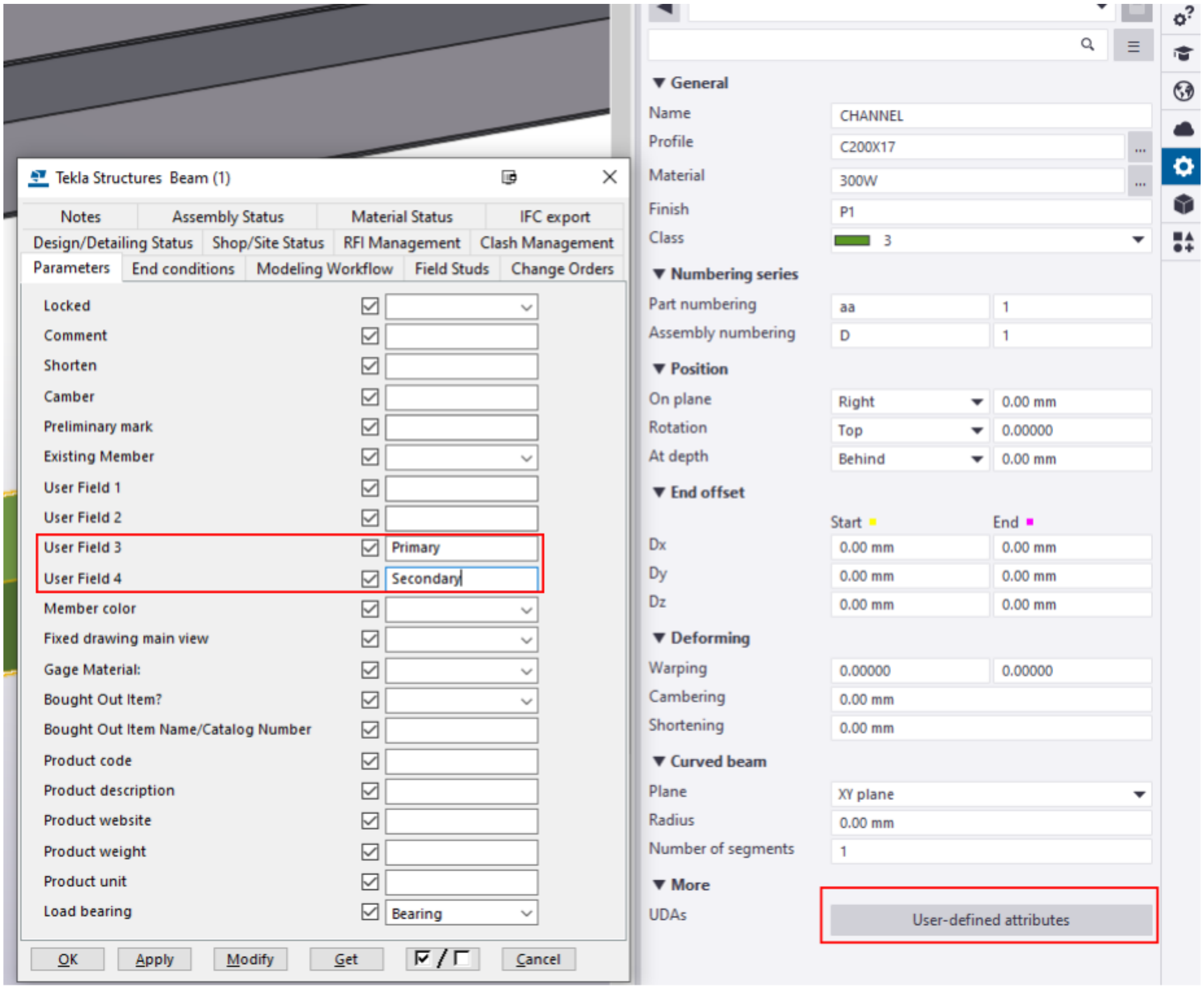
ColumnCollision(*TeklaMember* mem) – Checks to see if the members that a TeklaMember collides with are columns.

DeterminePrimaryMember(*Beam* beam1, *Beam* beam2) – Differentiates between the Primary and Secondary member of the two Tekla Beams. The method determines the distances of the end point of both members that is closest to the intersection point. The distance that is larger between the two results in that corresponding member being the Primary member. For example, the intersection point in both figures is the yellow point. In the left figure, the distance from the end point of Beam2 is larger than that of Beam1 (distance = 0), therefore Beam2 is considered the Primary member. In the right figure, the opposite occurs and Beam1 is considered the Primary member. This method isn't a fool proof way to determine which beam should be the Primary member. In case of uncertainty, the user can define which member should be the Primary member based on the next method.





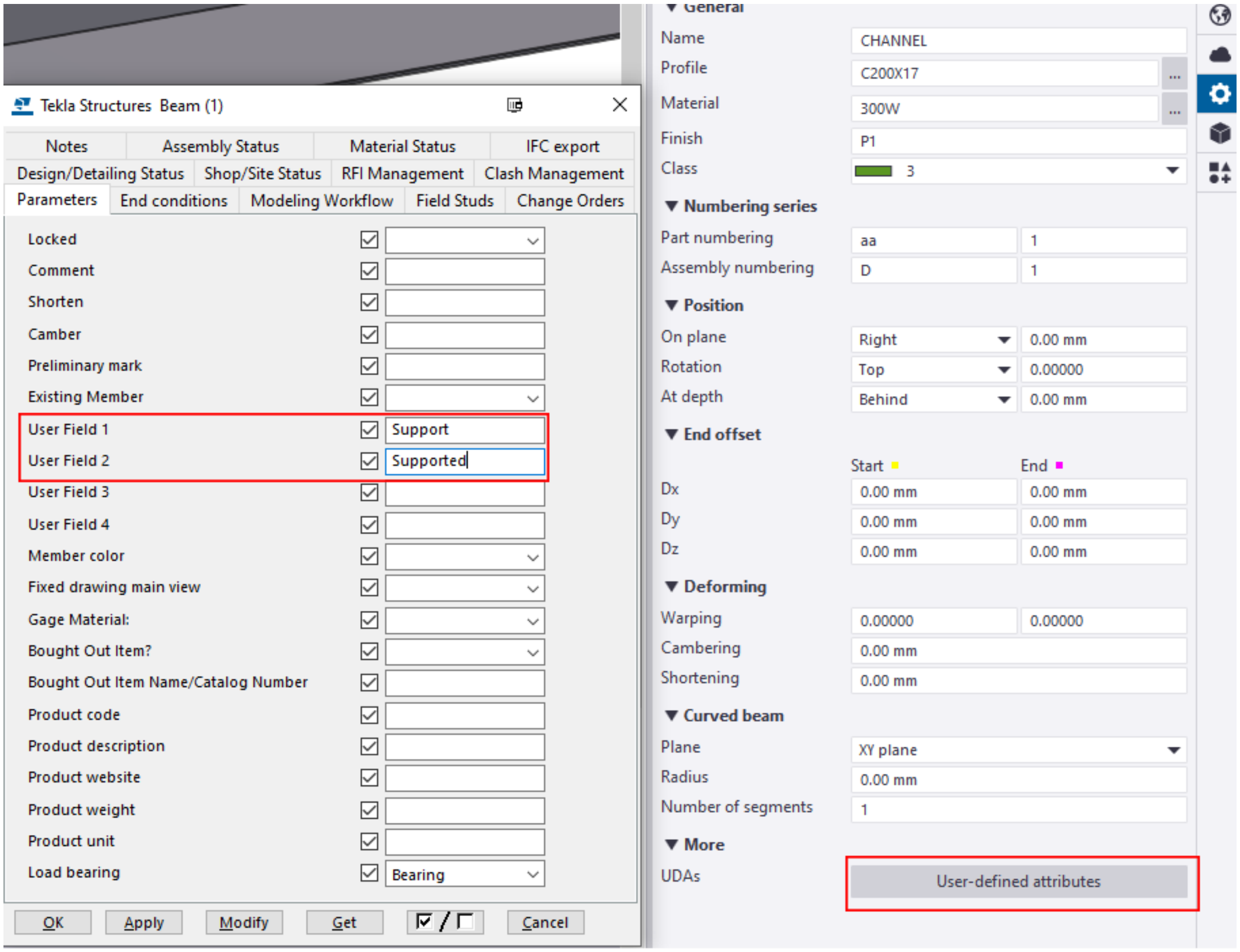
DeterminePrimaryMemberUDA(*Beam* mainBeam, *Beam* conBeam) – Determines the Primary member between two Tekla Beams using User Defined Attributes (UDA). Based on what is inputted by the user in the Beam's UDA under fields "USER FIELD 3" (for the start point of the member) and "USER FIELD 4" (for the end point of the member), the method manually determines the Primary member. In the future, there should be a method that can automatically determine this without user input; however, based on the Tekla Model alone, this is currently not possible.



Locked	☑	
Comment	☑	
Shorten	☑	
Camber	☑	
Preliminary mark	☑	
Existing Member	☑	
User Field 1	☑	
User Field 2	☑	
User Field 3	☑	Primary
User Field 4	☑	Secondary
Member color	☑	
Fixed drawing main view	☑	
Gage Material:	☑	
Bought Out Item?	☑	
Bought Out Item Name/Catalog Number	☑	
Product code	☑	
Product description	☑	
Product website	☑	
Product weight	☑	
Product unit	☑	
Load bearing	☑	Bearing

General		
Name	CHANNEL	
Profile	C200X17	
Material	300W	
Finish	P1	
Class	🟩 3	
Numbering series		
Part numbering	aa	1
Assembly numbering	D	1
Position		
On plane	Right	0.00 mm
Rotation	Top	0.00000
At depth	Behind	0.00 mm
End offset		
	Start	End
Dx	0.00 mm	0.00 mm
Dy	0.00 mm	0.00 mm
Dz	0.00 mm	0.00 mm
Deforming		
Warping	0.00000	0.00000
Cambering	0.00 mm	
Shortening	0.00 mm	
Curved beam		
Plane	XY plane	
Radius	0.00 mm	
Number of segments	1	
More		
UDAs	User-defined attributes	

DetermineSupportMember(*Beam* mainBeam, *Beam* conBeam) - Determines the Support member between two Tekla Beams using User Defined Attributes (UDA). Based on what is inputted by the user in the Beam's UDA under fields "USER FIELD 1" (for the start point of the member) and "USER FIELD 2" (for the end point of the member), the method manually determines the Support member. In the future, there should be a method that can automatically determine this without user input; however, based on the Tekla Model alone, this is currently not possible.



Locked ☑
Comment ☑
Shorten ☑
Camber ☑
Preliminary mark ☑
Existing Member ☑
User Field 1 ☑ Support
User Field 2 ☑ Supported
User Field 3 ☑
User Field 4 ☑
Member color ☑
Fixed drawing main view ☑
Gage Material: ☑
Bought Out Item? ☑
Bought Out Item Name/Catalog Number ☑
Product code ☑
Product description ☑
Product website ☑
Product weight ☑
Product unit ☑
Load bearing ☑ Bearing

▼ General
Name CHANNEL
Profile C200X17
Material 300W
Finish P1
Class 🟩 3

▼ Numbering series
Part numbering aa 1
Assembly numbering D 1

▼ Position
On plane Right 0.00 mm
Rotation Top 0.00000
At depth Behind 0.00 mm

▼ End offset
Start End
Dx 0.00 mm 0.00 mm
Dy 0.00 mm 0.00 mm
Dz 0.00 mm 0.00 mm

▼ Deforming
Warping 0.00000 0.00000
Cambering 0.00 mm
Shortening 0.00 mm

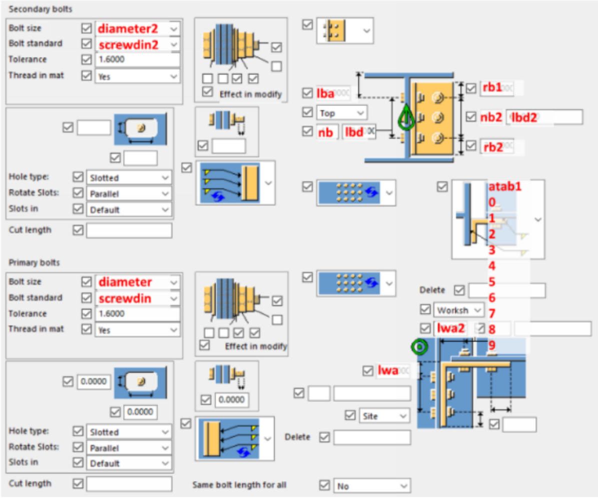
▼ Curved beam
Plane XY plane
Radius 0.00 mm
Number of segments 1

▼ More
UDAs User-defined attributes

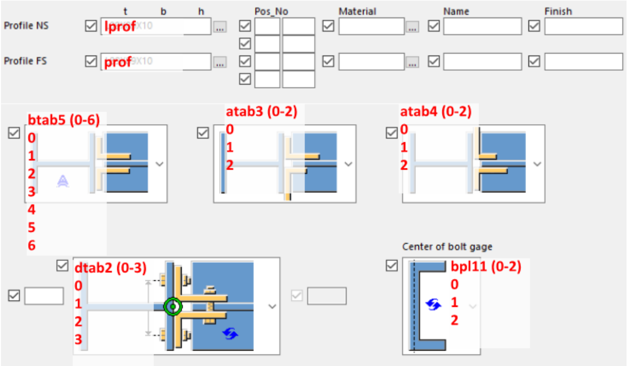
FindTeklaMemberCorrespondingToBeam(*List<TeklaMember>* teklaMembers, *Beam* beam) – Returns the index given a list of TeklaMember objects and a Tekla Beam.

Get141Connection(*Connection* con, *TeklaConnection* teklaCon) – Generates an angle connection in Tekla based on the properties of the TeklaConnection. The figure below represents the variables that correspond to their respective values in Tekla for Component141. It is important to note that the variables must be inputted in their proper types. This can be determined in many different ways the easiest is by breaking the component down using the TeklaGrasshopperLiveLink.

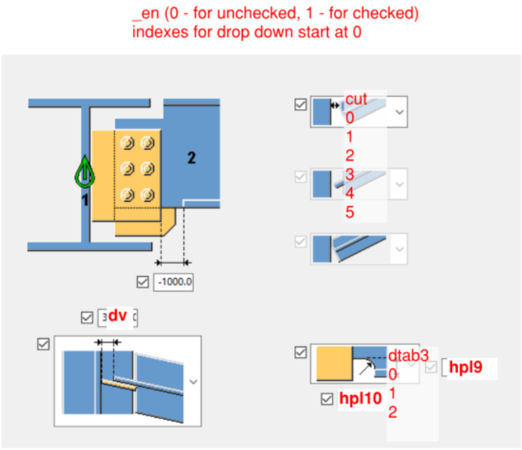
BOLTS

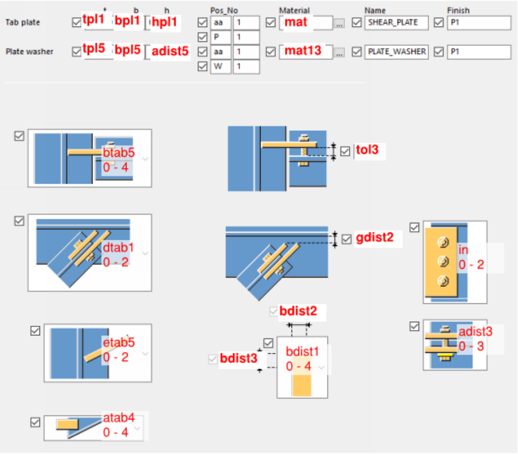


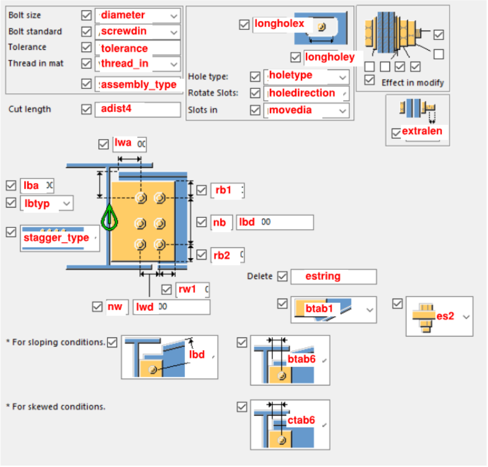
PARTS

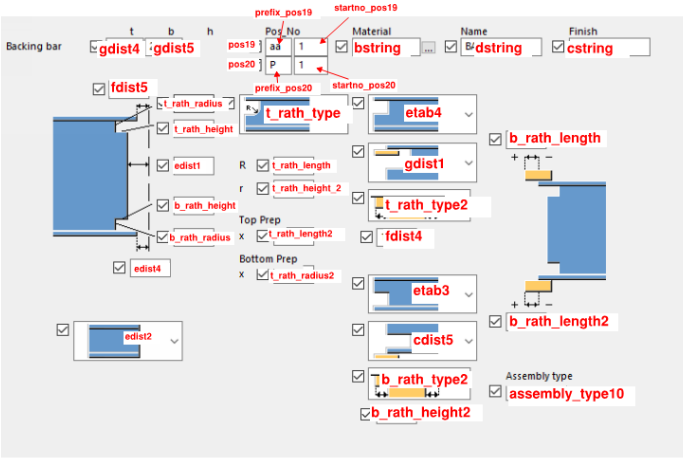


Get146Connection(*Connection* con, *TeklaConnection* teklaCon) – Generates a shear tab connection in Tekla based on the properties of the TeklaConnection. The figure below represents the variables that correspond to their respective values in Tekla for Component146. It is important to note that the variables must be inputted in their proper types. This can be determined in many different ways the easiest is by breaking the component down using the TeklaGrasshopperLiveLink.


_en (0 - for unchecked, 1 - for checked)
indexes for drop down start at 0
2
1
cut
0
1
2
3
4
5
-1000.0
dv
dtab3
0
1
2
hpl9
hpl10







GetEmptyConnection(*Connection* con, *TeklaConnection* teklaCon) – Returns an empty Tekla connection. This method is used to generate a Red Cone Connection in Tekla to signify to the user that the optimization procedure was unable to produce a suitable shear connection at the joint.

Get50PercentVr(*Beam* mem) – Returns 50% of the shear capacity of the member in kN. Based off of S16-19 Cl. 13.4.1.1 for unstiffened members only.

Get50PercentUDL(*Beam* mem) – Returns the corresponding shear force that is required to reach 50% of the members plastic moment capacity assuming that the load is uniformly distributed and the member is laterally braced.

Get60PercentUDL(*Beam* mem) – Returns the corresponding shear force that is required to reach 60% of the members plastic moment capacity assuming that the load is uniformly distributed and the member is laterally braced.

GetCostOptimizationData(*List<TeklaConnection>* conList) - Returns the total cost of the list of TeklaConnections. This is based on the cost function which may very well be updated in the future.

GetElevationDifference(*Beam* support, *Beam* supported, *Model* model) – Returns the elevation difference between the top of the supported member relative to the top of the support member. A positive value means that the supported member is elevated above the support member and a negative value means that the supported member is elevated below the support member.

GetEncroachment(*Section* sec) – Returns the encroachment of the given section based on the CISC Handbook 12th Edition page 3-64.

GetEndPointCollidingMember(*TeklaMember* member) – Returns the member (CollidingMember object) that collides at the End point of the TeklaMember. It determines this by going through the list of members that collide with the TeklaMember and determining the minimum distance between these members and the endpoint of the TeklaMember.

GetExcelColumnsForOptimizationData(*GH_Structure<IGH_Goo>* cons) – Populates the column headers for the spreadsheet that contains the optimization data.

GetHSSOrientation(*Beam* supported, *Beam* support) – Returns the orientation of the rectangular HSS with respect to the supported member.

GetMemberDepth(*Beam* mem) – Returns the depth of the member. Based off the property of the section in Tekla. If this property does not exist, it attempts to retrieve the depth of the member through geometry (may not be accurate).

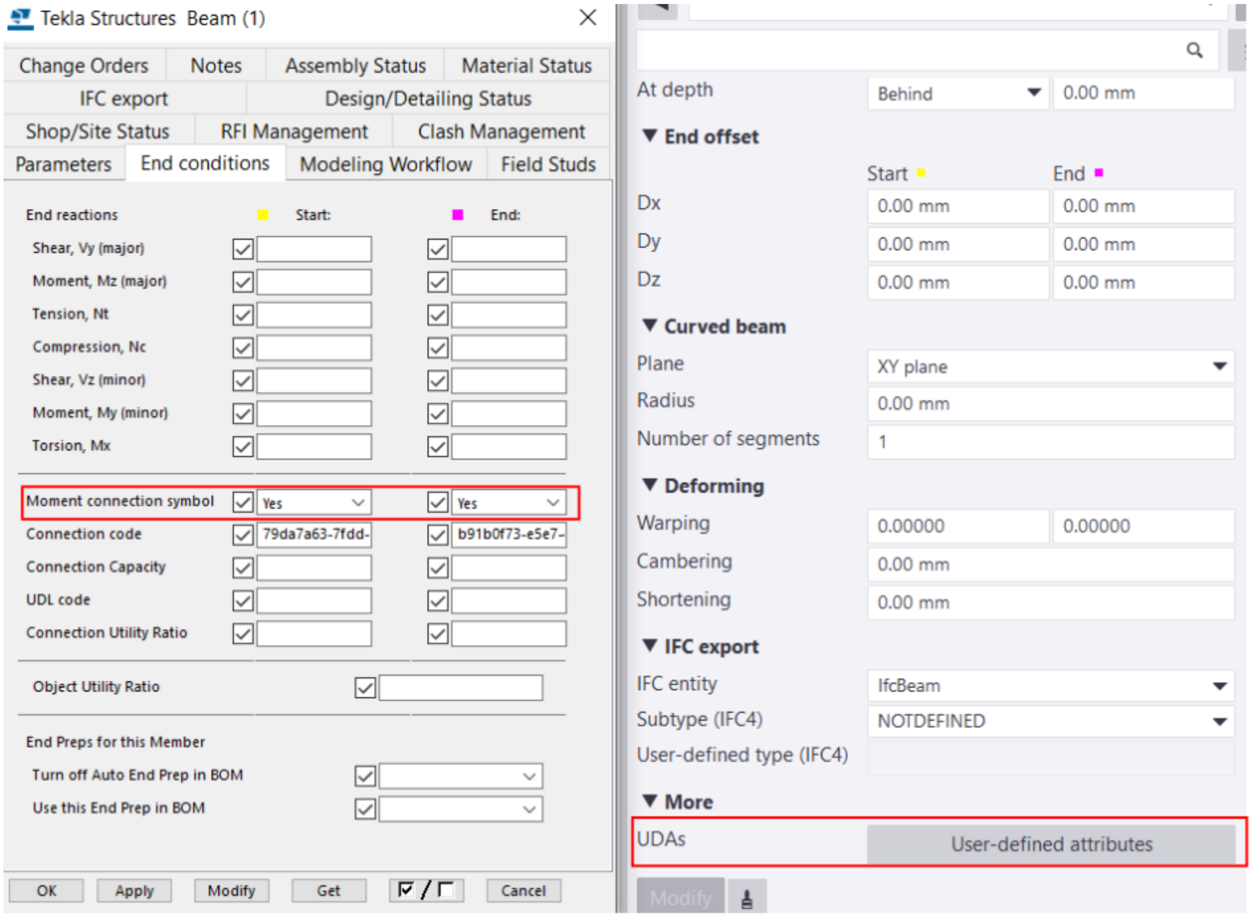
GetMemberWidth(*Beam* mem) – Returns the width of the member. Based off the property of the section in Tekla. If this property does not exist, it attempts to retrieve the depth of the member through geometry (may not be accurate).

GetWeightOptimizationData(*List<TeklaConnection>* conList) – Returns the total width of all the TeklaConnections.

LinearInterpolation(*double* x, *double* x0, *double* x1, *double* y0, *double* y1) – Performs linear interpolation.

RemoveAngleThicknesses(*List<string>* angleList, *double* thickness) – Removes angles from the list that do not satisfy the thickness requirement per S16.

RemoveMomentConnections(*TeklaMember* teklaMember) – Removes moment connections from CollidingMembers of a TeklaMember. Since moment connections cannot be geometrically determined, the user has to specify any moments that are present in the Tekla Beam through its UDA. See the figure below to determine which boxes should be used in the UDA of the Beam.



End reactions	🟨	Start:	🟪	End:
Shear, Vy (major)	☑		☑	
Moment, Mz (major)	☑		☑	
Tension, Nt	☑		☑	
Compression, Nc	☑		☑	
Shear, Vz (minor)	☑		☑	
Moment, My (minor)	☑		☑	
Torsion, Mx	☑		☑	
Moment connection symbol	☑	Yes	☑	Yes
Connection code	☑	79da7a63-7fdd-	☑	b91b0f73-e5e7-
Connection Capacity	☑		☑	
UDL code	☑		☑	
Connection Utility Ratio	☑		☑	

Object Utility Ratio ☑

End Preps for this Member

Turn off Auto End Prep in BOM ☑

Use this End Prep in BOM ☑

At depth: Behind 0.00 mm

▼ End offset

	Start 🟨	End 🟪
Dx	0.00 mm	0.00 mm
Dy	0.00 mm	0.00 mm
Dz	0.00 mm	0.00 mm

▼ Curved beam

Plane: XY plane

Radius: 0.00 mm

Number of segments: 1

▼ Deforming

Warping: 0.00000 0.00000

Cambering: 0.00 mm

Shortening: 0.00 mm

▼ IFC export

IFC entity: IfcBeam

Subtype (IFC4): NOTDEFINED

User-defined type (IFC4)

▼ More

UDAs: User-defined attributes

RemoveShortLegForBothLegsOfAngles(*List<string>* angleList) – Removes angles from the list of angles that cannot accommodate 2 columns of bolts on BOTH legs.

RemoveShortLegForLongLegOfAngles(*List<string>* angleList) – Removes angles from the list of angles that cannot accommodate 2 columns of bolts on the long leg.

RemoveWeldSizes(*List<int>* weldList, *double* thickness) – Removes weld sizes from the list of fillet welds that do not satisfy requirements based on an angle thickness.

SearchDictionaryForKey(*Dictionary<string, double>* dic, *string* str) – Returns the first key of a dictionary<string, double> based on a substring that exists within the key.

SortAngleList(*List<string>* angleList) – Sorts the list of angles from ascending order of long leg sizes then short leg sizes then thicknesses.

StartPointConnection(*Beam* mem1, *Beam* mem2) – Determines if Start Point of mem1 is closer than the EndPoint of mem1 to another mem2. Returns TRUE if this is the case and FALSE if not.

Plate.cs

The plate class is used to store information regarding a shear tab. The properties are not populated until the connection is designed.

SBB_Data_GH.cs

This class is used to create an object that designs and optimizes a single angle bolted-bolted (SBB) connection. The file is heavily commented therefore not much description is required in this document.

Section.cs

Section(*Beam* mem) or Section(*string* name) – A Section object can be created by passing a Tekla Beam or the name of the section. The properties of the section are determined from a JSON file containing all the properties of the sections. This JSON file was created based off the Sections Database.xlsx file. In the future, if any other sections are to be added, the spreadsheet should be updated and subsequently the JSON file. If the section is an HSS, there is additional case where thickness of the wall can be either 7.9 or 8.0 in the name. Since the JSON file only recognizes 8.0, it will convert 7.9 to 8.0 so that it can index the proper section. Two separate JSON files were used, one for standard shapes such as W- and C-shapes and another one for HSS shapes. This was required because the properties of these shapes differed enough from each other that using two JSON files was easier.

The properties ctRESET, cbRESET, dctRESET, and dcbRESET, are required due to issues with pass-by-reference during the optimization process.

Two additional classes, StandardShapes and HSSShapes, are used to pull information from the JSON files.

ST_Data_GH.cs

This class is used to create an object that designs and optimizes a shear tab (ST) connection. The file is heavily commented therefore not much description is required in this document.

SWB_Data_GH.cs

This class is used to create an object that designs and optimizes a single angle welded-bolted connection. The file is heavily commented therefore not much description is required in this document.

TeklaConnection.cs

This class is used to store the shear connection that will be created in Tekla. It is instantiated based off the Primary member (the member in Tekla that connects to this member) and the Secondary member (the member in Tekla is being connected to another member). The properties of the class are explained through the comments in the code and won't be discussed in this document.

GetAllowableConnectionDepth() – Obtains the allowable connection depth based off the framing configuration and whether the top flange, bottom flange or both flanges are coped. This method also considers the allowable encroachment based off the CISC Handbook 12th Edition.

GetCopingRequirements() – Obtains the coping configuration based on the framing geometry and user preferences. It accounts for the elevation difference between the two members as well.

GetFramingConfiguration() – Obtains the framing configuration of the connection, that is, "Beam-to-Beam", "Beam-to-HSS", "Beam-to-Column Flange", or "Beam-to-Column Web". It distinguishes between a connection to a column flange versus column web based off Tekla's GeometricPlane of the column. The GeometricPlane returns the plane that extends in the direction of the column's web. Then by comparing the angle between the GeometricPlane of the column and the vector of the supported member we can determine whether it connects to the column web or the column flange.

GetSkewAngle() – Obtains the skew angle of the supported member with respect to the support member. To obtain this angle, it determines the GeometricPlane of the support member and then projects the vector of the supported member onto the plane of the support member. It then uses the dot product to obtain the angle between vector of the support member and the projected vector of the supported member. Obtaining the correct plane to project the supported member's vector depends on the framing configuration. The reason the projection of the supported member's vector is required is for the cases where the supported member is also sloped. This method also assumes that the support member is not sloped.

GetSlopeAngle() – Obtains the slope angle of the supported member with respect to the support member. First, it determines if the slope is positive (rotated upwards with respect to the support member) or negative (rotated downwards with respect to the support member). It does this by comparing the Z values of the start and end points of the supported member with respect to the connection point. To obtain the slope angle, it projects the supported member's vector to the plane of the support member. This ensure that the true slope angle is determined for members that are skewed as well. It then uses the dot product, a vector(0, 0, 1) and the projected vector of the supported member to determine the angle.

GetVariableDataSBB(*List<string>* angleList) – Gets the ranges of variables that are associated with the single angle bolted-bolted (SBB) connection. It is dependent on the list of angles given by the user as it will eliminate angle sizes that cannot be possible based on the framing configuration and S16-19. This method is important when using metaheuristics to optimize the connection design.

GetVariableDataDBB(*List<string>* angleList) – Gets the ranges of variables that are associated with the double angle bolted-bolted (DBB) connection. It is dependent on the list of angles given by the user as it will eliminate angle sizes that cannot be possible based on the framing configuration and S16-19. This method is important when using metaheuristics to optimize the connection design.

GetVariableDataSWB(*List<string>* angleList, List<int> weldList) – Gets the ranges of variables that are associated with the single angle welded-bolted (SWB) connection. It is dependent on the list of angles and fillet weld sizes given by the user as it will eliminate angle sizes and fillet weld sizes that cannot be possible based on the framing configuration and S16-19. This method is important when using metaheuristics to optimize the connection design.

GetVariableDataDWB(*List<string>* angleList, List<int> weldList) – Gets the ranges of variables that are associated with the double angle welded-bolted (DWB) connection. It is dependent on the list of angles and fillet weld sizes given by the user as it will eliminate angle sizes and fillet weld sizes that cannot be possible based on the framing configuration and S16-19. This method is important when using metaheuristics to optimize the connection design.

GetVariableDataST(*List<double>* plateList) – Gets the ranges of variables that are associated with the shear tab (ST) connection. It is dependent on the list of plate thicknesses given by the user. This method is important when using metaheuristics to optimize the connection design.

LoadUserPreferences(*UserPreferences* userPref) – Generates the UserPreference object for the TeklaConnection object. Assigns the necessary properties to the connection based on user preferences such as steel grade, coping requirements, bolt size, etc.

PrintDesignInfo*(string* filePath) – Prints out all the relevant design information about the shear connection into a text file at the corresponding file path.

TeklaMember.cs

There are two classes in this file. TeklaMember is a class used to store information about a Tekla Beam. Information from this object is used to determine where the shear connection are in the model. The other class is CollidingMember. This class is used to determine which members collide with a specific Tekla Beam in model space. It is also used to determine whether the TeklaMember is the Primary, Secondary, Support, and Supported member. The methods in this file are all self-explanatory. This method is also primarily used in the ObtainShearConnectionJoints.cs Grasshopper component. Refer to that component for more information.

VariableData.cs

This class is used to store the parametric data used for metaheuristic optimization.

00. Miscellaneous

This folder contains a few miscellaneous Grasshopper components I created to help with the case studies.

ConnectionExtraction.cs

This component extracts the connection data from a Tekla Model. It only works for Tekla's component 141 and 146. The model must be filtered out for these two connections before the component can be used. It extracts the information about these connections and exports them into an Excel spreadsheet.

ConnectionFilter.cs

When using the ConnectionExtraction component, it's important that only Tekla Connection objects are taken from the model. This Grasshopper component filters out any objects that the user selects that are not Connection objects and removes them.

ConnectionObject.cs

This component is used to extract the Primary and Secondary model objects of a Tekla Component (connection). It only works for connections that have only ONE Secondary member.

1. Connections

This folder contains the Grasshopper components that are capable of assessing the shear connections that are currently available in the library (i.e., single angle bolted-bolted (SBB), double angled bolted-bolted (DBB), single angle welded-bolted (SWB), double angle welded-bolted (DWB), and shear tab (ST)). These components are not necessary for the optimization process, but can be used by the user as separate components if they wish to design individual shear connections.

DBB.cs

Component that is capable of assessing the adequacy of a double angle bolted-bolted (DBB) connection based on the provisions of S16 and AISC360. The capacity, the list of limit states as well as the governing limit state is outputted.

DWB.cs

Component that is capable of assessing the adequacy of a double angle welded-bolted (DWB) connection based on the provisions of S16 and AISC360. The capacity, the list of limit states as well as the governing limit state is outputted.

SBB.cs

Component that is capable of assessing the adequacy of a single angle bolted-bolted (DBB) connection based on the provisions of S16 and AISC360. The capacity, the list of limit states as well as the governing limit state is outputted.

ShearConnection.cs

This component is a combination of the other 5 components. Instead of having 5 separate Grasshopper components for each connection, this component allows the user to choose which one they want to use.

ST.cs

Component that is capable of assessing the adequacy of a shear tab (ST) connection based on the provisions of S16 and AISC360. The capacity, the list of limit states as well as the governing limit state is outputted.

SWB.cs

Component that is capable of assessing the adequacy of a single angle welded-bolted (SWB) connection based on the provisions of S16 and AISC360. The capacity, the list of limit states as well as the governing limit state is outputted.

2. Tekla

This folder contains all the relevant Grasshopper components that carry out the optimization process.

ConnectionGenerativeDesign.cs

This component is responsible for generating a list of feasible connections for every shear joint selected by the user in the Tekla Model. The user must input the list of shear connections which is obtained using ObtainShearConnectionJoints.cs. The user must also input a list of angles, plate thicknesses, and fillet weld sizes that they prefer to use. The user can also select the type of shear connections they wish to use for the joints.

Based on the selected shear connections from the user, each joint will use the Optimize method for the selected shear connections to create a potential feasible connection. Assuming the user selects all 5 types of shear connections in the library (i.e., SBB, DBB, SWB, DWB, ST), a joint at most will have 5 feasible types of connections. In the case that a specified connection type is incompatible for the joint, it will not produce a feasible connection. If none of connection types are adequate, then no feasible connections are possible for the joint and a red cone connection will be created in the Tekla Model. This indicates to the user that they must manually design the connection.

CreateShearConnection.cs

This component details the shear connection (only Tekla Components 141 and 146) into the Tekla Model.

ExportDesignCalculations.cs

This component exports the design calculation of the connection into a text file.

ExportToExcel.cs

Exports all the optimization data into an Excel Spreadsheet. Two sheets are created. The first sheet summarizes the information for the least cost and least weight option while the second sheet contains all the raw optimization data.

ObtainBuildingSteelVolume.cs

Determines the volume of steel in mm^3 of all the selected member in Tekla. This component is used to determine the volume of total steel material for the optimization process.

ObtainNumberOfParts.cs

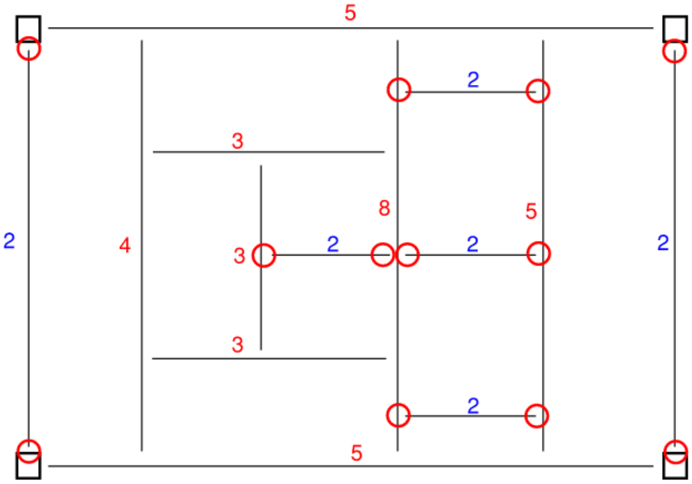
Determines the number of individual steel parts of all the selected member in Tekla. This component is used to determine the total number of steel members for the optimization process.

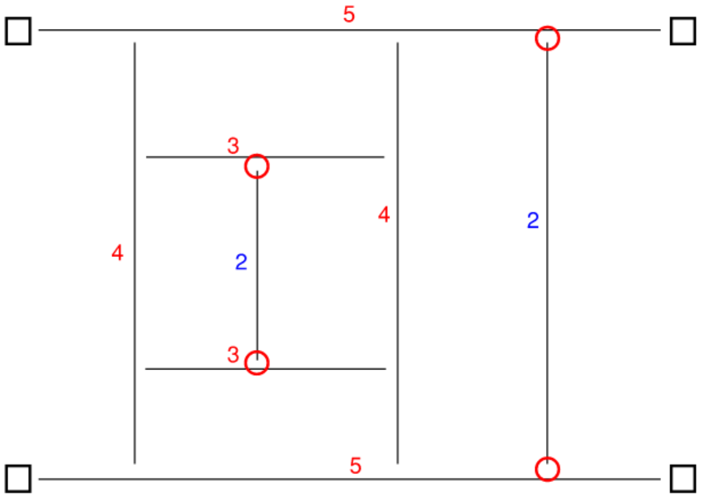
ObtainShearConnectionJoints.cs

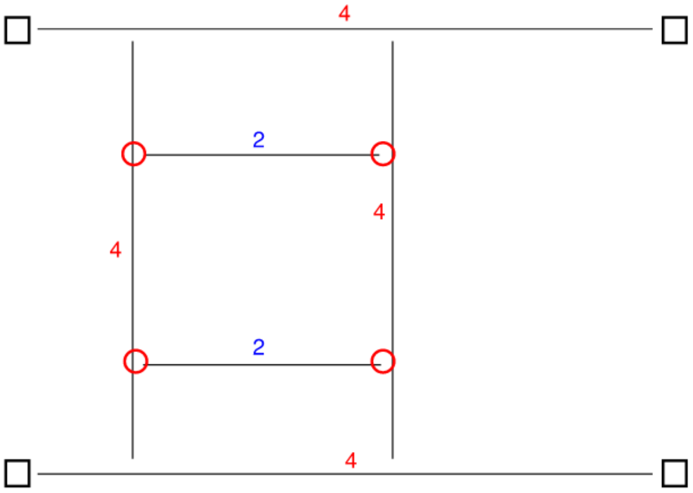
Obtains the **shear** joints in the model and its required design force. Since the user will be selecting all the members from the Tekla Model, this component is responsible for filtering out any joints that do not require shear connections.

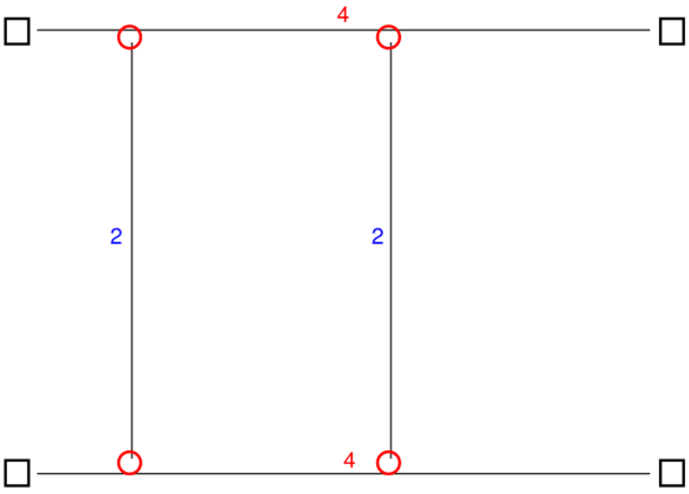
It first determines all the members that a beam collides with in model space. Secondly, it removes any colliding members from the beam that do not form a shear connection using the methods from Methods.cs. It then determines all the shear connection for beams with 2 or less collisions. As it determines where the shear connections exist, it removes the beam from the CollidingMembers of other beams until all the shear connections are determined.

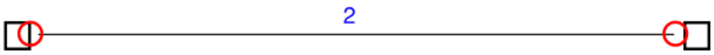
The figures below represent how the code works in a simplified case. The number indicates the number of colliding members with the beam while the red circles represent the shear connection that will be created at that location. The code executes in a sequential order of the figures from top to bottom.

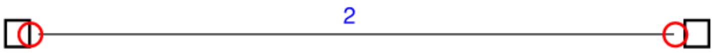



5
3
4
2
4
2
3
5


4
2
4
4
2
4


4
2
2
4





ProjectGenerativeDesign.cs

This Grasshopper component is used to optimize all the shear connections on a project-to-project basis. The optimization process uses the NSGAII algorithm. The user can choose whether they want to treat the problem as a single-objective problem (only minimizes cost) or a bi-objective problem (minimizes both cost and weight). More information about the logistics of how the optimization works can be found in the JMetalCSharp section.

Note that depending on whether the user selects a coping preference, it will change the calculation of the cost function. In the case of that the user prefers coping, any coping is assumed to be done on an automated coping machine and does not add any cost to the fabricator. However, in the case that the user does not prefer coping, any coped beams is assumed to be done manually and additional cost is incurred based on the cost function.

TestComponent.cs

Purely a component used for testing. The code in this component changes depending on what I'm trying to test.

JMetalCSharp

The metaheuristics used in the code was taken from jMetal which stands for Metaheuristics Algorithms in Java. It's an object-oriented Java-based framework for multi-objective optimization with metaheuristics. It was developed by Juan J. Durillo and Antonio J. Nebro. In 2015, a version of jMetal for the .NET framework was released which is used in this project. Additionally information regarding jMetal can be found in the following websites:

- https://jmetal.sourceforge.net/ (Java)
- https://jmetalnet.sourceforge.net/ (.NET Version)

The websites include the download as well as the manual, documentation, and academic papers associated with the framework.

The algorithms, problems, operators, and encoding in JMetalCSharp will not be explained in this manual with the exception of the problems I created in order to optimize the design of shear connections which are explained in the following sections.

Problems – ShearConnections

The Problems written for the 5 different types of shear connections are used if the user wishes to optimize the connections using metaheuristics as opposed to the written optimization script in the code. I believe that shear connections are simple enough that an optimization script is suitable as it drastically reduces the computational time. However, for more complex connections (future projects), optimization through metaheuristics may be ideal.

DoubleAngleAllBolted.cs

Optimization Problem for double angle bolted-bolted (DBB) connections. The variables and objectives can be found in the comments of the code. A constraint was included such that a connection that did not meet the requirements of S16 and AISC360 would not be included in the solutions.

DoubleAngleWeldedBolted.cs

Optimization Problem for double angle welded-bolted (DWB) connections. The variables and objectives can be found in the comments of the code. A constraint was included such that a connection that did not meet the requirements of S16 and AISC360 would not be included in the solutions.

ProjectCost.cs

Optimization Problem for optimizing all the shear connections in the project on a project-to-project basis as opposed to an individual connection basis. It takes the feasible connections at every joint in the project and evaluates different combination to produce options that result in the lowest fabrication costs only. It also considers erection issues that arise with bolted angle connections framing into the web of a column on either side.

ProjectCostWeight.cs

Optimization Problem for optimizing all the shear connections in the project on a project-to-project basis as opposed to an individual connection basis. It takes the feasible connections at every joint in the project and evaluates different combination to produce options that result in the lowest fabrication costs and material weight. It also considers erection issues that arise with bolted angle connections framing into the web of a column on either side.

ShearTab.cs

Optimization Problem for shear tab (ST) connections. The variables and objectives can be found in the comments of the code. A constraint was included such that a connection that did not meet the requirements of S16 and AISC360 would not be included in the solutions.

SingleAngleAllBolted.cs

Optimization Problem for single angle bolted-bolted (SBB) connections. The variables and objectives can be found in the comments of the code. A constraint was included such that a connection that did not meet the requirements of S16 and AISC360 would not be included in the solutions.

SingleAngleWeldedBolted.cs

Optimization Problem for single angle welded-bolted (SWB) connections. The variables and objectives can be found in the comments of the code. A constraint was included such that a connection that did not meet the requirements of S16 and AISC360 would not be included in the solutions.

ConnectionOptimizationTests

This class library contains all the unit tests that I wrote to test the majority of the methods as well as the code equations that were used for the design of all 5 types of shear connections. There are many methods that I couldn't write unit tests for, so I manually tested them instead. These tests are not captured here. I will note that I have come to realize that testing code is an art, and it takes many years of programming to become decent at it. This was the first time I ever did it for code I've written and did a relatively poor job doing it efficiently.

I will not go into detail as to what all the unit tests are here. The user should be able to read the code and understand what each test is aiming to achieve.

All the Tekla related unit tests required the TestModel to be open prior to executing the tests.