

UNIVERSITY OF BAHRAIN

College of Engineering



**Department of Civil
Engineering**

Optimum Design of Steel Frames with Concrete Filled Columns Using Metaheuristic Algorithms

A Thesis Submitted in Partial Fulfilment of the Requirements
for the M.Sc.Degree in Civil Engineering

Submitted by

Zaid Bashir Malkawi

20121435

Supervised by

Dr. Muhammad Ajmal

University of Bahrain

February 2024

بسم الله الرحمن الرحيم

Defense committee's approval

University of Bahrain
Deanship of Graduate Studies
And scientific Research



جامعة البحرين
عمادة الدراسات العليا والبحث العلمي

Thesis Discussion Committee Decision

Student Name: Zaid Bashir Malkawi	Student ID: 20121435
Department: Civil Engineering	College: Engineering
Program: Masters in Civil Engineering	
Thesis Title: Optimum Design of Steel Frames with Concrete Filled Columns Using Metaheuristic Algorithms	
Supervisor Name: Dr. Muhammad Ajmal	Student GPA: 3.46

The Discussion Committee met on 29-Feb-24 In Building S41, Room 0078. Upon discussing the student whose details are stated above in public, the discussion committee recommends the following:

The student is awarded a grade A- in the thesis subject.

- ☐ Without making any changes to the thesis.
- ☒ After making the modifications attached with this decision.

Discussion Committee Members Approval:

Name

1. Dr. Muhammad Ajmal (Advisor)
2. Dr. MD Shah Alam (Internal Examiner)
3. Dr. Andi Asiz (External Examiner)

Signature

Abstract

The aim of optimization is to achieve the most favorable design considering a prioritized set of criteria or constraints. In the realm of engineering advancements, it is crucial for engineers to explore beyond conventional methods. While traditional approaches may provide comfort and familiarity, designers must recognize the increasing significance of optimization methods. These methods play a crucial role in saving both time and money while simultaneously reducing errors. This research represents a modest stride toward comprehending the advantages that optimization can bring to structural design. It aims to contribute to one of the primary goals of every engineer: designing structures that are not only safe but also economically and environmentally sustainable.

To accomplish this, design optimization algorithms such as ABC (The Artificial Bee Colony), TLBO (Teaching Learning Based Optimization), and DE (Differential Evolution) were employed to determine the optimal cross-sectional dimensions of columns following the AISC-LRFD design provisions. The software tool ETABS was utilized to obtain the frame's response to external loads, and the optimization iterations were executed using MATLAB. Altering the cross-sectional properties of the members in the 3-D steel frame leads to changes in their performance under external loads, necessitating a new analysis. This is facilitated through the use of the Application Programming Interface (API) feature of ETABS, establishing a link between the coded MATLAB program and ETABS for bidirectional data exchange. Iterations continue until the convergence criteria are satisfied.

It was found that the utilized optimum design algorithms were able to optimize the weights and reduce them to the minimum possible sections. It was also found that the proposed TLBO and DE are efficient and better than the standard ABC in terms of the convergence history and providing the best results.

Publications

No publications.

Dedication

I dedicated this project to God first I wouldn't have gotten here without his grace and the strength that he gave me to start a new chapter in life to learn more and benefit the nation. I also dedicated this work to my supervisor Dr. Muhammad Ajmal who has encouraged me to do my best and be the source of wisdom I need in my journey. To my parents, my support is Bashar (father), Samar (mother), and my wife Batool. Thank you for being there for me. God bless you all.

Acknowledgments

I want to take this opportunity to express my gratitude thanks for my supervisor, Dr. Muhammad Ajmal to give me from his time to listen, guide me through my research, being very helpful and consider me as one of his priorities. And Prof. Dr. M.P. Saka for his support during proposal writing and initiating this research topic.

My thanks to my parents who support me from the start and my wife.

My thanks to my friends my brothers.

And my special thanks to my partner Marwan Ezdini for his moral support and being by my side whenever I need him

Table of contents

Defense committee’s approval	A
Abstract.....	B
Publications	C
Dedication	D
Acknowledgements	E
Table of Contents	F
List of Tables	H
List of Figures	J
List of Abbreviations	K
List of Appendices	M
1 Introduction.....	2
1.1 General.....	2
1.2 Research Problem	8
1.3 Research Objectives.....	8
1.4 Research Significance.....	9
1.5 Research Scope	10
1.6 Research Outline.....	11
2 Literature Review	13
2.1 Introduction.....	13
2.2 Column Design	15
2.3 Columns Optimum Design	17
2.3.1 Solution Techniques	19
2.4 Metaheuristic Algorithms	20
2.4.1 Genetic Algorithms (GA)	20
2.4.2 Simulated Annealing (SA).....	21

2.4.3	Harmony Search (HS).....	22
2.4.4	Particle Swarm Optimizer (PSO)	23
2.4.5	Cuckoo Search (CS)	23
2.4.6	Colliding Bodies Optimization (CBO)	24
2.4.7	The Artificial Bee Colony (ABC)	25
2.4.8	Teaching-Learning-Based Optimization (TLBO)	27
2.4.9	Differential Evolution (DE).....	28
2.5	Literature Review for the Optimum Design of Columns	29
3	Optimum Design of Composite Steel Column Filled with Concrete	35
3.1	Introduction.....	35
3.2	Composite Column Model.....	36
3.3	Design Optimization of the Composite Columns.....	44
3.3.1	Objective Function.....	45
3.3.2	Design Variables.....	45
3.3.3	Design Constraints.....	58
4	Methods and Procedures.....	63
4.1	Application Programming Interface (API)	64
4.2	Metaheuristic Algorithms	68
4.2.1	Artificial Bee Colony Algorithm.....	68
4.2.2	Differential Evolution (DE).....	71
4.2.3	Teaching-Learning-Based Optimization (TLBO)	73
4.3	Composite Column Design Optimization Algorithm.....	76
4.4	Summary	79
5	Results and Discussion	81
5.1	Composite Steel column filled with concrete design optimization.	81
5.2	Hand Calculation	92
5.2.1	For the inner column on floor 1	92

5.2.2	For the outer column on floor 1.....	95
5.3	Summary	98
6	Conclusions.....	100
6.1	Main Findings	100
6.2	Recommendations for Future Work	103
6.3	Recommendations and Guidelines for this Research	104
	References	106
	Appendix A Column Design MATLAB Code.....	112
	Appendix B Algorithms and API MATLAB Code	123
	المخلص	139

List of Tables

3.1	Details of beams and columns members and grouping for a single-storey building	38
3.2	Details of beams and columns members and grouping for a three-storey building	40
3.3	Details of load combination arrangement	42
3.4	W-sections	46
3.5	HSS	54
4.1	API functions used	65
5.1	Metaheuristic parameters for the composite column example	82
5.2	Metaheuristic algorithms result in the frame structure example	82
5.3	Metaheuristic algorithm results for the composite column example	91
6.1	Optimum cross-sectional properties for the steel I-beam	101
6.2	Optimum cross-sectional properties for the composite column	101

List of Figures

1.1	Shapes of Columns	5
1.2	Types of Reinforcement in Columns	6
1.3	Materials of Columns	8
2.1	Ancient columns at Petra Jordan	13
2.2	Bahrain Airport	14
2.3	The flowchart of the basic ABC algorithm	26
2.4	The flowchart of the TLBO algorithm	27
3.1	Composite-filled steel column cross-section	35
3.2	Composite Covered steel column cross-section	36
3.3	Single-storey building model 3D view	36
3.4	Single-storey building model top view	37
3.5	Three-storey building model 3D view	39
3.6	Storey 3 of a three-storey building model top view	39
4.1	The interaction between the three main optimization parties	77
5.1	DE algorithm running history	89
5.2	TLBO algorithm running history	89
5.3	ABC algorithm running history	90
5.4	The three metaheuristic algorithms running history	90
5.5	The three metaheuristic algorithm constraint running history	92
5.6	Inner column Mux max	94
5.7	Inner column Pu max	94
5.8	Inner column Muy max	95
5.9	Outer column Mux max	97
5.10	Outer column Pu max	97
5.11	Outer column Muy max	98

List of Abbreviations

AASHTO	American Association of State Highway and Transportation Officials
ABC	Artificial Bee Colony
ACI	American Concrete Institute
ACO	Ant Colony Optimization
AISC	American Institute of Steel Construction
API	Application Programming Interface
ASCE	American Society of Civil Engineers
BS	British Standard
CBO	Colliding Bodies Optimization
CS	Cuckoo Search
CSI	Computers and Structures Incorporation
DE	Differential Evolution
DL	Dead Load
GA	Genetic Algorithm
GCC	Gulf Cooperation Council
GSA	Glowworm Swarm Algorithm
HS	Harmony Search
IB	Inner Beam
IC	Inner Column
LRFD	Load and Resistance Factor Design
LL	Live Load
OB	Outer Beam
OC	Outer Columns
PSO	Particle Swarm Optimizer
SA	Simulated Annealing
SID	superimposed dead load

TLBO	Teaching-Learning-Based-Optimization
W_x	wind load in x direction
W_y	wind load in the y direction

List of Appendices

Appendix A	Column Design MATLAB Code
Appendix B	Algorithms and API MATLAB Code

Chapter One

Introduction

1 Introduction

1.1 General

Optimization involves choosing the option based on a specific criterion, from a range of available alternatives. In the simplest case, the optimization problem consists of maximizing or minimizing a real function by systematically selecting input values that satisfy several constraints if there are any. Applied mathematics encompasses a domain where optimization theory and techniques are widely generalized to formulas. In general, optimization involves finding the best available values for some objective functions.

Optimization problems typically have three fundamental elements. The first is an objective function, which is to be maximized or minimized. For the objective functions many types of engineering problems such as figuring out the most suitable path for vehicles to reach a certain point, figuring out the ideal shape of planes, and getting the minimum weight of structures. The second element is the variables, which are the values that can be manipulated in order to optimize the objective. Examples include the route to be followed by a vehicle through a traffic network, column characteristics, unit weight of each material, and air friction. The third element of an optimization problem is a set of constraints, which are restrictions on the values that the variables can take. For instance, a manufacturing process cannot require more resources than are available, nor can it employ less than zero resources, and the thickness of the beam should not be negative. The application of optimization saves time, energy, and cost, which leads to efficiency in the economy.

There are two main approaches for solving optimization problems, deterministic and stochastic. Deterministic or traditional methods are mathematical-based techniques that use derivatives of the objective function and the constraints in finding the solution and do not use random search. On the other hand, stochastic methods – also known as metaheuristic methods – are heuristic techniques that apply random searches and do not use derivatives of the equations (Saka & Geem, 2013).

In structural engineering, the column is a structural element that transfers the weight of the structure above to the foundation by compression. Columns play a role in supporting the structure and if columns fail it can lead to the complete collapse of the entire building. Columns are commonly used to support beams or arches that support the upper portions of walls or ceilings. It can also be a decorative element that is not required for structural purposes.

Different column types are used in structures. It can be classified into these conditions:

- 1) Based on the shape.
- 2) Based on the type of reinforcement.
- 3) Based on the type of loading.
- 4) Based on the slenderness ratio.
- 5) Based on materials.

Column types based on shape

- a) Square and rectangular columns are the most used in building construction because they are economical and easy to construct due to readily shuttering and reinforcement placement. Also,

because it is used in square and rectangular rooms we can hide the columns on the side of the wall.

- b) Circular columns are used usually for piling and elevation of buildings for aesthetic purposes. It is almost higher than the square and rectangular columns in bending resistance. It is also good in deflection resistance, so it is used as a bridge pillar.
- c) L Type column is used usually in boundary wall and flyover construction.
- d) T Type column has the same features as square and rectangular columns, and it is used usually in bridge and retaining wall construction.
- e) Y Type column is used usually in bridge and flyover construction.
- f) Hexagonal columns are usually reformed columns. It contains six sides, and it offers an aesthetic view normally applied in elevation. The purpose of this column is to provide a good appearance to the column. It is used usually in open verandahs, auditoriums cinema halls, etc.
- g) Octagonal column
- h) V-type column
- i) Arch type column

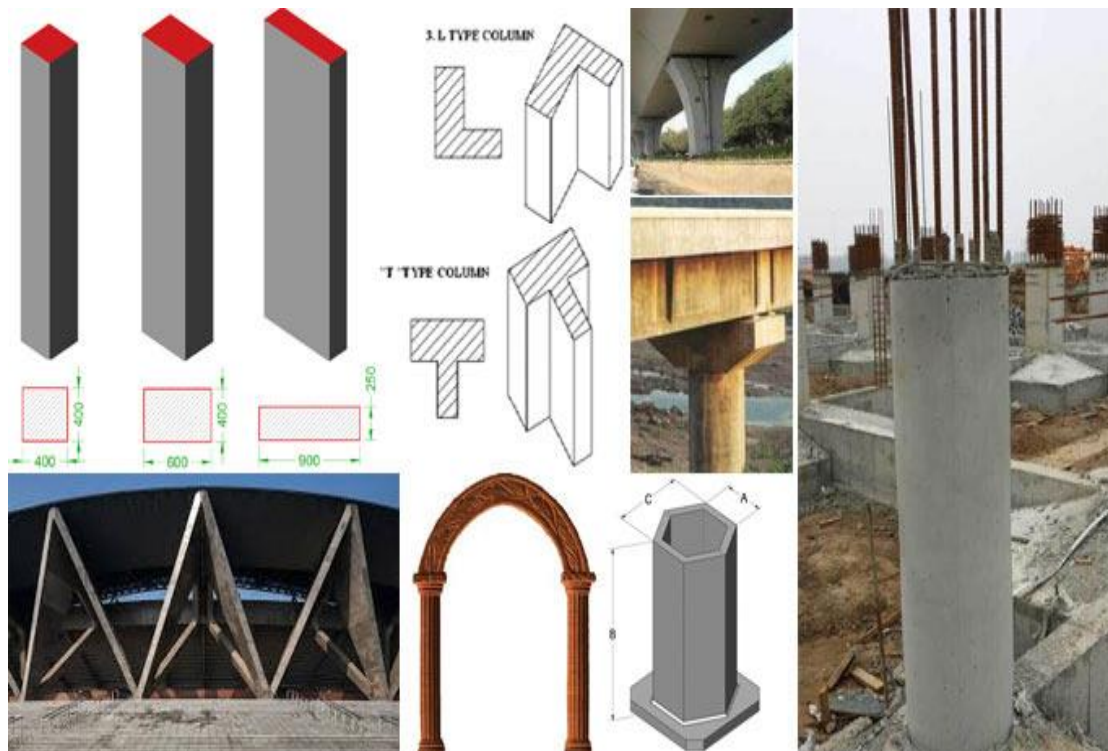


Figure 1.1 Shapes of Columns (Construction Cost)

Column types are based on the type of reinforcement.

a) Tied column

These columns feature side links that are evenly spaced and uniformly distributed along the column. The distance between the links is carefully controlled to ensure that the column remains structurally sound while allowing for installation. If the links are placed apart the column may suffer from shear failure and experience separation, between the links.

b) Spiral Column

Spiral columns are formed when round columns are reinforced with stiffeners that converge and are evenly spaced. The purpose of the spiral is to offer support in the direction and prevent the column from getting flattened.

c) Composite Column

These columns are typically made of either steel or cast iron. Are surrounded by concrete. They are reinforced with spiral reinforcement to provide strength and durability. Alternatively, a tube steel column filled with concrete can also be used. Composite columns are commonly employed in structures to prevent steel corrosion, local buckling and to increase the load-bearing capacity.

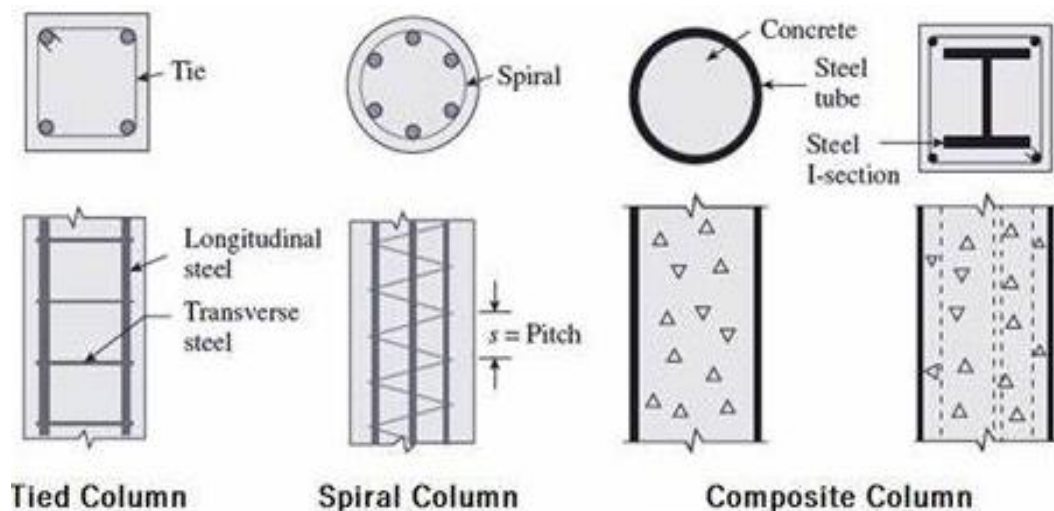


Figure 1.2 Types of Reinforcement in Columns (Ramesh konchada & Bruce Pelletier, 2023)

Column types based on the type of loading

a) Axially loaded column

In construction projects usually they use these types of columns. When the resultant of the loads passes through the centroid of the cross-section of a column.

b) Eccentrically loaded column

If a column's resultant load does not align with the centroid of its cross-section.

- Columns are loaded uniaxially eccentrically.

When the eccentric load acts on either X-axis or Y-axis separately.

- Columns are loaded biaxially eccentrically.

When the eccentric load acts on the X-axis or Y-axis together.

Column types based on slenderness ratio

A column is considered short if the ratio of its effective length to its least lateral dimension is less than 12; otherwise, it is considered long.

a) Short column

b) Long column

Short columns are generally preferred over long columns of a similar cross-sectional area as they tend to be stronger.

Column types based on materials used

a) Reinforced concrete column

b) Steel column

c) Timber column

d) Stone column

e) Brick column

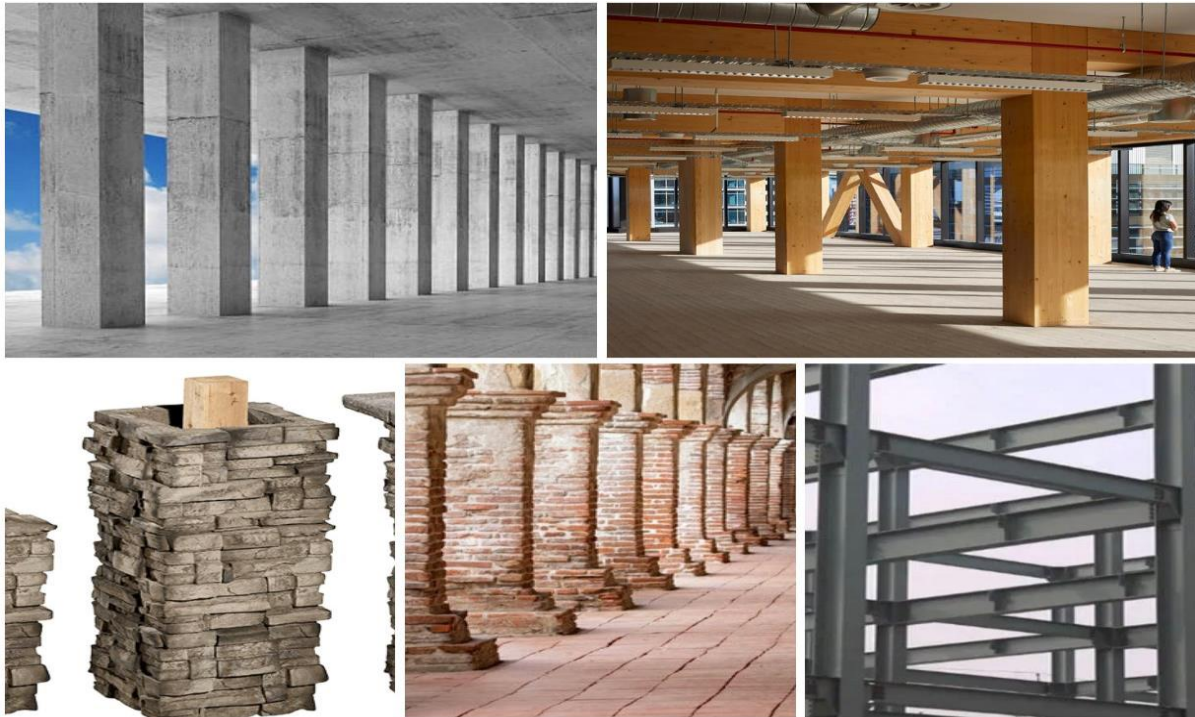


Figure 1.3 Materials of Columns (M. Fahim, 2022)

1.2 Research Problem

Optimizing the design of composite steel columns filled with concrete to reduce their weight while meeting AISC-LRFD design code requirements by changing the cross-sectional properties of column elements.

1.3 Research Objectives

- To develop an optimum design algorithm for composite steel columns filled with concrete.
- To obtain the optimum safe design by AISC-LRFD design code specifications.
- To achieve the minimum weight design for the selected columns.

- To select some of the recent metaheuristic algorithms that have not been used previously in the column design optimization literature and study their performance in obtaining the optimum solution.
- This research intends to make the algorithm to decide whether the column sections should be steel rectangular hollow sections or tubular sections.
- To study the capability of these metaheuristic algorithms in solving large dimensional problems.
- To improve the performance of one of the selected metaheuristic algorithms and test its efficiency in reaching the optimum solution.

1.4 Research Significance

Designing buildings optimally reduces the use of construction material which in turn reduces the cost of the structure. Furthermore, there will be a contribution to sustainability because of the reduction achieved in the production of these construction materials. Using fewer materials means less CO₂ emission to the atmosphere. The use of concrete-filled columns provides cost reduction in steel structure construction. Providing optimum design will offer a further reduction in the material cost. Designing with the developed algorithm will increase the margin of profit in the construction because the material cost will be minimal.

1.5 Research Scope

The scope of this research is to develop an optimum design algorithm for 3-D steel frames that necessitates a combination of two separate actions. The first one is to model the steel frame in a commercial structural analysis software such as ETABS (building analysis and design program developed by Computers and Structures, Berkeley, USA). This software will be used to attain the response of the frame under the external loads. After completion of the modeling of the structure in ETABS(CSI Computer and Structures, 2020), optimization algorithms will be programmed in MATLAB(MathWorks, 2019). For the composite column, the objective function was defined as minimizing the weight of the steel and concrete sections. The optimum design problem of 3-D steel frames when formulated by taking the cross-sectional properties of its members as design variables and implementing the design constraints from the AISC-LRFD design code turns out to be a discrete nonlinear programming problem. The metaheuristic algorithms are efficient in obtaining the solution to such optimization problems. There is a large number of metaheuristic algorithms available in the literature. Their performance is problem-dependent. The study intends to find the appropriate one among the metaheuristic algorithms recently developed that performs the best in obtaining the optimum solution. The solution to the optimum design problem of the composite column was obtained using three metaheuristic algorithms: artificial bee colony algorithm (ABC), teaching learning-based optimization (TLBO) and differential evolution (DE) are the recent ones that are planned to be employed in the study. These algorithms incorporate a natural phenomenon to update design variables in each design cycle. When the cross-sectional properties of the members in the 3-D steel frame are altered, their performance under

external loads also changes. Consequently, a new analysis is necessary. This will be achieved by making use of the application programming interface (API) feature of ETABS. This facility establishes a link between the coded MATLAB program and ETABS, enabling the exchange of data in both directions. Iterations will persist until the convergence criteria are met.

1.6 Research Outline

The thesis comprises six chapters. Chapter One outlines the introduction to the study. Chapter two discusses the theoretical framework and provides a literature review of composite column design optimization. Chapter three explains the optimum design of the selected frame examples. Chapter four describes the methods and procedures. Chapter five illustrates the results and discussions. Finally, chapter six provides the conclusions.

Chapter Two

Literature

Review

2 Literature Review

2.1 Introduction

The column is one of the oldest building elements, and its shape and material composition vary depending on age and locality. The column was an architectural development that allowed roofs to be supported without solid walls, enhancing the amount of space a roof could cover. Columns could also be integrated into the walls to add beauty to the room's designs and grant more unrestraint organizing and decoration of those rooms. A column can also be non-structural and used for decoration or as a standalone memorial.

Columns were initially used in construction by the ancient Sumerians and Yemenis, and they were made of a bundle of reeds. The Mesopotamian civilization employed clay to create columns, but it did not last. Stones were employed in the construction of columns, which received worldwide acclaim. This column type was popular among Middle Eastern civilizations, particularly among the ancient Egyptians, Persians, and others. Stone columns evolved and flourished enormously in Greece and Rome, and columns arose in a variety of styles. Remnants of these columns can still be found to this day in ancient ruins around the world.



Figure 2.1 Ancient columns at Petra Jordan (The Treasury, 2008)

Modern columns tend to be made of iron, steel, concrete, or a combination of those various materials and are simply designed. The design of today's columns and the materials used are efficiently nominated to meet the requirements. Reinforced columns are widely used nowadays for their ability to withstand greater loads. They are designed to withstand both compression and tension loads as opposed to the old design of columns which didn't contain steel for reinforcement. Steel columns with their high durability allowed for the construction of skyscrapers and in hangers and showrooms, they minimized the number of columns needed which created bigger entrances if required.



Figure 2.2 Bahrain Airport (Aviation Week Network, 2021)

In the future, there will be developments in all aspects of life including structures. Modern construction is moving towards the composite materials. Many structural engineers are utilizing concrete and structural steel composite elements to get the strength requirements by minimizing structural weight. One of the example of such composite elements is

concrete-filled steel column which can take greater loads with less materials, making it much more cost-effective. This design prevents the local buckling of the steel tube by filling it with concrete. In doing so one can eliminate one of the most challenging issues in using steel hollow sections which is lateral buckling.

2.2 Column Design

Columns are designed according to available code specifications and standards. One of the most widespread specifications applied for column design around the world, including in Bahrain and the surrounding regions, is the ACI 318-19 : Building Code Requirements for Structural Concrete and Commentary (Reapproved 2022) and AISC Code of Standard Practice for Steel Buildings and Bridges: ANSI/AISC 360-16. These codes apply the general rules, loads, and resistance factor procedure for building structures. These factors were developed from the reliability theory, which depends on the current statistical data of loads and performance of structures and is concerned with the probability of failure of a system. Accordingly, the columns are designed for particular limit states that consider the number of loads and the resistance of the structure.

The limit states in these codes are defined according to the LRFD method. A general statement for assuring safety in engineering design is that the resistance of the components must not under any circumstances be exceeded by the loads applied on it, that is:

$$\text{Resistance } (R) \geq \text{Applied Loads } (Q) \quad (2.1)$$

To account for the variability of both sides of the equation, the resistance side is multiplied by a statistically based resistance factor (ϕ), [$\phi \leq 1$] and

the applied load side is multiplied by a statistically based load factor (γ), [$\gamma \geq 1$]. Because the load effect at a particular limit state involves a combination of different load types (Q_i) that have different degrees of predictability, the load effect is represented by a summation of ($\gamma_i Q_i$) values. If nominal resistance is given by (R_n), the safety criterion is:

$$\phi R_n \geq \sum \gamma_i Q_i \quad (2.2)$$

Since the above equation involves both load factors and resistance factors. The design technique is referred to as Load Resistance Factor Design (LRFD). The resistance factor (ϕ), for any limit state, must account for the uncertainties in material properties, strength predicting equations, workmanship, quality control, and failure consequence. (Awad E. Ajeel, 2021)

Also, the load factor (γ_i), for any load type must consider the uncertainties in loads magnitude, loads arrangement (positions), and possible loads combinations.

Steel-concrete composite columns may be designed either by requirements of the American Concrete Institute Building Code ACI 318-99 or by the American Institute of Steel Construction Specifications for Load and Resistance Factor Design AISC-LRFD. Each design standard is described for application to a concrete-filled steel tube and to a concrete-encased structural shape as each is designed for the same dimensional and service load conditions. These standard types of column sections are used for the comparison, as the LRFD specification can be used directly only for such standard sections. The design exercise demonstrates that a) the LRFD specification requires fewer computational steps and is, therefore, easier to apply, b) the ACI rules tend to exaggerate the influence of slenderness, and c) different but very similar results were obtained for the

two methods applied to the same design problem. (R. W. Furlong, 2000)
Columns are subjected to two types of loading: dead loads and live loads. Dead loads are loads that do not change with time, such as the self-weight of structural and non-structural elements and loads of wearing surfaces and utilities, while live loads are loads that do change with time, for example, residential live loads, furniture live loads, wind loads and seismic loads.

2.3 Columns Optimum Design

Column design is a complex decision-making problem. To produce a safe and cost-effective design, the designer must meet the design requirements. Column analysis is complicated. The development of design optimization algorithms has given structural design engineers a powerful tool for optimizing structural design challenges. In these cases, the engineer can optimize the cross-sectional size of members, the number of structural parts, and so forth, leading to a minimum weight or cost while still adhering to the design code and safety requirements. This is accomplished by transforming the design challenge into an optimization problem and solving it as such.

In general, the mathematical formulation of a structural design problem can take the following form:

Minimize

$$W = f(x_i) \quad i = 1, 2, \dots, n \quad (2.3)$$

Subject to:

$$g_j(x_i) \leq 0 \quad j = 1, 2, \dots, m \quad (2.4)$$

$$x_i^l \leq x_i \leq x_i^u \quad (2.5)$$

where W is the objective function $f(x_i)$ is the function that has to be minimized, $g_j(x_i)$ is the design constraint j , and x_i is the design variable i that has a lower bound x_i^l and an upper bound x_i^u .

The function $f(x_i)$ that represents the objective function measures the effectiveness of the design. It represents the minimum weight or minimum cost in the structural design problems. The design constraints $g_j(x_i)$ represent the restrictions imposed on the structure. These restrictions are the design code requirements such as ultimate strength and serviceability requirements in addition to practical manufacturing limitations.

In structural optimization problems, the design variables x_i are the parameters that govern the structural geometry or material properties. The value of the design variable x_i may be continuous or discrete. For continuous design variables, the value of x_i can be any real value within the lower and upper bounds interval $[x_i^l, x_i^u]$. Further, if the design variable has to be an integer value selected from a list of design variables within the limits of bounds, it becomes a discrete variable. For example, if the design variable is the cross-sectional area of a structural steel member that has to be manufactured locally, then it is considered a continuous design variable. Otherwise, if the steel members have to be selected from the available commercial list, then the design variable is discrete.

Mathematical formulation of an optimum column design problem may differ based on the selected column type. Moreover, the method of structural analysis can be changed from linear to nonlinear.

The mathematical formulations of composite steel columns filled with concrete according to ACI and AISC specifications are described later, in Chapter 3.

2.3.1 Solution Techniques

There are two types of optimization problem-solving approaches available: deterministic and stochastic. Mathematical programming algorithms are another name for deterministic algorithms. These techniques require the gradient information/derivatives of both the objective function and the design constraints. They also need the starting points of the design variables to start the solution. The selection of these points impacts the final solution's quality. As a result, if the selection is poor, the solution may fall into a locally optimal point or there may be no solution. All of the preceding arguments render those methods unsuitable for solving discrete nonlinear programming problems where the design variables must be chosen from a commercially accessible list. (Saka & Geem, 2013; Singiresu S. Rao, 2019)

The emergence of new optimization approaches based on simulating the behavior of natural phenomena, on the other hand, has extended the field of structural optimization. These are stochastic search methods, commonly known as metaheuristics. Gradient information for the objective function or constraints is not necessary for them. Furthermore, they can tackle combinatorial problems with a high number of options. (Akin & Saka, 2015; X.-She. Yang, 2010).

2.4 Metaheuristic Algorithms

“Metaheuristic” is a combination of two words: meta and heuristic. The word “heuristic” means to find by trial and error while “meta” means higher level (X.-She. Yang, 2010). Therefore, metaheuristic means a higher level to discover. Metaheuristic algorithms are nature-inspired algorithms such as the ant colony optimization (ACO) algorithm that simulate the behavior of natural phenomena. ACO simulates the behavior of ants in finding the shortest path from the food source to their colony. This behavior is studied by scientists and formulated into computerized algorithms that can be applied to and used in solving various types of real-life problems. Other famous examples of metaheuristics are (Xing & Gao, 2014).

2.4.1 Genetic Algorithms (GA)

Genetic Algorithms (GA) is a representation or conceptualization of biological evolution inspired by Charles Darwin's theory of natural selection. It was created by John Holland and his associates in the 1960s and 1970s (Holland, 1975; De Jong, 1975). Holland is credited with being the likely pioneer in the application of crossover, recombination, mutation, and selection in the study of artificial and adaptive systems. The Genetic algorithms represent randomized search techniques designed to mimic the principles of natural selection and genetics. They function by operating on string structures, akin to biological systems, evolving over time based on the principle of survival of the fittest through a randomized yet structured information exchange. Each generation involves the creation of a new set of strings, incorporating components

from the fittest members of the previous set. (Papanicolaou & Papantoniou, 2016)

The key characteristics of genetic algorithms include:

1. The algorithm manipulates a coding of the parameter set rather than the parameters themselves.
2. The search begins with a population of points, as opposed to a single point.
3. Payoff information, not derivatives, guides the algorithm's decision-making.
4. Probabilistic transition rules, rather than deterministic ones, govern the genetic algorithm's operations.

2.4.2 Simulated Annealing (SA)

The Simulated Annealing (SA) algorithm stands out as a highly favored heuristic approach for addressing optimization problems. Kirkpatrick et al. (1983) initially introduced SA, drawing inspiration from the annealing process used in metalworking. This annealing procedure aims to achieve optimal molecular arrangements of metal particles by gradually cooling them after exposure to high heat, minimizing the potential energy of the mass. (Kirkpatrick et al., 1983)

In a broad sense, the SA algorithm follows an iterative process guided by a variable temperature parameter, simulating the annealing transformation observed in metals. (Eren et al., 2017)

Over the past few years, a multitude of innovative optimization algorithms have emerged and been applied to address structural damage

detection challenges through inverse solutions. Despite these advancements, conventional optimization techniques, including particle swarm optimization, simulated annealing (SA), and genetic algorithms, remain widely utilized for detecting damages in structures for example.

2.4.3 Harmony Search (HS)

Since its introduction in 2001, the Harmony Search Algorithm (HSA) has gained global attention, primarily due to its adept combination of exploration and exploitation as well as its user-friendly application. Drawing inspiration from musical performance processes, HSA is composed of three operators: random search, harmony memory considering rule, and pitch adjusting rule. The distinctive ways in which these operators handle exploration and exploitation contribute to the uniqueness of HSA as a metaheuristic algorithm. (Kim, 2016)

In the Harmony Search Algorithm (HSA), the functioning of its operators mirrors the activities of a musician striving to create perfect harmony. Analogous to a musician experimenting with different tones until achieving the ideal blend, HSA's operators undergo various variations. If the outcomes of these variations prove favorable, the algorithm updates its memory by incorporating the successful variation and eliminating the less desirable one. This process aligns with the musician's practice of storing a collection of tones in memory once a perfect harmony is discovered.

During the music production phase, one of three events takes place: (Z. W. Geem et al., 2001)

(i) Selection of one stored tone from the memory (Harmony Memory) among a collection of different tones.

(ii) Selection of any adjacent tone closer to one of the stored tones (pitch adjustment).

(iii) Random selection of any arbitrary tone falling within the standard scope.

(Dubey et al., 2021)

2.4.4 Particle Swarm Optimizer (PSO)

Swarm-based algorithms have risen as potent optimization techniques, drawing inspiration from the collective behaviors observed in social animals. In the context of particle swarm optimization (PSO), the candidate solutions for an optimization problem are conceptualized as a swarm of particles. These particles traverse the parameter space, following trajectories influenced by both their individual and their neighbors' best performances. This collaborative approach is akin to the coordinated movements seen in social animal groups, contributing to the algorithm's efficiency in finding optimal solutions. (Marini & Walczak, 2015)

The Glowworm Swarm Optimization (GSO) algorithm is a relatively new swarm intelligence algorithm that simulates the movement of the glowworms in a swarm based on the distance between them and on a luminescent quantity called luciferin. (Marinaki & Marinakis, 2016)

2.4.5 Cuckoo Search (CS)

The Cuckoo Search (CS) algorithm has found application in various domains of optimization and computational intelligence, demonstrating promising efficiency. In engineering design applications, CS has proven to outperform other algorithms across a spectrum of continuous

optimization problems, including but not limited to spring design and welded beam design. This showcases the algorithm's superior performance and its potential as a valuable tool in addressing complex optimization challenges within the field of engineering design.(X.-S. Yang, 2014)

CS Simulates the behavior of cuckoos in placing their eggs in other birds' nests. Another nonlinear optimization method is the Cuckoo Search (CS) algorithm, which was developed by (Xin-She Yang & Suash Deb, 2009). This algorithm draws inspiration from the aggressive brooding parasitism observed in certain cuckoo species and their egg-laying strategy, as discussed by Davies in 2011. CS exhibits scale-free characteristics in its search movements. (Kumar et al., 2022)

In the actual cuckoo-host species system, cuckoos lay eggs that closely resemble those of the host species in terms of texture, size, and color. This dynamic establishes a system akin to an arms race between cuckoos and host species, where the eggs' similarity plays a crucial role.

2.4.6 Colliding Bodies Optimization (CBO)

Colliding Bodies Optimization (CBO) is an evolutionary algorithm based on population dynamics, employing an analogy of the laws governing collisions between objects, as introduced by Kaveh and Mahdavi in 2014. CBO has demonstrated effective results across various constrained and unconstrained benchmark functions, as well as in solving engineering single-objective problems (Kaveh & Mahdavi, 2015). The algorithm is characterized by its simplicity, as it requires no memory and involves no parameter tuning.(Kaveh et al., 2021; Kaveh & Mahdavi, 2019)

The application of multi-objective optimization algorithms to structural optimization problems has engrossed the interest of many researchers. In the majority of truss structural design applications, the fitness function typically centers around a single evaluation criterion. These problems are oriented towards minimizing a specific criterion, such as weight while adhering to constraints related to factors like stresses, displacements, and dynamic considerations. However, real-world optimization problems often encompass multiple objectives that need to be optimized simultaneously. These may include, among others, minimizing weight or cost, maximizing stiffness, minimizing displacement at specific structural points, maximizing structural strain energy, and maximizing the natural frequency of free vibration. In such cases, the optimization process involves navigating the trade-offs and finding solutions that offer a balanced performance across these diverse objectives.

2.4.7 The Artificial Bee Colony (ABC)

The Artificial Bee Colony (ABC) algorithm is a recently developed optimization technique that emulates the intelligent foraging behavior of honey bees. A group of honey bees is referred to as a swarm, and through social cooperation, they effectively accomplish tasks. Within the ABC algorithm, three types of bees play distinct roles: employed bees, onlooker bees, and scout bees. Employed bees search for food around the sources stored in their memory and share this information with onlooker bees. Onlooker bees, in turn, choose promising food sources from those discovered by the employed bees. The likelihood of selection by onlooker bees is influenced by the quality (fitness) of the food source, with higher-quality sources having a greater chance of being chosen. Scout bees

emerge from a subset of employed bees, abandoning their current food sources to explore new ones.(G. Zhu and S. Kwong, 2010)

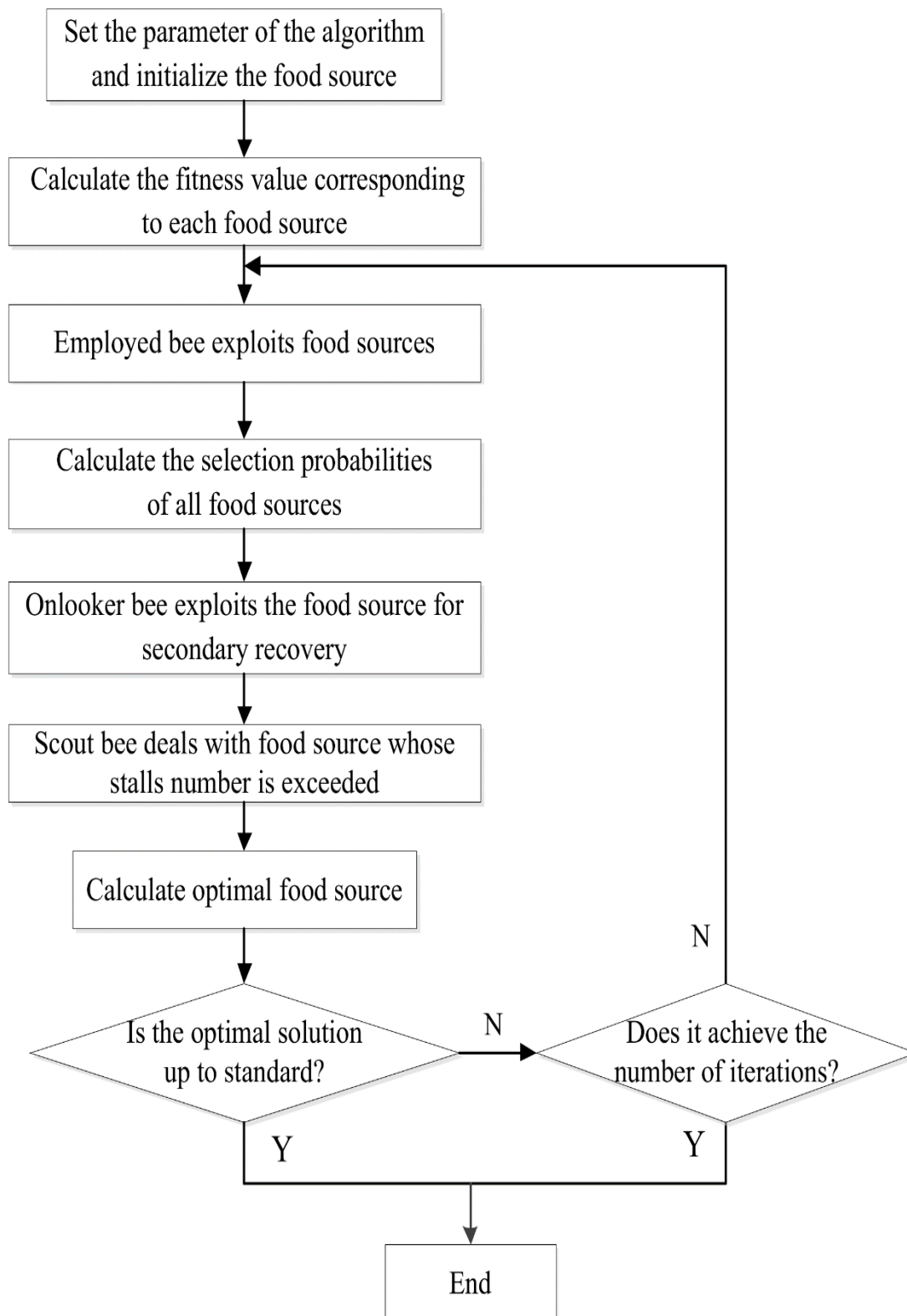


Figure 2.3 The flowchart of the basic ABC algorithm (J. Yang & Peng, 2018)

2.4.8 Teaching-Learning-Based Optimization (TLBO)

Teaching-Learning-Based Optimization (TLBO) is a metaheuristic algorithm introduced by Rao et al. in 2011 (R. V. Rao, 2011)[17]. This method draws inspiration from the teaching-learning process observed in a classroom, simulating the impact of a teacher on learners. With its notable benefits such as rapid convergence, the absence of algorithm-specific parameters, and ease of implementation, TLBO has gained popularity as an optimization algorithm. Its versatility is evident in its successful application to real-world problems across various fields.(Wu et al., 2022)

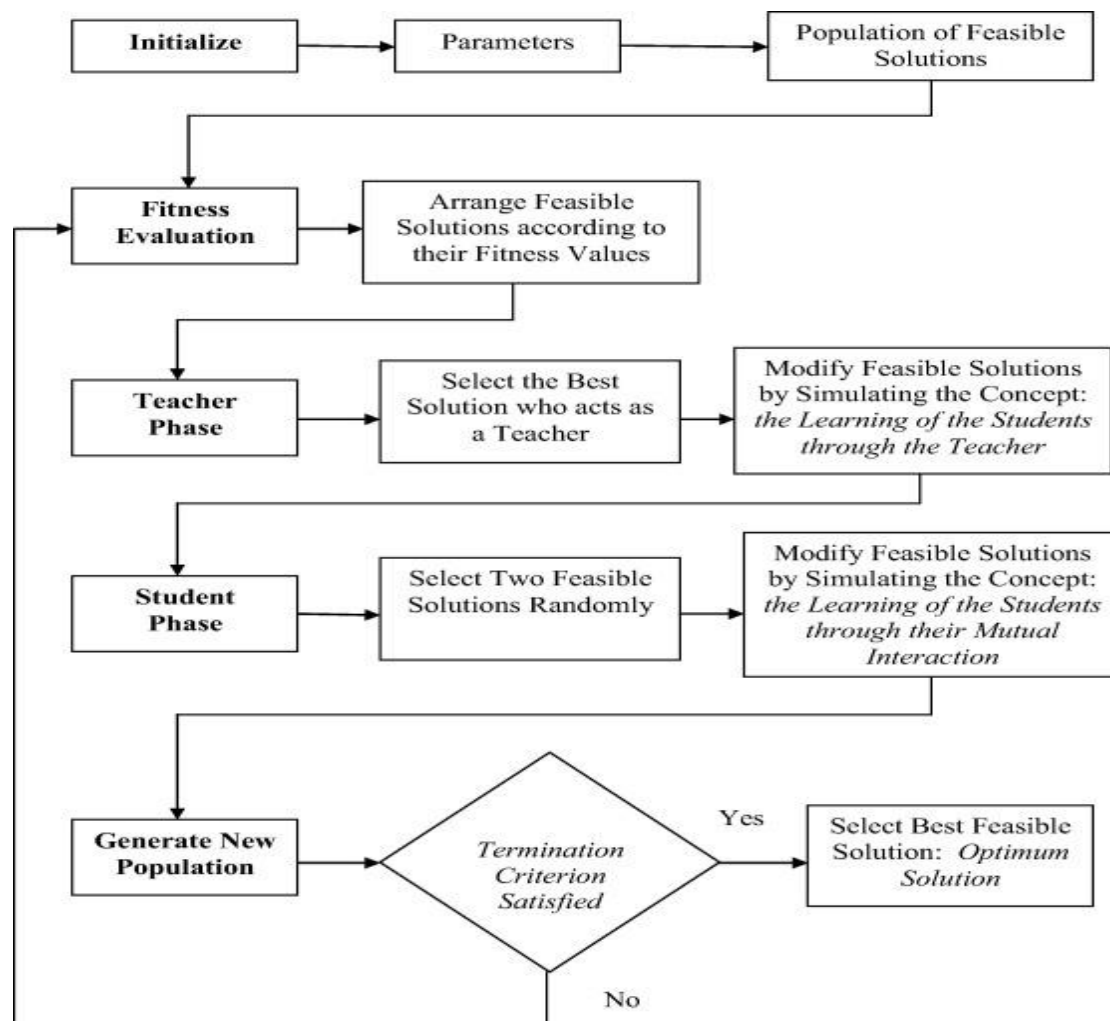


Figure 2.4 The flowchart of the TLBO algorithm (Singh Gill et al., 2019)

2.4.9 Differential Evolution (DE)

DE, or Differential Evolution, is a population-based optimization algorithm that belongs to the category of evolutionary algorithms. It was introduced by Rainer Storn and Kenneth Price in 1997. Differential Evolution is designed for solving optimization problems, where the goal is to find the minimum or maximum of a given objective function. Differential Evolution is known for its simplicity, efficiency, and effectiveness in solving a wide range of optimization problems, especially in continuous and high-dimensional search spaces.(Lu et al., 2023)

Metaheuristic algorithms can deal with unconstrained optimization problems. To make them capable of handling constrained optimization problems, the constrained optimization problem needs to be transferred into an unconstrained problem. One of the ways to do that is to apply a penalty function method. In this method, the following equation is applied:

$$\emptyset = Q \left\{ 1 + \alpha \left(\sum_{i=1}^n g_i \right) \right\} \quad (2.6)$$

where the $\emptyset$ is the unconstrained or penalized objective function, Q is the objective function, α here is a penalty constant taken preferably as 1, g_i is the value of the i^{th} normalized constraint and n is the number of constraints (Aydođdu et al., 2016; Lute et al., 2011; Saka et al., 2013).

Metaheuristic algorithms also require normalization of the design constraints before seeking a solution. A normalized constraint means that the constraint equation is written in a form less than or equal to zero, as shown in the following example equation:

$$g(x) > h(x) \quad \text{before normalization} \quad (2.7)$$

$$\frac{h(x)}{g(x)} - 1 \leq 0 \quad \text{after normalization} \quad (2.8)$$

After the application of the normalization and penalty function method, the optimization problem becomes suitable to be handled by the metaheuristic algorithms.

Metaheuristic algorithms can solve complex and large types of constrained design optimization problems with discrete variables. This has been achieved with various types of structural design problems such as steel and reinforced concrete structures (Akin & Saka, 2015; Hasançebi et al., 2009).

2.5 Literature Review for the Optimum Design of Columns

The process of designing a reinforced concrete column to withstand both axial load and bending moment is an iterative process that entails intricate computations. Numerous factors, including load eccentricity, column cross-sectional dimensions, steel percentage, neutral axis location, and steel and concrete grades, impact the design and necessitate the utilization of interaction diagrams.

Due to the particular nature of the column design challenge, the subject of column optimization received little attention in the literature as compared to building structures in previous decades. Column design and analysis take a long time and a lot of effort, making the trial-and-error procedure in the design stage laborious. On the other hand, the typical optimization approaches available at the time were insufficient. This is owing to the

nature of the challenge, which included a high number of discrete variables that had to be chosen from design tables (Aydodu & Saka, 2012).

The number of column design optimization examples has increased dramatically in recent years. Even though there is already sophisticated software for column analysis, the introduction of stochastic optimization approaches, which are metaheuristics, has resulted in significant growth in the field of optimization.

This section provides a quick overview of the design optimization of column architectures reported in the literature.

The research conducted by Mohammad and Seyan (2016) for instance, delves into achieving the most cost-effective design for reinforced concrete rectangular columns under axial compression force and biaxial bending moments along the x and y axes. The optimization procedure employed the Generalized Reduced Gradient (GRG) technique, integrated into the Excel Solver add-in tool. The choice of the GRG method is based on its recognized robustness and efficiency in addressing diverse engineering challenges, as evidenced by numerous studies in the literature. GRG is widely acknowledged as one of the top deterministic local optimization methods. The models formulated for the design of reinforced concrete columns, along with the associated constraints, were based on Eurocode 2 (EC2) provisions. The design variables included the cross-sectional dimensions (width and depth) and the area of reinforcing steel. Notably, the position and orientation of the neutral axis were also treated as design variables. This approach aimed to avoid the complex and time-consuming task of solving highly nonlinear simultaneous equations, which may not necessarily yield convergent solutions. Various

design parameters, such as material cost ratios and depth-to-width ratios, were systematically examined. Consequently, optimal design charts were developed to encompass a broad range of practical combinations of axial compression forces and eccentricities in both the x and y directions. The study has reached the following conclusions: The optimal cost of the column section exhibits an increase within the cost ratio range C_s/C_c , specifically at values of 5, 10, and 20 while maintaining the same h/b ratio. Additionally, the impact of the cost ratio C_s/C_c becomes more pronounced under larger loads and heightened eccentricities.

Rafiq & Southcombe (1998), introduced a pioneering approach to the optimal design and detailing of reinforced concrete biaxial columns using genetic algorithms (GAs) at the time of its publication. The methodology illustrates how GAs perform a global search to determine the optimal sizes and arrangements of reinforcement bars for a given set of design specifications, encompassing section size, axial load, and bending about both axes of the column. The aim is to meet the maximum bending capacity of both axes of the column section while minimizing the area of reinforcements, resulting in an economical design. The detailing of reinforcement bar arrangements within the column section adheres to the requirements of the British Standard (BS 8110) to ensure satisfaction of both the ultimate limit state (ULS) and buildability (ease of construction) criteria. The research draws the following general conclusions from the discussions and details presented in the paper:

- The developed design method minimizes the requirements for reinforcement bars and optimizes the resistance of biaxially loaded columns.

- Evolutionary optimization techniques prove effective in achieving superior, cost-effective, and safe designs. Notably, these techniques are applicable on personal computer platforms with ease.

Bekdaş & Nigdeli (2016), explores the optimal design of reinforced concrete columns through the application of an educational metaheuristic algorithm known as teaching-learning-based optimization (TLBO). The methodology incorporates considerations for the slenderness of the columns, utilizing a straightforward approach outlined in the ACI 318 design code. This approach defines the factored design flexural moments based on the buckling load and axial load of the columns. The design variables encompass the cross-sectional dimensions of the column and detailed reinforcement design, with the optimization objective focused on minimizing the maximum material cost of the column.

In contrast to other metaheuristic algorithms, TLBO does not rely on a probability parameter to determine the optimization type (global or local search). Instead, the TLBO involves two phases: the teacher phase for global search and the learner phase for local search, sequentially applied to identify the best design variables. The proposed approach proves effective for addressing structural optimization problems.

Pires & Silva (2021), introduces the development of a computational tool designed for optimizing slender reinforced concrete columns under biaxial bending, incorporating rigorous considerations of material and geometric nonlinearities. The optimization process employs genetic algorithms to identify the optimal cross-sectional dimensions and the most efficient distribution and quantity of reinforcement. The primary objective is to minimize the cost of the column, subject to constraints related to strength, stability, feasibility, and regulatory requirements. The

analysis is specifically applied to rectangular cross-sections of columns with a constant and symmetrical distribution of reinforcement.

Results obtained from the developed software demonstrate proper convergence to an optimal solution or a solution near the optimum within a significantly shorter time frame compared to considering all solutions within the established domain. Additionally, the study analyses the impact of slenderness on the optimal design of reinforced concrete columns. Findings reveal that as the slenderness ratio increases, the cost of the column rises, with steel emerging as the predominant material in establishing structural equilibrium.

Chapter Three

Optimum Design of Composite Column Filled with Concrete

3 Optimum Design of Composite Steel Column Filled with Concrete

3.1 Introduction

Any structural element that comprises two or more members that are made of a combination of different materials that act together as a single element is called a composite structure. The most common example of composite structures is the steel beams that support a reinforced concrete slab. Composite Columns are one of those so-called composite structures as well.

Composite Columns are considered composite members, a type of structure that is not yet widely used in the field of structural buildings. On the other hand, it is easy to construct and leads to cost and material savings, especially in high-rise buildings. There are two different types of composite columns, the first one is a filled steel column with concrete as shown in Figure 3.1 and the other one steel column covered with concrete as shown in Figure 3.2.

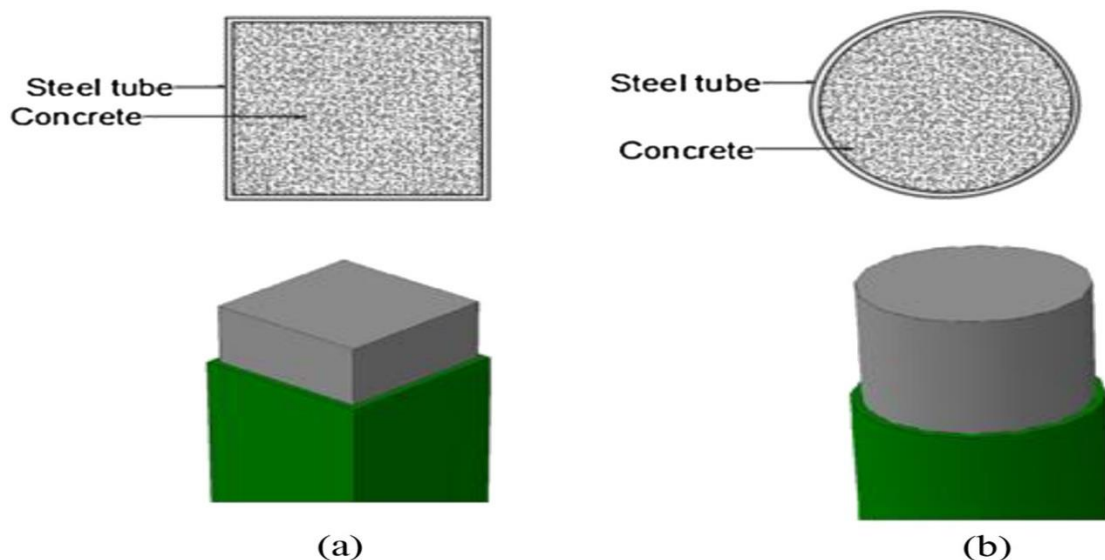


Figure 3.1 Composite-filled steel column cross-section (M. Fahim, 2022)

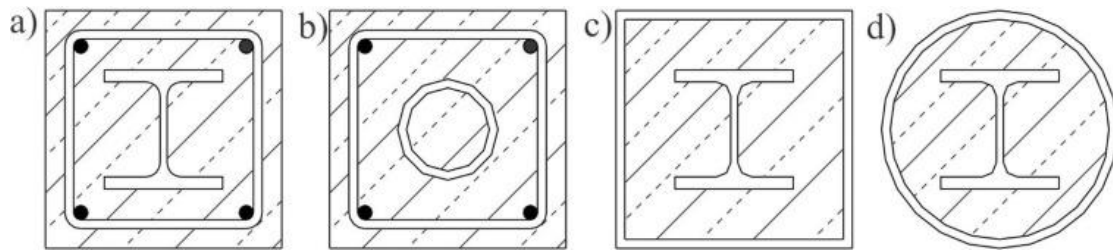


Figure 3.2 Composite Covered steel column cross-section (M. Fahim, 2022)

3.2 Composite Column Model

A single-storey two-bay building was selected at the beginning as a start to check if the methodology that is to be described in Chapter 4 would work properly or not.

The building is a two-way reinforced slab with 10m length with 10m width. It consists of a 200mm reinforced concrete slab and three steel I-beams spaced at 5 m intervals in each direction. The building was created without any walls or parapets. In addition, the building also consists of 5m height fixed nine composite tube steel columns filled with concrete supporting the beams.

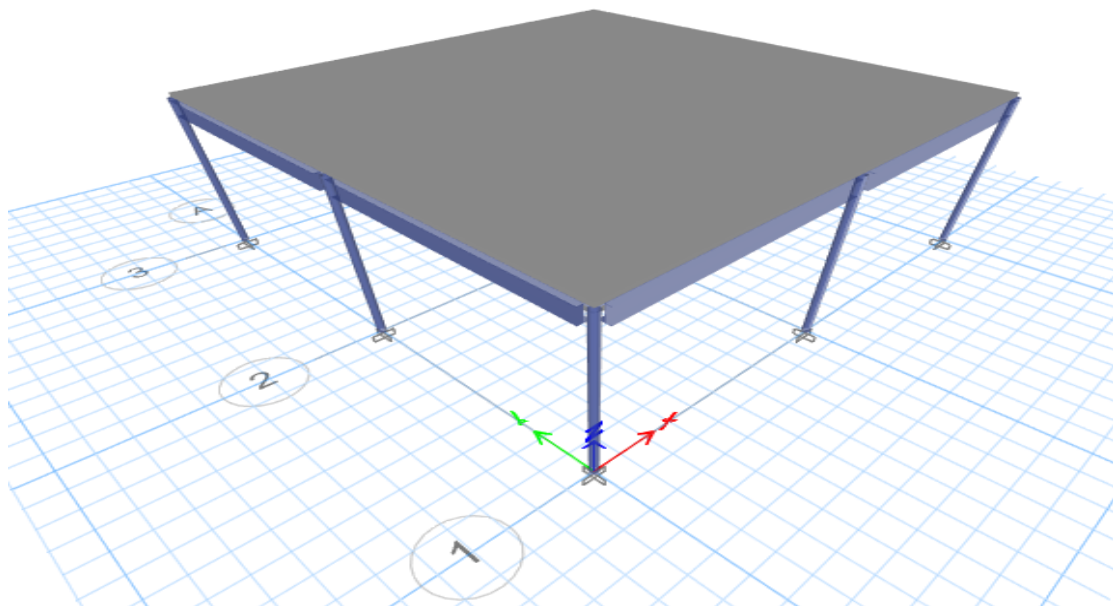


Figure 3.3 Single-storey building model 3D view

The building is modeled in ETABS by selecting the grid dimensions then, the structure was drawn after that, the materials and sections were defined that will be the initial variables, and the sections and the supports were assigned to the frame structure as shown in Figure 3.3.

Each beam in the model has two spans and is subjected to three composite columns, which means that each span is 5m as shown in Figure 3.4. In the optimization problem, the outer beams, inner beams, outer columns, and inner columns so that every symmetrical beam and column yields the same section, refer to Table 3.1.

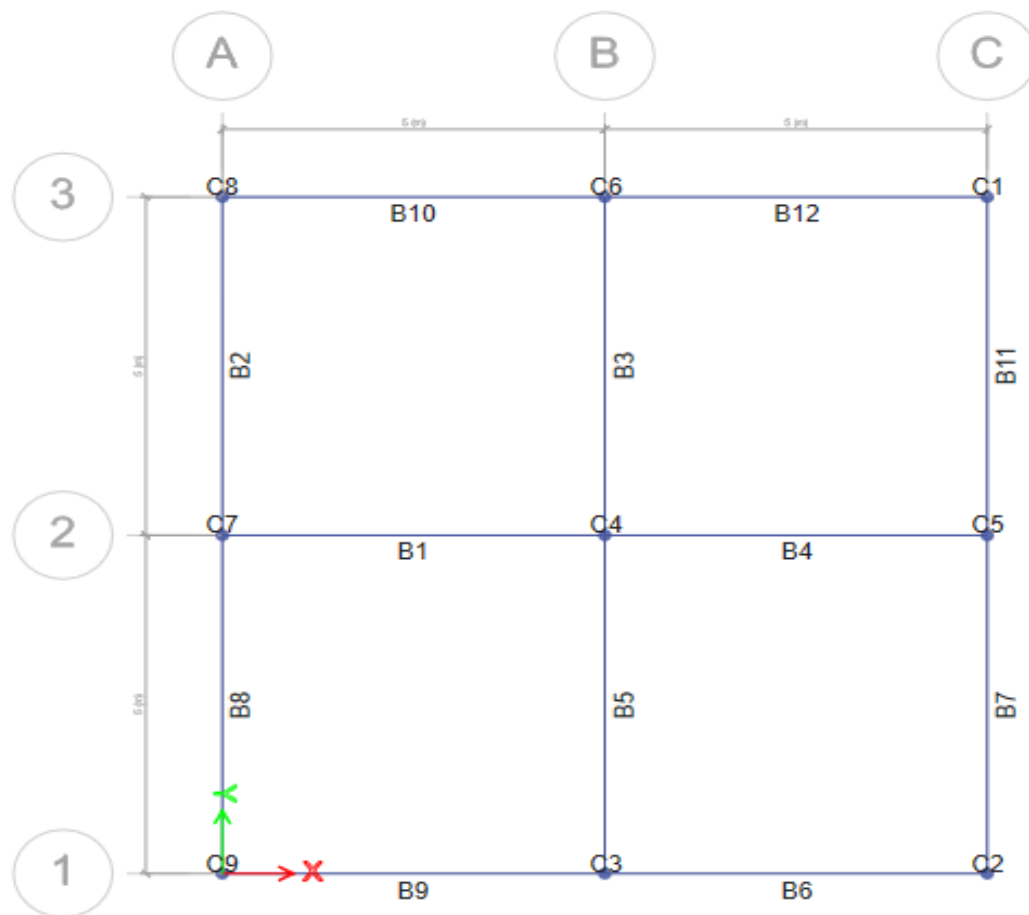


Figure 3.4 Single-storey building model top view

Table 3.1 Details of beams and columns members and grouping for single-storey building.

Group Name	Group Description	Grouped Members
IC	Inner Column	C4
OC	Outer Columns	C1-3, C5-9
IB	Inner Beam	B1, B3-5
OB	Outer Beam	B2, B6-12

But in the end, the structure load transferred to the beams and columns was increased, so a three-storey building with 20m length and 20m width was modeled. It consists of a 300mm reinforced concrete slab and five steel I-beams spaced at 5 m intervals in each direction. The building was created with glass walls and 1.25m glass parapets and any inside dividers will be made of gypsum.

In addition, the building also consists of three stories 5m height each with twenty-five composite tube steel columns filled with concrete supporting the beams.

The building is modeled in ETABS by selecting the grid dimensions then, the structure was drawn after that, the materials and sections were defined that will be the initial variables, and the sections and the supports were assigned to the frame structure as shown in Figure 3.5.

Each beam in the model has four spans and is subjected to five composite columns, which means each span is 5m as shown in Figure 3.6. In the optimization problem, the outer beams and inner beams also outer columns and inner columns for each floor separately so that every symmetrical beam and column yields the same section, refer to Table 3.2.

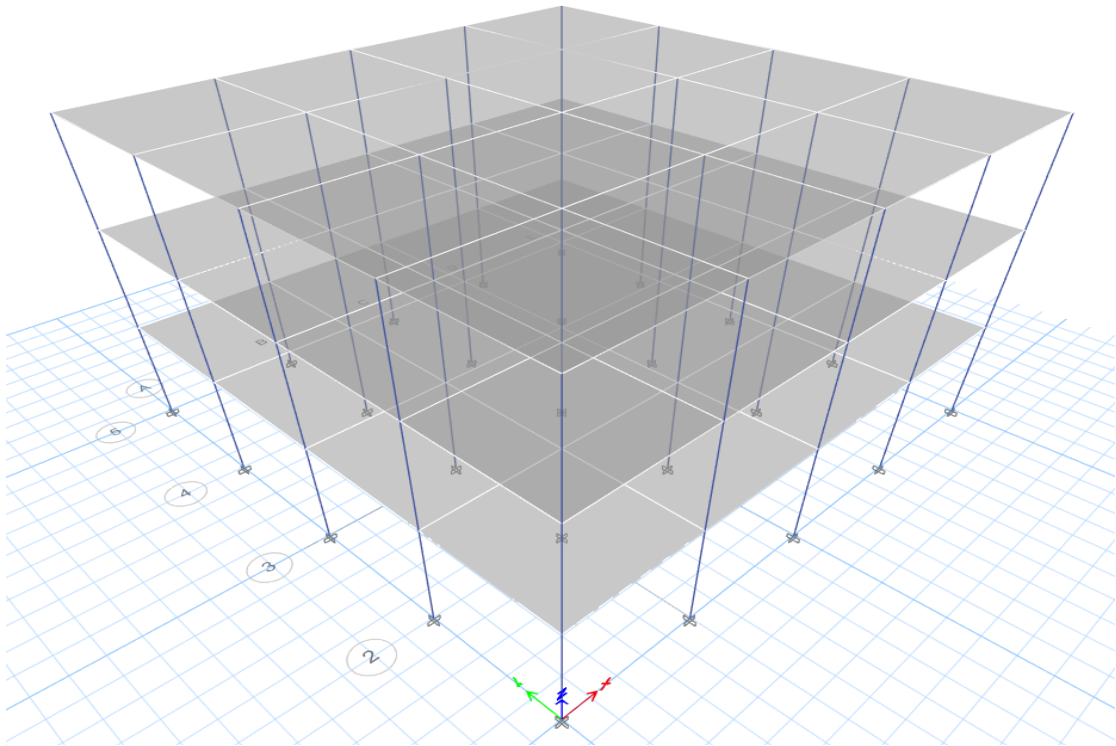


Figure 3.5 Three-storey building model 3D view

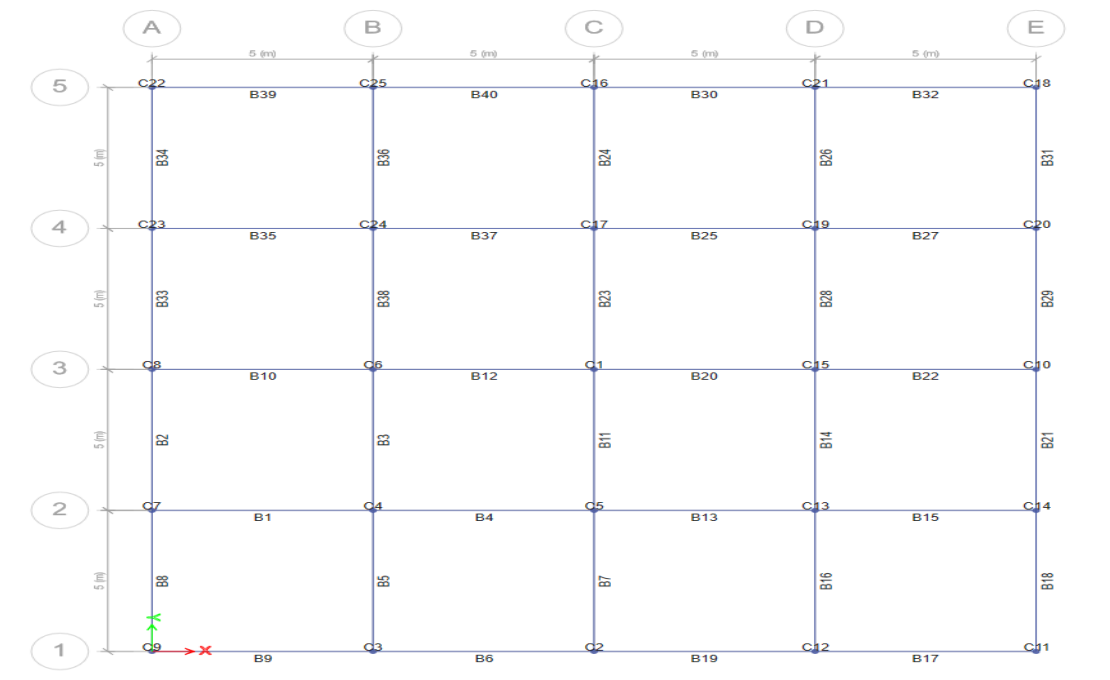


Figure 3.6 Storey 3 of a three-storey building model top view

Table 3.2 Details of beams and columns members and grouping for a three-storey building

Group Name	Group Description	Storey Level	Grouped Members
IB	Inner Beam	1	B1, B3-5, B7, B10-16, B20, B22-28, B35-38
		2	B1, B3-5, B7, B10-16, B20, B22-28, B35-38
		3	B1, B3-5, B7, B10-16, B20, B22-28, B35-38
OB	Outer Beam	1	B2, B6, B8-9, B17-19, B21, B29-34, B39-40
		2	B2, B6, B8-9, B17-19, B21, B29-34, B39-40
		3	B2, B6, B8-9, B17-19, B21, B29-34, B39-40
IC1	Inner Column for the first storey	1	C1, C4-6, C13, C15, C17, C19, C24
OC1	Outer Columns for the first storey	1	C2-3, C7-12, C14, C16, C18, C20-23, C25
IC2	Inner Column for the second storey	2	C1, C4-6, C13, C15, C17, C19, C24
OC2	Outer Columns for the second	2	C2-3, C7-12, C14, C16, C18, C20-23, C25

	storey		
IC3	Inner Column for the third storey	3	C1, C4-6, C13, C15, C17, C19, C24
OC3	Outer Columns for the third storey	3	C2-3, C7-12, C14, C16, C18, C20-23, C25

This structure is subjected to the following loads:

1. Dead loads

The dead load for the structure represents the superstructure's self-weight. The self-weight of the structure is calculated automatically by the software.

2. Live loads

The live load refers to the weight or force that a building or structure experiences from the people, furniture, and other movable objects inside it. It is an important factor to consider in structural design to ensure the safety and stability of the building.

The live load on this building is 3 KN/m² according to ASCE.

3. Superimposed dead load

It is one of the dead loads in structural engineering, superimposed load is the additional weight that is added to a structure over time, such as partitions, ceilings, flooring, equipment, or machinery. This load is important to consider when designing a structure, as it affects the load-bearing capacity and stability of the building.

The superimposed dead load on this building is 4 KN/m² according to ASCE.

4. Wind loads

Wind load is the force exerted on a structure by the movement of air. It is an important consideration in structural engineering, as it can have a significant impact on the design and safety of a structure.

The building wind load was considered from x and y directions and assumed its speed of 120 MPH as the wind load. So, the ETABS software will calculate its pressure automatically.

5. Wall loads

Wall loads are the forces that act on a wall due to the weight of the structure above it, the wind pressure on it, and the weight of itself. Wall loads can be classified into two types: load-bearing walls and non-load-bearing walls. Load-bearing walls are walls that support the weight of the roof and the upper floors of a building and transfer it to the foundation. Non-load-bearing walls are walls that do not support any structural load and are mainly used for partitioning or decoration.

The wall in the building was considered as non-load bearing walls and the outer walls were glass with a weight of 2.36 KN/m^2 and the inner walls were made of gypsum boards and their weight was considered in the superimposed load.

The loading combinations that are used for these load cases were generated from ETABS directly as per the following, see Table 3.3.

Table 3.3 Details of load combination arrangement

Combo. Name	DL	LL	SID	Wx	Wy	Wall
DCmpC1	1.4	0	0	0	0	1.4

DCmpC2	1.2	0	0	0	0	1.2
DCmpD1	1	0	1	0	0	1
DCmpD2	1	1	1	0	0	1
DCmpS1	1.4	0	1.4	0	0	1.4
DCmpS2	1.2	1.6	1.2	0	0	1.2
DConS1	1.4	0	1.4	0	0	1.4
DConS2	1.2	1.6	1.2	0	0	1.2
DConS3	1.2	1	1.2	1	0	1.2
DConS4	1.2	1	1.2	-1	0	1.2
DConS5	1.2	1	1.2	0	1	1.2
DConS6	1.2	1	1.2	0	-1	1.2
DConS7	0.9	0	0.9	1	0	0.9
DConS8	0.9	0	0.9	-1	0	0.9
DConS9	0.9	0	0.9	0	1	0.9
DConS10	0.9	0	0.9	0	-1	0.9
DStlD1	1	0	1	0	0	1
DStlD2	1	1	1	0	0	1
DStlS1	1.4	0	1.4	0	0	1.4
DStlS2	1.2	1.6	1.2	0	0	1.2
DStlS3	1.2	1	1.2	1	0	1.2
DStlS4	1.2	1	1.2	-1	0	1.2
DStlS5	1.2	1	1.2	0	1	1.2

DStlS6	1.2	1	1.2	0	-1	1.2
DStlS7	0.9	0	0.9	1	0	0.9
DStlS8	0.9	0	0.9	-1	0	0.9
DStlS9	0.9	0	0.9	0	1	0.9
DStlS10	0.9	0	0.9	0	-1	0.9

Where:

DL is the dead load.

LL is the Live load.

SID is the superimposed dead load.

W_x is the wind load in the x direction.

W_y is the wind load in the y direction.

Wall is the wall load.

In the end, the duplicated load combinations were removed, and finalized with 15 different load combinations.

3.3 Design Optimization of the Composite Columns

This example aims to find the optimum design of the structure by varying the cross-section of the composite columns, the cross-section of the steel I-beams, and the concrete-filled strength (f'_c).

Following is the mathematical model of the optimization problem designed according to AISC 360-16 and ACI CODE-318-19(22):

3.3.1 Objective Function

The objective function is to minimize the total weight of the structure elements as follows:

$$W_T = W_S + W_C \quad (3.1)$$

where W_T is the total weight of both steel I-beams and composite columns (kN), W_S is the weight of the steel I-beams and the steel tube in composite columns (kN) and W_C is the weight of the concrete filled in the composite columns (kN).

The total weight of steel is calculated from:

$$W_S = \rho \sum_{i=1}^{N_m} L_i \sum_{j=1}^{N_g} A_j \quad (3.2)$$

where N_m is the total number of steel members, L_i is the length of the i^{th} steel member (mm), N_g is the number of members, A_j is the area of the i^{th} steel member in the j^{th} section (mm²) and ρ is the density of steel (76.9426×10^{-9} kN/mm³).

The total weight of concrete is equal to:

$$W_C = \rho \sum_{i=1}^{N_m} L_i \sum_{j=1}^{N_g} A_j \quad (3.3)$$

where N_m is the total number of composite columns, L_i is the length of the i^{th} column (mm), N_g is the number of columns, A_j is the area of the i^{th} concrete cross-section in the j^{th} column (mm²) and ρ is the density of concrete (23.5631×10^{-9} kN/mm³).

3.3.2 Design Variables

The design variables in this example are the beam's steel members section, the composite columns made of steel tubes filled with concrete,

and the concrete strength. The steel members are the W-sections for the beams in Table 3.4 and HSS (hollow structural sections) selected from the available lists in ETABS from the AISC 15M list and the 105 section for the HSS was created in Table 3.5. Two hundred eighty-three W-sections and two hundred thirty-three HSS were selected. The Concrete strength (f'_c) is arranged from the interval [20 Mpa, 100 Mpa] with an increment of 5 Mpa.

Table 3.4 W-sections

Sec. No.	Sec. Name	Cross-section area (mm)	Depth (mm)	Width (mm)	Web thickness (mm)	Flange thickness (mm)
1	W150X13	1630	148	100	4.32	4.95
2	W150X13.5	1730	150	100	4.32	5.46
3	W200X15	1910	200	100	4.32	5.21
4	W250X17.9	2280	251	101	4.83	5.33
5	W150X18	2290	153	102	5.84	7.11
6	W100X19.3	2470	106	103	7.11	8.76
7	W200X19.3	2480	203	102	5.84	6.48
8	W310X21	2680	302	101	5.08	5.72
9	W250X22.3	2850	254	102	5.84	6.86
10	W200X22.5	2860	206	102	6.22	8
11	W150X22.5	2860	152	152	5.84	6.6
12	W310X23.8	3040	305	101	5.59	6.73
13	W130X23.8	3040	127	127	6.1	9.14
14	W150X24	3060	160	102	6.6	10.3
15	W250X25.3	3220	257	102	6.1	8.38
16	W200X26.6	3390	207	133	5.84	8.38
17	W130X28.1	3590	131	128	6.86	10.9
18	W310X28.3	3590	310	102	5.97	8.89
19	W250X28.4	3630	259	102	6.35	10
20	W150X29.8	3790	157	153	6.6	9.27
21	W200X31.3	3970	210	134	6.35	10.2

22	W310X32.7	4180	312	102	6.6	10.8
23	W250X32.7	4190	259	146	6.1	9.14
24	W360X32.9	4190	348	127	5.84	8.51
25	W200X35.9	4570	201	165	6.22	10.2
26	W150X37.1	4740	162	154	8.13	11.6
27	W250X38.5	4910	262	147	6.6	11.2
28	W310X38.7	4940	310	165	5.84	9.65
29	W410X38.8	4950	399	140	6.35	8.76
30	W360X39	4960	353	128	6.48	10.7
31	W200X41.7	5320	205	166	7.24	11.8
32	W310X44.5	5670	312	166	6.6	11.2
33	W250X44.8	5700	267	148	7.62	13
34	W360X44	5710	351	171	6.86	9.78
35	W410X46.1	5890	404	140	6.99	11.2
36	W200X46.1	5890	203	203	7.24	11
37	W250X49.1	6260	247	202	7.37	11
38	W360X51	6450	356	171	7.24	11.6
39	W460X52	6650	450	152	7.62	10.8
40	W310X52	6650	318	167	7.62	13.2
41	W200X52	6650	206	204	7.87	12.6
42	W410X53	6840	404	178	7.49	10.9
43	W360X57.8	7230	358	172	7.87	13.1
44	W250X58	7420	252	203	8	13.5
45	W200X59	7550	210	205	9.14	14.2
46	W310X60	7550	302	203	7.49	13.1
47	W460X60	7610	455	153	8	13.3
48	W410X60	7610	406	178	7.75	12.8
49	W360X64	8130	348	203	7.75	13.5
50	W530X66	8390	526	165	8.89	11.4
51	W310X67	8450	307	204	8.51	14.6
52	W410X67	8580	409	179	8.76	14.4
53	W250X67	8580	257	204	8.89	15.7
54	W460X68	8710	460	154	9.14	15.4
55	W200X71	9100	216	206	10.2	17.4
56	W530X72	9100	523	207	8.89	10.9

57	W360X72	9100	351	204	8.64	15.1
58	W250X73	9290	254	254	8.64	14.2
59	W310X74	9420	310	205	9.4	16.3
60	W530X74	9480	528	166	9.65	13.6
61	W460X74	9480	457	191	9.02	14.5
62	W410X75	9480	414	180	9.65	16
63	W360X79	10100	353	205	9.4	16.8
64	W310X79	10100	307	254	8.76	14.6
65	W250X80	10200	257	254	9.4	15.6
66	W610X82	10500	599	178	10	12.8
67	W530X82	10500	528	209	9.53	13.3
68	W460X82	10500	460	191	9.91	16
69	W530X85	10800	536	167	10.3	16.5
70	W410X85	10800	417	181	10.9	18.2
71	W310X86	11000	310	254	9.14	16.3
72	W200X86	11000	222	209	13	20.6
73	W460X89	11400	462	192	10.5	17.7
74	W250X89	11400	259	257	10.7	17.3
75	W360X91	11500	353	254	9.53	16.4
76	W610X92	11700	602	179	10.9	15
77	W530X92	11800	533	209	10.2	15.6
78	W460X97	12300	467	193	11.4	19.1
79	W310X97	12300	307	305	9.91	15.4
80	W410X100	12600	414	259	10	16.9
81	W200X100	12700	229	210	14.5	23.7
82	W250X101	12800	264	257	11.9	19.6
83	W530X101	12900	536	210	10.9	17.4
84	W360X101	12900	356	254	10.5	18.3
85	W610X101	13000	602	228	10.5	14.9
86	W460X106	13500	470	194	12.6	20.6
87	W310X107	13600	312	305	10.9	17
88	W530X109	13900	538	211	11.6	18.8
89	W360X110	14100	361	257	11.4	19.9
90	W460X113	14400	462	279	10.8	17.3
91	W610X113	14500	607	228	11.2	17.3

92	W410X114	14600	419	262	11.6	19.3
93	W250X115	14600	269	259	13.5	22.1
94	W310X117	15000	315	307	11.9	18.7
95	W360X122	15500	363	257	13	21.7
96	W530X123	15700	544	212	13.1	21.2
97	W690X125	15900	678	254	11.7	16.3
98	W610X125	15900	612	229	11.9	19.6
99	W460X128	16300	467	282	12.2	19.6
100	W310X129	16500	318	307	13.1	20.6
101	W250X131	16800	274	262	15.4	25.1
102	W410X132	16900	427	264	13.3	22.2
103	W760X134	17000	749	264	11.9	15.5
104	W360X134	17100	356	368	11.2	18
105	W530X138	17600	549	214	14.7	23.6
106	W690X140	17800	683	254	12.4	18.9
107	W610X140	17900	617	230	13.1	22.2
108	W310X143	18200	323	310	14	22.9
109	W460X144	18400	472	282	13.6	22.1
110	W760X147	18700	754	267	13.2	17
111	W360X147	18800	361	371	12.3	19.8
112	W250X149	18900	282	262	17.3	28.4
113	W410X149	19000	432	264	14.9	25
114	W530X150	19200	544	312	12.7	20.3
115	W690X152	19400	688	254	13.1	21.1
116	W610X153	19500	622	229	14	24.9
117	W610X155	19800	612	325	12.7	19.1
118	W460X158	20100	475	284	15	23.9
119	W310X158	20100	328	310	15.5	25.1
120	W760X161	20500	757	267	13.8	19.3
121	W360X162	20600	363	371	13.3	21.8
122	W530X165	21000	546	312	14	22.2
123	W250X167	21200	290	264	19.2	31.8
124	W690X170	21700	693	257	14.5	23.6
125	W760X173	22100	762	267	14.4	21.6
126	W610X174	22200	617	325	14	21.6

127	W840X176	22400	836	292	14	18.8
128	W460X177	22600	483	287	16.6	26.9
129	W310X179	22700	333	312	18	28.2
130	W360X179	22800	368	373	15	23.9
131	W530X182	23200	551	315	15.2	24.4
132	W760X185	23500	767	267	14.9	23.6
133	W690X192	24400	701	254	15.5	27.9
134	W840X193	24700	841	292	14.7	21.7
135	W460X193	24700	490	284	17	30.5
136	W610X195	24900	622	328	15.4	24.4
137	W760X196	25000	770	267	15.6	25.4
138	W530X196	25000	554	315	16.5	26.4
139	W360X196	25000	373	373	16.4	26.2
140	W920X201	25700	904	305	15.2	20.1
141	W310X202	25700	340	315	20.1	31.8
142	W840X210	26800	846	292	15.4	24.4
143	W460X213	27100	495	284	18.5	33.5
144	W360X216	27500	376	394	17.3	27.7
145	W610X217	27700	627	328	16.5	27.7
146	W690X217	27900	696	356	15.4	24.8
147	W530X219	27900	561	318	18.3	29.2
148	W760X220	28100	780	267	16.5	30
149	W1000X222	28300	970	300	16	21.1
150	W920X223	28600	912	305	15.9	23.9
151	W310X226	28800	348	318	22.1	35.6
152	W840X226	29000	851	295	16.1	26.9
153	W460X235	29900	500	287	20.6	36.6
154	W360X237	30100	381	396	18.9	30.2
155	W920X238	30300	914	305	16.5	25.9
156	W690X240	30700	701	356	16.8	27.4
157	W610X241	30800	635	330	17.9	31
158	W530X248	31500	572	315	19.1	34.5
159	W1000X249	31800	980	300	16.5	26.2
160	W840X251	31900	859	292	17	31
161	W920X253	32300	919	305	17.3	27.9

162	W310X253	32300	356	320	24.4	39.6
163	W760X257	32800	772	381	16.6	27.2
164	W460X260	33200	508	290	22.6	40.4
165	W610X262	33400	640	328	19.1	34
166	W360X262	33400	386	399	21.1	33.3
167	W690X265	33900	706	358	18.4	30.2
168	W1000X272	34400	991	300	16.5	30.5
169	W920X271	34600	922	307	18.4	30
170	W530X272	34600	577	318	21.1	37.6
171	W310X283	36100	366	323	26.9	44.2
172	W760X284	36200	780	381	18	30.2
173	W460X286	36300	518	292	24.4	44.5
174	W610X285	36500	648	330	20.6	37.1
175	W360X287	36600	394	399	22.6	36.6
176	W920X289	36800	927	307	19.4	32
177	W690X289	36800	714	356	19.1	34
178	W1000X296	37900	983	401	16.5	27.2
179	W840X299	38100	856	399	18.2	29.2
180	W530X300	38300	584	320	23.1	41.4
181	W610X307	39200	653	330	22.1	39.9
182	W920X313	39900	932	310	21.1	34.5
183	W310X313	39900	373	325	30	48.3
184	W360X314	40000	399	401	24.9	39.6
185	W1000X314	40100	1000	300	19.1	36.1
186	W760X314	40200	785	384	19.7	33.5
187	W460X315	40200	526	295	26.9	48.5
188	W1000X321	41000	991	401	16.5	31
189	W690X323	41200	721	358	21.1	38.1
190	W840X329	42100	861	401	19.7	32.5
191	W530X332	42900	594	323	25.4	45.5
192	W610X341	43400	660	333	24.4	43.9
193	W310X342	43700	384	328	32.8	52.6
194	W1100X343	43700	1090	401	18	31
195	W920X345	43900	943	307	22.1	39.9
196	W920X344	44000	927	419	19.3	32

197	W360X347	44200	406	404	27.2	43.7
198	W460X349	44300	536	297	29.5	53.6
199	W1000X350	44600	1010	302	21.1	40.1
200	W760X350	44700	795	384	21.1	38.1
201	W690X350	44800	729	361	23.1	40.9
202	W840X359	45900	869	404	21.1	35.6
203	W920X368	46800	932	419	20.3	34.3
204	W1000X371	47400	1000	401	19.1	36.1
205	W610X372	47400	668	335	26.4	48
206	W530X369	47600	602	325	27.9	50.5
207	W310X375	47800	391	330	35.6	57.2
208	W920X381	48600	951	310	24.4	43.9
209	W360X382	48800	417	406	30	48
210	W460X384	49000	546	300	32.5	58.4
211	W690X384	49100	737	363	24.9	45
212	W760X389	49700	803	386	23.6	41.9
213	W1100X390	49800	1100	401	19.9	36.1
214	W920X390	49800	937	422	21.3	36.6
215	W840X392	49900	876	401	22.1	39.9
216	W1000X393	49900	1020	302	24.4	43.9
217	W1000X412	52600	1010	401	21.1	40.1
218	W530X409	52800	612	328	31	55.6
219	W610X415	52800	678	338	29.5	53.1
220	W310X415	52800	404	333	38.9	62.7
221	W1000X415	53100	1020	305	26.2	46
222	W920X420	53500	942	422	22.5	39.9
223	W690X419	53600	744	366	26.9	49
224	W460X421	53700	556	302	35.6	63.5
225	W360X421	53700	424	409	32.8	52.6
226	W1100X433	55100	1110	401	22	40.1
227	W840X433	55200	884	404	24.4	43.9
228	W760X434	55500	813	389	25.9	47
229	W1000X438	55600	1030	305	26.9	49
230	W1000X443	56300	1010	401	23.6	41.9
231	W920X449	57400	947	424	24	42.7

232	W310X454	57700	414	335	41.4	68.8
233	W610X455	57900	688	340	32	57.9
234	W690X457	58200	752	366	29.5	53.1
235	W360X463	59000	434	411	35.8	57.4
236	W460X464	59100	566	305	38.6	69.6
237	W840X473	60500	894	406	26.4	48
238	W1000X483	61500	1020	404	25.4	46
239	W760X484	61900	823	391	29	52.1
240	W1000X486	61900	1040	307	30	54.1
241	W920X491	62500	958	422	25.9	47
242	W1000X494	63000	1040	310	31	54.1
243	W610X498	63400	699	343	35.1	63
244	W1100X499	63500	1120	404	26.2	45
245	W310X500	63800	427	340	45.2	75.2
246	W690X500	64000	762	371	32	57.9
247	W360X509	65200	445	417	39.1	62.7
248	W840X527	67100	904	409	29.5	53.1
249	W760X531	67700	833	394	31.5	56.9
250	W920X537	68400	965	424	28.4	51.1
251	W1000X539	68400	1030	406	28.4	51.1
252	W690X548	70300	772	373	35.1	63
253	W610X551	70300	711	348	38.6	69.1
254	W360X551	70300	455	419	42.2	67.6
255	W1000X554	71000	1030	409	29.5	52.1
256	W840X576	73500	914	411	32	57.9
257	W760X582	74200	843	396	34.5	62
258	W1000X584	74800	1060	315	36.1	64
259	W920X588	74800	975	427	31	55.9
260	W1000X591	75500	1040	409	31	55.9
261	W360X592	75500	465	422	45	72.4
262	W360X634	80600	475	424	47.8	77.2
263	W1000X642	81900	1050	411	34	59.9
264	W920X656	83900	988	432	34.5	62
265	W360X677	86500	483	427	51.3	81.5
266	W920X725	92300	998	434	38.1	68.1

267	W360X744	94800	498	432	55.6	88.9
268	W1000X748	95500	1070	417	39.1	70.1
269	W920X787	101000	1010	437	40.9	73.9
270	W690X802	103000	826	389	50	89.9
271	W360X818	105000	513	437	60.5	97
272	W1000X883	112000	1090	424	45.5	82
273	W360X900	115000	531	442	66	106
274	W920X970	124000	1040	447	50	89.9
275	W1000X976	125000	1110	429	50	89.9
276	W360X990	126000	549	450	71.9	115
277	W920X1077	137000	1060	452	55.1	99.1
278	W360X1086	139000	569	455	78	125
279	W920X1194	152000	1080	457	60.5	109
280	W360X1202	154000	579	472	95	130
281	W920X1269	162000	1090	462	64	115
282	W360X1299	166000	599	478	100	140
283	W920X1377	175000	1090	472	76.7	115

Table 3.5 HSS

Sec. No.	Sec. Name	Diameter (mm)	Wall Thickness (mm)	Sec. No.	Sec. Name	Diameter (mm)	Wall Thickness (mm)
1	HSS42.2x3.6	42.2	3.6	118	C19	225	10
2	HSS48.3x3	48.3	3	119	C20	225	12
3	HSS48.3x3.7	48.3	3.7	120	C21	225	14
4	HSS48.3x4.8	48.3	4.8	121	C22	225	16
5	HSS60.3x3.2	60.3	3.2	122	C23	225	18
6	HSS60.3x3.9	60.3	3.9	123	C24	225	20
7	HSS60.3x4.8	60.3	4.8	124	C94	225	22
8	HSS60.3x5.5	60.3	5.5	125	HSS244.5x4.8	244.5	4.8
9	HSS60.3x6.4	60.3	6.4	126	HSS244.5x6.4	244.5	6.4
10	HSS63.5x3.2	63.5	3.2	127	HSS244.5x7.9	244.5	7.9
11	HSS63.5x4.8	63.5	4.8	128	HSS244.5x9.5	244.5	9.5
12	HSS63.5x6.4	63.5	6.4	129	HSS244.5x12.7	244.5	12.7

13	HSS73x3.2	73	3.2	130	C25	250	10
14	HSS73x4.8	73	4.8	131	C26	250	12
15	HSS73x5.2	73	5.2	132	C27	250	14
16	HSS73x6.4	73	6.4	133	C28	250	16
17	HSS76.2x3.2	76.2	3.2	134	C29	250	18
18	HSS76.2x3.4	76.2	3.4	135	C30	250	20
19	HSS76.2x3.9	76.2	3.9	136	C95	250	22
20	HSS76.2x4.8	76.2	4.8	137	HSS254x4.8	254	4.8
21	HSS76.2x5.2	76.2	5.2	138	HSS254x6.4	254	6.4
22	HSS76.2x5.5	76.2	5.5	139	HSS254x7.9	254	7.9
23	HSS76.2x6.4	76.2	6.4	140	HSS254x9.5	254	9.5
24	HSS88.9x3.2	88.9	3.2	141	HSS254x12.7	254	12.7
25	HSS88.9x4.8	88.9	4.8	142	HSS254x15.9	254	15.9
26	HSS88.9x5.2	88.9	5.2	143	HSS273.1x6.4	273.1	6.4
27	HSS88.9x5.5	88.9	5.5	144	HSS273.1x9.5	273.1	9.5
28	HSS88.9x6.4	88.9	6.4	145	HSS273.1x12.7	273.1	12.7
29	HSS88.9x7.6	88.9	7.6	146	C31	275	10
30	HSS88.9x8	88.9	9	147	C32	275	12
31	HSS101.6x3.2	101.6	3.2	148	C33	275	14
32	HSS101.6x4.8	101.6	4.8	149	C34	275	16
33	HSS101.6x5.6	101.6	5.6	150	C35	275	18
34	HSS101.6x5.7	101.6	5.7	151	C36	275	20
35	HSS101.6x6	101.6	6	152	C96	275	22
36	HSS101.6x6.4	101.6	6.4	153	C37	300	10
37	HSS101.6x8	101.6	8	154	C38	300	12
38	HSS114.3x3.2	114.3	3.2	155	C39	300	14
39	HSS114.3x4.8	114.3	4.8	156	C40	300	16
40	HSS114.3x6	114.3	6	157	C41	300	18
41	HSS114.3x8.6	114.3	8.6	158	C42	300	20
42	HSS114.3x9.5	114.3	9.5	159	C97	300	22
43	HSS127x3.2	127	3.2	160	HSS323.9x6.4	323.9	6.4
44	HSS127x4.8	127	4.8	161	HSS323.9x9.5	323.9	9.5
45	HSS127x6.4	127	6.4	162	HSS323.9x12.7	323.9	12.7
46	HSS127x6.6	127	6.6	163	C43	325	10
47	HSS127x7.9	127	7.9	164	C44	325	12

48	HSS127x9.5	127	9.5	165	C45	325	14
49	HSS127x12.7	127	12.7	166	C46	325	16
50	HSS139.7x6.6	139.7	6.6	167	C47	325	18
51	HSS139.7x9.5	139.7	9.5	168	C48	325	20
52	HSS139.7x12.7	139.7	12.7	169	C98	325	22
53	HSS141.3x3.4	141.3	3.4	170	C49	350	10
54	HSS141.3x4.8	141.3	4.8	171	C50	350	12
55	HSS141.3x6.6	141.3	6.6	172	C51	350	14
56	HSS141.3x9.5	141.3	9.5	173	C52	350	16
57	HSS141.3x12.7	141.3	12.7	174	C53	350	18
58	C1	150	10	175	C54	350	20
59	C2	150	12	176	C99	350	22
60	C3	150	14	177	HSS355.6x6.4	355.6	6.4
61	C4	150	16	178	HSS355.6x7.9	355.6	7.9
62	C5	150	18	179	HSS355.6x9.5	355.6	9.5
63	C6	150	20	180	HSS355.6x12.7	355.6	12.7
64	C91	150	22	181	HSS355.6x15.9	355.6	15.9
65	HSS152.4x3.2	152.4	3.2	182	C55	375	10
66	HSS152.4x4.8	152.4	4.8	183	C56	375	12
67	HSS152.4x6.4	152.4	6.4	184	C57	375	14
68	HSS152.4x7.1	152.4	7.1	185	C58	375	16
69	HSS152.4x7.9	152.4	7.9	186	C59	375	18
70	HSS152.4x9.5	152.4	9.5	187	C60	375	20
71	HSS152.4x12.7	152.4	12.7	188	C100	375	22
72	HSS168.3x3.2	168.3	3.2	189	C61	400	10
73	HSS168.3x4.8	168.3	4.8	190	C62	400	12
74	HSS168.3x6.4	168.3	6.4	191	C63	400	14
75	HSS168.3x7.1	168.3	7.1	192	C64	400	16
76	HSS168.3x7.9	168.3	7.9	193	C65	400	18
77	HSS168.3x9.5	168.3	9.5	194	C66	400	20
78	HSS168.3x11	168.3	11	195	C101	400	22
79	HSS168.3x12.7	168.3	12.7	196	HSS406.4x6.4	406.4	6.4
80	HSS174.6x4.8	174.6	4.8	197	HSS406.4x7.9	406.4	7.9
81	HSS174.6x6.4	174.6	6.4	198	HSS406.4x9.5	406.4	9.5
82	HSS174.6x7.9	174.6	7.9	199	HSS406.4x11.1	406.4	11.1

83	HSS174.6x9.5	174.6	9.5	200	HSS406.4x12.7	406.4	12.7
84	HSS174.6x12.7	174.6	12.7	201	HSS406.4x15.9	406.4	15.9
85	C7	175	10	202	C67	425	10
86	C8	175	12	203	C68	425	12
87	C9	175	14	204	C69	425	14
88	C10	175	16	205	C70	425	16
89	C11	175	18	206	C71	425	18
90	C12	175	20	207	C72	425	20
91	C92	175	22	208	C102	425	22
92	HSS177.8x3.2	177.8	3.2	209	C73	450	10
93	HSS177.8x4.8	177.8	4.8	210	C74	450	12
94	HSS177.8x6.4	177.8	6.4	211	C75	450	14
95	HSS177.8x7.9	177.8	7.9	212	C76	450	16
96	HSS177.8x9.5	177.8	9.5	213	C77	450	18
97	HSS177.8x12.7	177.8	12.7	214	C78	450	20
98	HSS190.5x4.8	190.5	4.8	215	C103	450	22
99	HSS190.5x6.4	190.5	6.4	216	HSS457.2x9.5	457.2	9.5
100	HSS190.5x7.9	190.5	7.9	217	HSS457.2x12.7	457.2	12.7
101	HSS190.5x9.5	190.5	9.5	218	C79	475	10
102	HSS190.5x12.7	190.5	12.7	219	C80	475	12
103	HSS193.7x8.3	193.7	8.3	220	C81	475	14
104	HSS193.7x9.5	193.7	9.5	221	C82	475	16
105	C13	200	10	222	C83	475	18
106	C14	200	12	223	C84	475	20
107	C15	200	14	224	C104	475	22
108	C16	200	16	225	C85	500	10
109	C17	200	18	226	C86	500	12
110	C18	200	20	227	C87	500	14
111	C93	200	22	228	C88	500	16
112	HSS219.1x4.8	219.1	4.8	229	C89	500	18
113	HSS219.1x6.4	219.1	6.4	230	C90	500	20
114	HSS219.1x8.2	219.1	8.2	231	C105	500	22
115	HSS219.1x9.5	219.1	9.5	232	HSS508x9.5	508	9.5
116	HSS219.1x12.7	219.1	12.7	233	HSS508x12.7	508	12.7
117	HSS219.1x15.9	219.1	15.9				

It is required to find a vector of integer values representing the design variables as follows:

$$\mathbf{I} = [I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9] \quad (3.4)$$

where $\mathbf{I}$ is the design variables vector, I_1 is the first design variable, which represents the HSS of the inner column on floor 1, I_2 is the second design variable, which represents the HSS of the outer column on floor 1, I_3 is the third design variable, which represents the HSS of the inner column on floor 2, I_4 is the fourth design variable, which represents the HSS of the outer column on floor 2, I_5 is the fifth design variable, which represents the HSS of the inner column on floor 3, I_6 is the sixth design variable, which represents the HSS of the outer column on floor 3, I_7 is the seventh design variable, which represents the W-section of the inner beams in the three floors, I_8 is the eighth design variable, which represents the W-section of the outer beams in the three floors, and I_9 is the ninth design variable, which represents the concrete strength (f'_c).

3.3.3 Design Constraints

The composite column and W-section beam design constraints are applied as follows:

1. Strength Constraint in column

This constraint to make sure of the applied shear and moment as follows:

$$\lambda_p = 0.09 \frac{E_s}{f_y} \geq \frac{D}{t} \quad \text{Compact section} \quad (3.5)$$

Where: λ is the slenderness ratio

E_s is the modulus of elasticity of steel.

f_y is the yield stress of structural steel.

D is the section diameter

t is the wall thickness

For compact sections the compressive strength $P_{no}=P_p$ (3.6)

Where:

$$P_p = A_s \cdot f_y + C_2 f'_c A_c \quad (3.7)$$

$C_2 = 0.85$ for rectangular sections and 0.95 for round sections.

A_c = area of concrete, (mm²)

A_s = area of steel cross-section, (mm²)

When $\frac{P_{no}}{P_e} \leq 2.25$

$$P_n = P_{no} (0.658^{\frac{P_{no}}{P_e}}) \quad (3.8)$$

When $\frac{P_{no}}{P_e} > 2.25$

$$P_n = 0.877 P_e \quad (3.9)$$

$$\text{Where: } P_e = \frac{\pi^2 (EI_{eff})}{(KL)^2} \quad (3.10)$$

EI_{eff} = effective stiffness of composite section (N-mm²)

$$= E_s I_s + C_3 E_c I_c \quad (3.11)$$

C_3 = coefficient for calculation of effective rigidity of an encased composite compression member

$$= 0.45 + 3 \left(\frac{A_s}{A_g} \right) \leq 0.9 \quad (3.12)$$

$$P_u (\text{applied force}) \leq \phi P_n \quad (3.13)$$

$$\phi = 0.75 \text{ (LFRD)}$$

$$\text{For compact sections the flexural strength } M_n = M_p \quad (3.14)$$

$$M_p = 4f_y t r_m^2 \cos(r_o) + 0.95 \left(\frac{2}{3} f'_c r_i^3 \cos^3(r_o) \right) \quad (3.15)$$

$$r_m = \frac{d-t}{2} \quad (3.16)$$

$$r_i = \frac{d-2t}{2} \quad (3.17)$$

$$r_o = \frac{\pi}{2} F_{CHS} \quad (3.18)$$

$$F_{CHS} = \frac{\frac{1}{8} \left(\frac{0.95 f'_c}{f_y} \right) \left(\frac{d-2t}{t} \right)}{1 + \frac{1}{4} \left(\frac{0.95 f'_c}{f_y} \right) \left(\frac{d-2t}{t} \right)} \quad (3.19)$$

$$M_u = \phi M_n \quad (3.20)$$

$$\phi = 0.9 \text{ (LFRD)}$$

Combined flexure and axial force (for compact sections)

$$\text{If } \frac{P_u}{\phi P_n} \geq 0.2$$

$$\frac{P_u}{\phi P_n} + \frac{8}{9} \left(\frac{M_{ux}}{\phi M_{nx}} + \frac{M_{uy}}{\phi M_{ny}} \right) \leq 1.0 \quad (3.21)$$

$$\text{If } \frac{P_u}{\phi P_n} < 0.2$$

$$\frac{P_u}{2\phi P_n} + \left(\frac{M_{ux}}{\phi M_{nx}} + \frac{M_{uy}}{\phi M_{ny}} \right) \leq 1.0 \quad (3.22)$$

2. Strength Constraint in beams

This constraint is to make sure of the PMM ratio (the moment capacity ratio), Vmaj ratio (the shear about the major axis ratio), and Vmin ratio (the shear about the minor axis ratio) for the beams and these values can be imported directly through ETABS as follows:

$$\text{PMM ratio} - 1 \leq 0 \quad (3.23)$$

$$\text{Vmaj ratio} - 1 \leq 0 \quad (3.24)$$

$$\text{Vmin ratio} - 1 \leq 0 \quad (3.25)$$

3. Deflection Constrain

The displacement constraints shall comply with:

$$\frac{\delta_{max}}{\delta_a} - 1 \leq 0 \quad (3.26)$$

where δ_{max} is the maximum displacement due to all the load combinations (mm) and δ_a is the allowable displacement which is equal to the beam span in millimeters divided by 360.

4. Geometry Constrain

This constraint is to make sure that the beam width is limited to the column diameter and to make sure that the column width.

$$\frac{b_{beam}}{d_{column}} - 1 \leq 0 \quad (3.27)$$

Chapter Four

Methods and Procedures

4 Methods and Procedures

In the previous chapter, the optimum design problem under AISC provisions for the composite columns in a building structure (composite steel column filled with concrete and W-section I-beams) was explained. This three-storey building was modeled in ETABS, and the mathematical modeling of the structure design optimization problems was established.

The primary goal of this study was to create an automated optimum design method capable of solving structure design optimization challenges. This is accomplished by programming in MATLAB the objective function and design restrictions. As described in previous chapters, metaheuristic algorithms are used to solve the design optimization problem. These algorithms seek the best solution by varying the values of the design variables with each move around the design domain. When the values of the design variables are altered, the new response of the structure under external loading must be determined. This involves an examination of the structure under external loads using the updated design variable values. The analysis is conducted using the ETABS program. This is accomplished by integrating the structure models with ETABS's application programming interface instructions. The structure system's response is passed to the MATLAB program, where it is analyzed, and new values of design variables are determined by metaheuristic algorithms.

This chapter discusses the methodologies and procedures used to create the structure design optimization algorithms that were used in this study.

4.1 Application Programming Interface (API)

ETABS is integrated software for structural analysis and design produced by Computers and Structures Incorporation (CSI). Its acronym comes from the state-of-the-art practice (CSI Knowledge Base, 2020). ETABS can be used to model different types of structures such as buildings, bridges, and infrastructures. It has the ability to model both 2D and 3D structures, analyze simple and complex structures, perform linear and non-linear analysis, execute static and dynamic analysis, apply different types of loadings such as wind, vehicle, and seismic according to integrated codes, generate tabular and graphical outputs, create reports, etc.

The main reason for selecting ETABS from the available software to model and analyze the structure was its ability to consider the composite filled sections and design it as one element, and its ability to interface with MATLAB through API commands. API is the abbreviation for application programming interface that allows two-way automated communication between two different software packages (CSI Computer and Structures, 2020). ETABS provides this facility to allow users to access the software's built-in functions and use them as per their needs. API functions in ETABS are compatible with different types of programming languages including Visual Basic, MATLAB, C++, java, etc. This interface is implemented by applying API functions that are available in the documentation found under the path C:\Program Files\Computers and Structures\ETABS21. API also assures safe and accurate transfer of data such as importing and assigning information to ETABS and extracting analysis results and design parameters from ETABS.

All the above-mentioned reasons make API an excellent tool for the two-way transfer of information between the ETABS model and the structural design optimization algorithm, which facilitates the optimization process and makes it faster and more accurate, especially given that the structural design problem has a composite members analysis.

Table 4.1 shows the main API functions that have been used in the structural design optimization algorithms.

Table 4.1 API functions used

No.	API Function	Description
1	ETABSObject.ApplicationStart	Start ETABS application
2	ETABSObject.SapModel	Create SapModel object
3	SapModel.InitializeNewModel	Initialize model
4	SapModel.File.OpenFile	Open an existing file
5	SapModel.SetPresentUnits	Set present units
6	SapModel.SetModelIsLocked	Unlock model
7	SapModel.PropMaterial.SetOConcrete	Add or modify concrete material properties
8	SapModel.FrameObj.SetSection	Assign a frame section property to a frame object
9	SapModel.PropFrame.SetISection	Add or modify the I-section shape to a new property
10	SapModel.FrameObj.SetSection	Assign a frame section property to a frame object

No.	API Function	Description
11	SapModel.PropFrame.Delete	Delete a frame section property
12	SapModel.PropFrame.GetSectProps	Get section properties
13	SapModel.DesignSteel.StartDesign	Start steel design
14	SapModel.DesignSteel.GetSummary Results_2	Get steel design results
15	SapModel.Analyze.RunAnalysis	Start analysis
16	SapModel.Results.Setup.DeselectAll CasesAndCombosForOutput	Deselect all load cases and combinations
17	SapModel.Results.Setup.SetComboS electedForOutput	Set load combinations selected for output
18	SapModel.Results.FrameForce	Get frame forces
19	SapModel.Results.JointDispl	Get joint displacements
20	SapObject.ApplicationExit	Save and close SAP2000

It should be mentioned that while utilizing API commands in MATLAB, it is critical to properly initialize the arrays in which the data is stored whenever appropriate. Some API commands were written in arrays to correctly retrieve the relevant information. It is also necessary to verify that these instructions function properly and retrieve the appropriate information. This is verified by writing the API commands as follows:

ret = SapModel.InitializeNewModel (4.1)

If the value of “ret” in the above example equation is equal to zero, this means that the API function is retrieving the information correctly. Additionally, if the value is equal to one, it means that there is a problem in executing the command and the API function is not working properly.

Identifying errors while performing API commands is difficult and time-consuming, especially when the commands are written in MATLAB. The API help document includes several examples using the Visual Basic language, but only one example shows how these commands are implemented in MATLAB, which is insufficient.

It is strongly advised to double-check the execution of the API commands at each step. Appendix A contains application examples of API functions from this thesis.

The method of interfacing the optimum design algorithm with ETABS through API commands was not found in some of the previous studies, so the previous studies on optimum design algorithm with ETABS through API such as the research carried out by (Sextos and Balafas, 2011). In this research, the researchers took advantage of automated bridge analysis by interfacing with a third-party engine self-developed program. (Kaveh et al., 2014) applied the interfacing successfully in the design optimization of box girder bridges using a CS algorithm. (Karakas, Ozgan and Daloglu, 2016) applied the interfacing with MATLAB for the structural analysis of hyperbolic cooling towers. The interfacing was also successfully applied with MATLAB in the design optimization of a braced steel space frame using HS and teaching-learning-based-optimization (TLBO) algorithms (Daloglu, Artar, Ozgan, & Karakas, 2018).

4.2 Metaheuristic Algorithms

We studied metaheuristic algorithms in Chapter 2. They were utilized to find the solution for the composite frame design optimization algorithms. As previously stated, metaheuristic algorithms are more suited and efficient than classical optimization procedures for dealing with discrete non-linear programming optimization problems. In addition, a survey of the literature proved their ability to address large and complex issues.

4.2.1 Artificial Bee Colony Algorithm

The artificial bee colony algorithm (ABC) is a nature-inspired method developed by Karaboga in 2005, modeling the foraging behavior of bees in their quest for the optimal food source for the hive.

ABC categorizes bees into three groups: employed, onlooker, and scout bees, each assigned specific roles. Employed bees are tasked with gathering nectar from the best food source and transporting it to the hive. Scouts, on the other hand, explore and discover new food sources, communicating their findings to onlooker bees through dance. Onlookers, positioned in the vicinity of the dancing scouts, assimilate this information and determine the most favorable food source.

Initially, scout bees initiate the search for the optimal food source, returning to the hive to convey their findings to onlooker bees. The onlookers then choose the best food source, with some being transformed into employed bees. As the selected food source depletes, the employed bees shift roles, becoming scouts to initiate a new search.

It's important to note that an increase in the number of scout bees enhances the exploration process, while an increase in onlookers

intensifies the exploitation process. In the ABC algorithm, the population corresponds to food sources, the best solution represents the optimal food source, and the nectar quantity signifies the solution quality, measured by the fitness value.

ABC has three control parameters that must be decided before running the algorithm. These are the colony size, limit, and maximum number of cycles (iterations). The colony size corresponds to the equal number of employed bees plus the onlooker bees. Note that half of the colony represents the number of initial food sources (solutions). The limit value controls the conversion of employed bees to scouts when the food sources become abandoned.

$$Limit = 0.5 * nVar * nPop \quad (4.2)$$

where $nPop$ equals the colony size and $nVar$ is the dimension of the problem, which represents the number of design variables.

The algorithm has the following four phases (Karaboga, 2010; Karaboga & Akay, 2011):

1- Initialization

At the beginning, the population is initialized randomly by the scout bees using the following equation:

$$x_i^j = x_{min}^j + r(x_{max}^j - x_{min}^j) \quad (4.3)$$

where x_i^j is the i^{th} solution ($i = 1, 2, 3, \dots, SN$) for j dimensional vector ($j = 1, 2, 3, \dots, D$), x_{min}^j is the lower bound, x_{max}^j is the upper bound and r is a random number within the range (0,1).

2- Employed Bees

They search for and find new food sources (solutions) within the vicinity such that:

$$v_i^j = x_i^j + \phi_i^j(x_i^j - x_k^j) \quad (4.4)$$

where v_i^j is the new food source, x_i^j is the previous food source calculated in equation (4.3), ϕ_i^j is a random number within the interval $[-1,1]$ and x_k^j is a randomly selected food source in the vicinity of i .

They also evaluate the amount of nectar in the food source by calculating its fitness:

$$fit(x_i) = \begin{cases} 1/(1 + f(x_i)), & \text{if } f(x_i) < 0 \\ 1 + abs(f(x_i)), & \text{if } f(x_i) \geq 0 \end{cases} \quad (4.5)$$

where $fit(x_i)$ is the fitness value of the i^{th} solution and $f(x_i)$ is the value of the objective function of the i^{th} solution.

3- Onlooker Bees

The onlooker bees select the best food source/s based on the probability values calculated using the following equation:

$$P_i = \frac{fit(x_i)}{\sum_{i=1}^{SN} fit(x_i)} \quad (4.6)$$

where P_i is the probability value of the food source i .

After that, a mutual solution of the i^{th} solution is to be produced randomly using equation (4.4).

4- Scout Bees

When the solutions provided by employed bees are deemed abandoned and unreceptive to further improvement, they undergo a transformation

into scout bees. This conversion process is regulated by the previously mentioned limit parameter. The identification of an exhausted food source triggers the replacement mechanism, wherein a new, random food source is generated. The calculation of this new food source is carried out using equation (4.3).

5- Phases 2 to 4 are repeated until the termination criterion is reached.

The MATLAB implementation of the ABC algorithm is available directly from Karaboga et al. (2018). This algorithm was chosen due to its widespread application and positive reputation for effectively addressing structural design optimization problems. The versatility of the ABC algorithm extends beyond small-scale structural optimization issues, such as the design of tension/compression strings, welded beams, pressure vessels, trusses, and steel plane frames. Notably, it has demonstrated effectiveness in handling large-dimensional problems characterized by high constraints and discrete variables, as encountered in real-world scenarios like multi-storey steel space frame design.

The performance of the ABC algorithm in these instances surpassed that of other techniques and algorithms, as evidenced by superior results compared to methodologies presented by Garg (2014) and Aydogdu, Akin, & Saka (2016).

4.2.2 Differential Evolution (DE)

Differential Evolution (DE) is an evolutionary algorithm falling under the category of population-based optimization methods. Rainer Storn and Kenneth Price introduced it in 1997 with the primary aim of addressing optimization problems, where the objective is to discover the minimum or maximum of a specified objective function. DE is renowned for its

simplicity, efficiency, and effectiveness, particularly in tackling a broad spectrum of optimization challenges, especially those involving continuous and high-dimensional search spaces (Lu et al., 2023).

Metaheuristic algorithms are adept at handling unconstrained optimization problems. When it comes to dealing with constrained optimization problems, a common approach involves transforming the constrained problem into an unconstrained one. One method to achieve this transformation is by applying a penalty function approach. In this method, the following equation is applied:

$$\emptyset = Q \left\{ 1 + \alpha \left(\sum_{i=1}^n g_i \right) \right\} \quad (4.7)$$

where the $\emptyset$ is the unconstrained or penalized objective function, Q is the objective function, α here is a penalty constant taken preferably as 1, g_i is the value of the i^{th} normalized constraint and n is the number of constraints (Aydođdu et al., 2016; Lute et al., 2011; Saka et al., 2013).

Metaheuristic algorithms also require normalization of the design constraints before seeking a solution. A normalized constraint means that the constraint equation is written in a form less than or equal to zero, as shown in the following example equation:

$$g(x) > h(x) \quad \text{before normalization} \quad (4.8)$$

$$\frac{h(x)}{g(x)} - 1 \leq 0 \quad \text{after normalization} \quad (4.9)$$

After the application of the normalization and penalty function method, the optimization problem becomes suitable to be handled by the metaheuristic algorithms.

Metaheuristic algorithms demonstrate the capability to address intricate and sizable constrained design optimization problems, even when discrete variables are involved. This effectiveness has been demonstrated across diverse structural design problems, encompassing both steel and reinforced concrete structures. The adaptability of metaheuristic algorithms makes them valuable tools for tackling the complexity inherent in such design challenges, providing solutions that account for constraints and discrete variables in the optimization process. (Akin & Saka, 2015; Hasançebi et al., 2009).

4.2.3 Teaching-Learning-Based Optimization (TLBO)

Teaching-Learning-Based Optimization (TLBO) is a metaheuristic algorithm devised by (R. V. Rao, 2011). This algorithm is inspired by the dynamics of the teaching-learning process found in a classroom, aiming to replicate the influence of a teacher on students. Known for its advantageous features, including swift convergence, a lack of algorithm-specific parameters, and straightforward implementation, TLBO has garnered recognition as an effective optimization algorithm. Its versatility is evident through successful applications to real-world problems spanning diverse fields, as highlighted in the work of (Wu et al., 2022).

TLBO functions as a population-centric algorithm, where a collective of students, referred to as learners, constitutes the population. In this analogy, the various subjects presented to the learners align with the distinct design variables involved in the optimization problem. The outcomes achieved by the learners are akin to the fitness values associated with the optimization problem. Within this framework, the most optimal solution within the entire learner population assumes the

role of the teacher. The mechanics of the TLBO algorithm can be delineated through its teacher and learner phases.(Rao & Patel, 2013a)

In the teacher phase of the algorithm, the learning process of students (learners) occurs through the teacher. The teacher imparts knowledge to the learners, aiming to enhance the overall mean result of the class. Consider a scenario where there are 'm' subjects (design variables) offered to 'n' learners (population size denoted by $k = 1, 2, \dots, n$). In each teaching-learning cycle 'i', $M_{j,i}$ represents the mean result of learners in a specific subject 'j' (where $j = 1, 2, \dots, m$). The most proficient learner in the entire population is designated as the teacher for that cycle, denoted by $X_{total-kbest,i}$. The teacher strives to elevate the knowledge level of the entire class, and the learners gain knowledge based on the teaching quality and the learners' inherent capabilities.

The disparity between the teacher's result and the mean result of learners in each subject is quantified as follows:

$$\text{Difference_Mean}_{j,i} = r_i \cdot (X_{j,kbest,i} - T_F \cdot M_{j,i}), \quad (4.10)$$

where $X_{j,kbest,i}$ signifies the teacher's result in subject j, T_F is the teaching factor determining the mean value to be modified, and r_i is a random number in the range [0, 1]. The T_F value, either 1 or 2, is randomly determined with equal probability using the expression:

$$T_F = \text{round}[1 + \text{rand}(0,1) \cdot (2-1)], \quad (4.11)$$

where rand denotes a random number in the range [0, 1]. Notably, T_F is not a predefined parameter but is dynamically decided by the algorithm.

The existing solution is updated during the teacher phase based on the $\text{Difference_Mean}_{j,i}$ using the expression:

$$X'_{j,k,i} = X_{j,k,i} + \text{Difference_Mean}_{j,i}, \quad (4.12)$$

where $X'_{j,k,i}$ represents the updated value of $X_{j,k,i}$. Accept $X'_{j,k,i}$ if it yields a superior function value. All accepted function values at the end of the teacher phase are retained, serving as input for the subsequent learner phase.

It's worth noting that the values of r_i and T_F play a role in TLBO's performance. While r_i is a random number in the range $[0, 1]$, and T_F is the teaching factor, their values are generated randomly within the algorithm. Unlike other optimization algorithms that require tuning of specific parameters, TLBO does not necessitate tuning of r_i and T_F . TLBO only requires tuning of common control parameters like population size and the number of generations, making it a parameter-less algorithm specific to its operation.

During the Learner phase of the algorithm, the learning process of students (learners) occurs through interactions among themselves. Students have the opportunity to gain knowledge by engaging in discussions and interacting with their peers. The learning mechanism in this phase is described as follows:

Two learners, denoted as P and Q, are randomly selected, ensuring that their updated values, $X'_{\text{total-P},i}$ and $X'_{\text{total-Q},i}$ (derived from the end of the teacher phase), are not equal:

$$X'_{\text{total-P},i} \text{ And } \neq X'_{\text{total-Q},i} . \quad (4.13)$$

This selection process ensures that the learners involved in the interaction have distinct and updated values from the teacher phase. The aim is to facilitate knowledge exchange among learners, allowing them to learn new information from their peers, especially if the selected peer has more knowledge. This interaction-based learning approach adds a collaborative

dimension to the algorithm, fostering the sharing of information among learners.

For maximization problems, the update is expressed as follows:

$$X''_{j,P,i} = X'_{j,P,i} + r_i \cdot (X'_{j,P,i} - X'_{j,Q,i}), \text{ if } X'_{\text{total}-P,I} > X'_{\text{total}-Q,I} \quad (4.14)$$

and for the reverse scenario, when

$$X''_{j,P,i} = X'_{j,P,i} + r_i \cdot (X'_{j,Q,i} - X'_{j,P,i}), \text{ if } X_{\text{total}-Q,I} > X_{\text{total}-P,i'} \quad (4.15)$$

These equations are applicable to maximization problems, and for minimization problems, the conditions and operations would be reversed. The updated value $X''_{j,P,i}$ is then accepted if it yields a superior function value, contributing to the ongoing optimization process. (Rao & Patel, 2013b)

4.3 Composite Column Design Optimization Algorithm

As previously stated, the column design optimization issue was written in MATLAB. The metaheuristic algorithm, the column design optimization method, and the structure model collaborated on the entire optimization process.

Figure 4.1 depicts the interactions between the three primary parties.

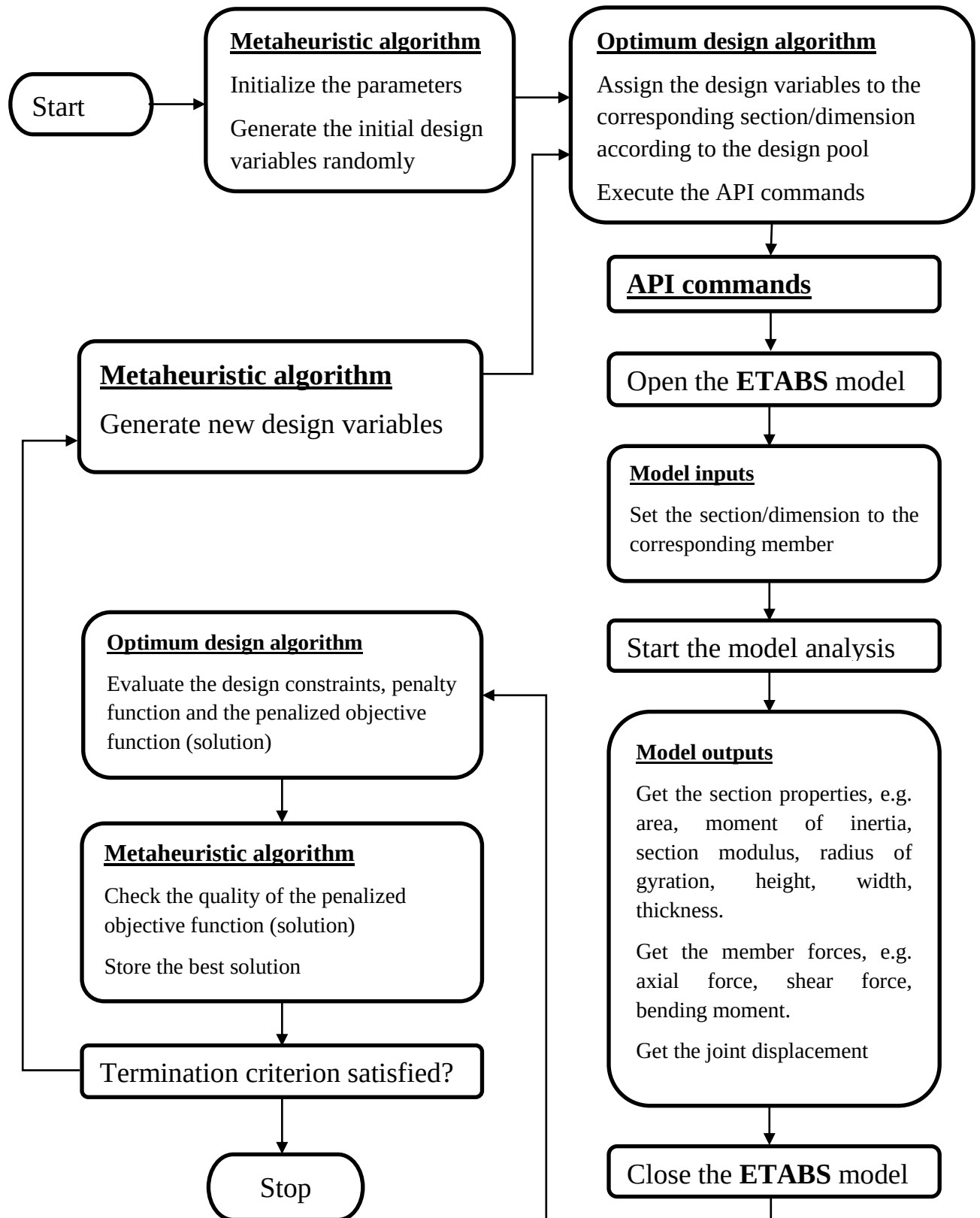


Figure 4.1 The interactions between the three main optimization parties (M.A.Latif, 2019)

The optimization procedure begins with the execution of the metaheuristic algorithm in MATLAB it was downloaded from the literature and then, was recoded to fit the column optimum design(Mostapha Kalami Heris, 2015c, 2015b, 2015a). The initial design variables are chosen at random by the metaheuristic algorithm and sent to the column design optimization algorithm. These design variables communicate with the ETABS model via API instructions. The model is scrutinized, and the section properties and analysis results are fed back into the column design optimization method. The column design program assesses the constraints, computes the penalty function and penalized objective function, and then passes the results to the metaheuristic algorithm. The metaheuristic algorithm evaluates the penalized objective function (solution) and saves the best solutions. If the termination criterion is not fulfilled, the metaheuristic algorithm creates new design variables, and the process is repeated until the termination criterion is met, which is the total number of iterations in this case.

As previously stated in Chapter 2, metaheuristic algorithms are based on randomness, which means that they generate random numbers in some of the phases. As a result, each time an algorithm is executed, the end, result is unique. The goal of seed values is to ensure that the end output of an algorithm is the same every time it is performed with the same range (random) number.

Computers with high-performance speed and substantial RAM storage are required for this type of challenge to successfully complete the entire optimum design procedure. The department loaned a PC with high-quality characteristics.

4.4 Summary

The following are general summaries of the methodology and procedures used in this study:

- Column design examples include a composite steel column filled with concrete.
- Create a structure in ETABS software.
- Create a mathematical model of the optimization problem for column design. This comprises defining the three optimization components: objective function, design variables, and restrictions.
- Prepare the design pool associated with the design variables.
- In MATLAB, create the bridge design optimization algorithm for the above mathematical model.
- In the column design optimization algorithm, include API instructions. This is the link between the algorithm and the structure model.
- To solve the optimum design problem, choose one of the metaheuristic algorithms already written in MATLAB.
- Modify the metaheuristic algorithm to accommodate constrained-type situations and discrete design variables.
- Connect the preceding MATLAB-coded column design optimization technique to the metaheuristic algorithm.
- Set and tweak the metaheuristic algorithm's tuning parameters, and then decide on the number of runs.
- Run the metaheuristic algorithm to get the best results.

The optimal design outcomes are provided in the following chapter.

Chapter Five

Results and

Discussion

5 Results and Discussion

This chapter illustrates the outcomes of the instances discussed in Chapter 4. The optimization process's outcomes are given and discussed in the pages that follow.

5.1 Composite Steel column filled with concrete design optimization.

The single-storey two-bay structure was chosen first to see if the methods used in this study worked appropriately. This structure was a nice starting point because it was a single-storey, two-bay structure with a limited amount of design choices.

ABC, TLBO, and DE were chosen as metaheuristic algorithms to solve this design optimization challenge.

The population size and the limit are the two tuning parameters for ABC. For each run, the population was modified to find the best population number that produces the best results in the shortest period of time while adjusting the limit value according to equation (4.2). It was discovered that when the population number was set to 30, the final answer was significantly better than when it was set to 10. Setting the population to 50 or higher did not change the final results, but it did increase computational time, which was unnecessary. As a result, the parameters chosen were 20 for the population and 135 for the limit.

DE has three variables. After multiple attempts, it was discovered that utilizing 0.25 for crossover probability was effective. The lower bound of 0.2 and the upper bound of 0.8 for the number of scaling factors were appropriate. The population size was set at 30. When this value is

increased, the computational time increases as well, with no influence on the final results.

The same method was used with TLBO. This algorithm was simpler to test because it only has one tuning parameter, which is the population. Again, the most appropriate population size was 30.

The parameters listed above are summarized in Table 5.1 below.

Table 5.1 Metaheuristic parameters for the composite column example

Algorithm	Population	Other Parameters	No. of Iterations
ABD	30	Limit= 135	200
DE	30	crossover probability = 0.2 scaling factors = LB 0.2 and UB 0.8	200
TLBO	30	-	200

The optimal outcomes for the algorithms for each iteration are tabulated in Table 5.2 below. The best result obtained for TLBO and DE algorithms was 170 kN. The worst result obtained by ABC was 173 kN.

Table 5.2 Metaheuristic algorithms result for the frame structure example

Iteration	ABC		DE		TLBO	
	Best Weight (KN)	Constraint	Best Weight (KN)	Constraint	Best Weight (KN)	Constraint
1	264665456	466.8991	264665456	482.4267	264665456	465.727
2	264665456	466.8991	264665456	466.8991	264665456	465.727
3	264665456	466.8991	264665456	461.8666	264665456	465.727
4	264665456	466.8991	216067271	375.038	264665456	465.727

5	264665456	466.8991	216067271	375.038	264665456	465.727
6	149163312	382.9444	133903759	290.2728	264665456	465.727
7	149163312	382.9444	133903759	290.2728	243884344	534.7295
8	133903759	290.2728	90003159	215.2728	243884344	534.7295
9	92157258	294.6295	90003159	215.1214	149163312	382.9444
10	92157258	294.6295	60747387	179.892	92157258	294.6295
11	90003159	215.1214	60747387	179.892	92157258	294.6295
12	60747387	179.892	60747387	179.892	65887826	244.202
13	60747387	179.892	60747387	179.892	65887826	244.202
14	60747387	179.892	56975942	175.8158	65887826	244.202
15	60747387	179.892	44677983	153.592	65887826	244.202
16	60747387	179.892	44677983	153.592	65887826	244.202
17	60747387	179.892	44677983	153.592	65887826	244.202
18	55486307	204.2221	44677983	153.592	65887826	244.202
19	55486307	204.2221	41616045	152.4563	65887826	244.202
20	52219928	229.9997	41616045	152.4563	62288295	241.5053
21	52219928	229.9997	41616045	152.4563	62288295	241.5053
22	52219928	229.9997	37393614	155.9049	62288295	241.5053
23	52219928	229.9997	22319513	110.6905	62288295	241.5053
24	31399219	143.3847	22319513	110.6905	55486307	204.2221
25	31399219	143.3847	22319513	110.6905	55486307	204.2221
26	31399219	143.3847	22319513	110.6905	52219928	229.9997
27	15810996	96.306	16612855	96.7561	52219928	229.9997
28	15810996	96.306	16612855	96.7561	50992813	215.5301
29	15810996	96.306	16612855	96.7561	50195887	201.3713
30	15810996	96.306	16612855	96.7561	50195887	199.4205
31	15810996	96.306	15810996	96.306	50195887	198.153
32	15810996	96.306	15810996	96.306	50195887	198.153
33	15810996	96.306	15810996	96.306	50195887	201.3713
34	15810996	96.306	15810996	96.306	31399219	143.3847
35	15810996	96.306	15810996	96.306	31399219	143.3847
36	15810996	96.306	15088821	98.0218	31399219	143.3847
37	15088821	98.0218	15088821	98.0218	31399219	143.3847
38	15088821	98.0218	15088821	98.0218	31399219	143.3847
39	14251490	90.6864	13994904	91.9877	31399219	143.3847
40	14251490	90.6864	13334528	87.9704	31399219	143.3847
41	14251490	90.6864	11346867	81.4065	29208637	134.497
42	14251490	90.6864	11346867	81.4065	29208637	134.497
43	8838545	71.1053	7617157	64.1884	25311459	131.8236
44	8838545	70.3226	7617157	64.1884	14251490	90.6864

45	8838545	70.3226	7617157	64.1884	14251490	90.6864
46	8838545	70.3226	7617157	64.1884	14251490	90.6864
47	8838545	70.3226	7617157	64.1884	14251490	90.6864
48	8838545	70.3226	7617157	64.1884	14251490	90.6864
49	8838545	69.1477	7617157	64.1884	14251490	90.6864
50	8838545	68.072	7601459	65.7164	14251490	90.6864
51	8838545	68.072	7601459	65.7164	14251490	90.6864
52	8838545	68.072	7491697	63.4678	14251490	90.6864
53	8838545	68.072	7491697	63.4678	14251490	90.6864
54	8838545	68.072	7491697	63.4678	14251490	90.6864
55	8838545	68.072	5369023	55.6408	14251490	90.6864
56	3937226	45.7555	5369023	55.6408	14251490	90.6864
57	3937226	45.7555	5369023	55.6408	14251490	90.6864
58	3937226	45.7555	5255376	55.531	14251490	90.6864
59	3937226	45.7555	5255376	55.531	14251490	90.6864
60	3912819	45.4906	5255376	55.531	14251490	90.6864
61	3912819	45.4906	5255376	55.531	14251490	90.6864
62	3912819	45.4906	5255376	55.531	14251490	90.6864
63	3912819	45.4906	5255376	55.531	14251490	90.6864
64	3912819	45.4906	5255376	55.531	14251490	90.6864
65	3912819	45.4906	5255376	55.531	13068825	89.4751
66	3912819	45.4906	5255376	55.531	13068825	89.4751
67	3873695	45.4906	5255376	55.531	13068825	89.4751
68	3873695	45.4906	4643006	50.4492	13068825	89.4751
69	3194069	41.2752	2932564	41.2719	12627489	86.0777
70	3194069	41.2752	2932564	41.2719	8838545	71.6526
71	3194069	41.2752	2932564	41.2719	8838545	71.1053
72	3194069	41.2752	2932564	41.2719	8838545	70.3226
73	3194069	41.2752	2932564	40.9036	8838545	70.3226
74	3194069	41.2752	2932564	40.9036	8838545	70.3226
75	3194069	41.2752	2932564	40.9036	8838545	70.3226
76	3194069	41.2752	2932564	40.9036	8838545	70.3226
77	3194069	41.2752	2932564	40.9036	8838545	69.1477
78	3194069	41.2752	2365540	36.9821	8838545	68.072
79	2641813	38.3838	2365540	36.9821	8838545	68.072
80	2641813	38.3838	2365540	36.9821	1369	0
81	2641813	38.3838	2365540	36.9821	396	0
82	2641813	38.3838	2365540	36.9821	303	0
83	2641813	37.5739	1931298	32.2008	303	0
84	2641813	37.5739	1931298	32.2008	276	0

85	2641813	37.5739	1931298	32.2008	264	0
86	2285889	35.3695	1931298	32.2008	256	0
87	2285889	35.3695	1931298	32.2008	233	0
88	2285889	35.3695	570	0	212	0
89	2285889	35.3695	570	0	212	0
90	2285889	35.3695	570	0	212	0
91	2285889	35.3695	466	0	212	0
92	466	0	466	0	205	0
93	466	0	466	0	205	0
94	466	0	466	0	202	0
95	466	0	466	0	202	0
96	466	0	466	0	193	0
97	466	0	466	0	193	0
98	466	0	202	0	193	0
99	466	0	202	0	193	0
100	466	0	202	0	193	0
101	466	0	202	0	193	0
102	466	0	202	0	193	0
103	466	0	202	0	193	0
104	466	0	202	0	193	0
105	466	0	202	0	190	0
106	466	0	202	0	190	0
107	466	0	202	0	189	0
108	466	0	202	0	189	0
109	466	0	202	0	189	0
110	466	0	202	0	189	0
111	466	0	202	0	186	0
112	466	0	202	0	183	0
113	466	0	202	0	182	0
114	466	0	202	0	182	0
115	466	0	202	0	182	0
116	466	0	202	0	182	0
117	466	0	202	0	178	0
118	466	0	189	0	173	0
119	466	0	189	0	173	0
120	466	0	189	0	173	0
121	466	0	160	0	172	0
122	466	0	189	0	172	0
123	466	0	189	0	172	0
124	466	0	189	0	172	0

125	466	0	189	0	172	0
126	466	0	189	0	172	0
127	466	0	189	0	172	0
128	466	0	189	0	172	0
129	466	0	189	0	172	0
130	466	0	189	0	172	0
131	466	0	189	0	172	0
132	466	0	189	0	172	0
133	466	0	178	0	172	0
134	466	0	178	0	172	0
135	202	0	178	0	172	0
136	193	0	178	0	172	0
137	189	0	178	0	172	0
138	189	0	178	0	171	0
139	189	0	178	0	171	0
140	189	0	178	0	170	0
141	189	0	178	0	170	0
142	189	0	178	0	170	0
143	189	0	178	0	170	0
144	183	0	178	0	170	0
145	182	0	178	0	170	0
146	182	0	178	0	170	0
147	182	0	178	0	170	0
148	182	0	178	0	170	0
149	182	0	178	0	170	0
150	182	0	178	0	170	0
151	173	0	178	0	170	0
152	173	0	178	0	170	0
153	173	0	178	0	170	0
154	173	0	178	0	170	0
155	173	0	178	0	170	0
156	173	0	178	0	170	0
157	173	0	178	0	170	0
158	173	0	178	0	170	0
159	173	0	178	0	170	0
160	173	0	178	0	170	0
161	173	0	178	0	170	0
162	173	0	178	0	170	0
163	173	0	178	0	170	0
164	173	0	172	0	170	0

165	173	0	172	0	170	0
166	173	0	172	0	170	0
167	173	0	172	0	170	0
168	173	0	172	0	170	0
169	173	0	172	0	170	0
170	173	0	172	0	170	0
171	173	0	172	0	170	0
172	173	0	172	0	170	0
173	173	0	171	0	170	0
174	173	0	171	0	170	0
175	173	0	170	0	170	0
176	173	0	170	0	170	0
177	173	0	170	0	170	0
178	173	0	170	0	170	0
179	173	0	170	0	170	0
180	173	0	170	0	170	0
181	173	0	170	0	170	0
182	173	0	170	0	170	0
183	173	0	170	0	170	0
184	173	0	170	0	170	0
185	173	0	170	0	170	0
186	173	0	170	0	170	0
187	173	0	170	0	170	0
188	173	0	170	0	170	0
189	173	0	170	0	170	0
190	173	0	170	0	170	0
191	173	0	170	0	170	0
192	173	0	170	0	170	0
193	173	0	170	0	170	0
194	173	0	170	0	170	0
195	173	0	170	0	170	0
196	173	0	170	0	170	0
197	173	0	170	0	170	0
198	173	0	170	0	170	0
199	173	0	170	0	170	0
200	173	0	170	0	170	0

In the best sections found in TLBO and DE, the variables were the same. The algorithms selected for the columns OC2, OC3, IC2, and IC3 section number 31 from the design pool (Table 3.5), which corresponds to

HSS101.6x3.2, for the OC1 section number 32 from the design pool (Table 3.5), which corresponds to HHS101.6x4.8, and for IC1 section 36 from design pool (Table 3.5), which corresponds to HSS101.6x6.4. The optimum beam was selected for IB section number 3 from the design pool (Table 3.4), which corresponds to W200x15, and for the OB section number 2 from the design pool (Table 3.4), which corresponds to W150x13.5. but ABC was selected for the columns OC2, and OC3 section number 31 from the design pool (Table 3.5), which corresponds to HSS101.6x3.2, for the IC2,4, and3 section number 32 from the design pool (Table 3.5), which corresponds to HHS101.6x4.8, for the OC1 section number 36 from the design pool (Table 3.5), which corresponds to HHS101.6x6.4, and for the IC1 section number 39 from the design pool (Table 3.5), which corresponds to HHS114.3x4.8. The optimum beam was selected for IB section number 3 from the design pool (Table 3.4), which corresponds to W200x15, and for the OB section number 2 from the design pool (Table 3.4), which corresponds to W150x13.5. The optimum concrete strength for the three algorithms was the same 25 MPa. (Figure 5.1 to Figure 5.4) shows the running history for the composite column optimum design problem.

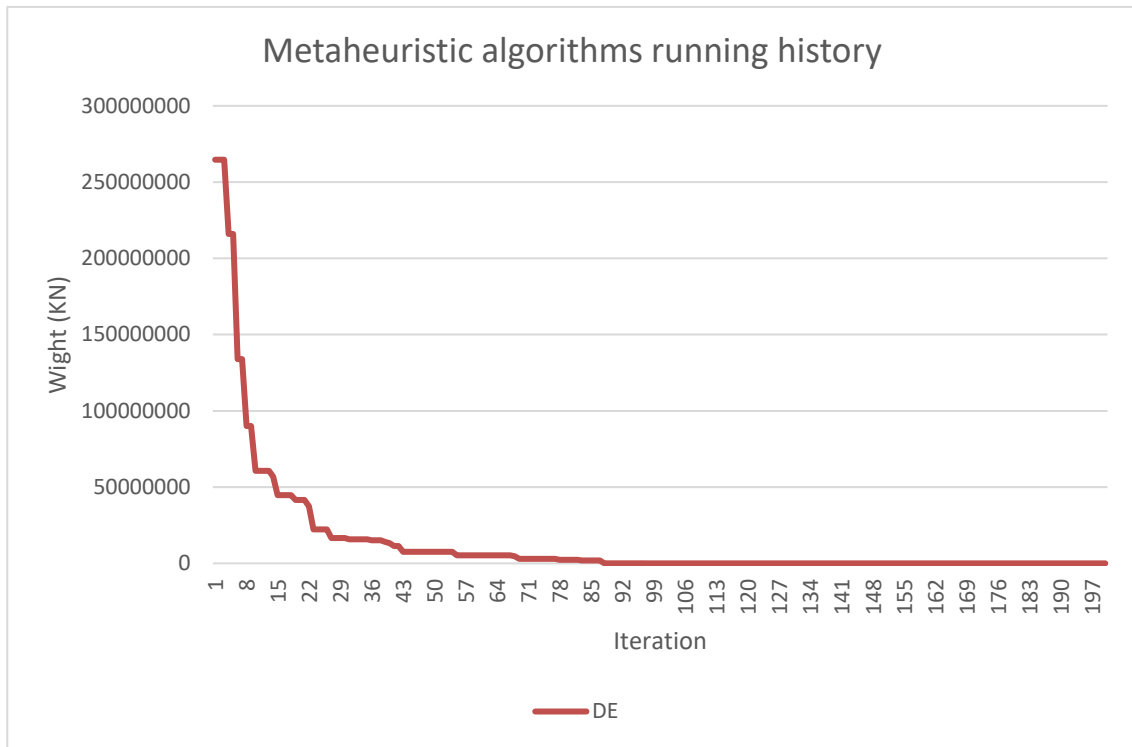


Figure 5.1 DE algorithm running history

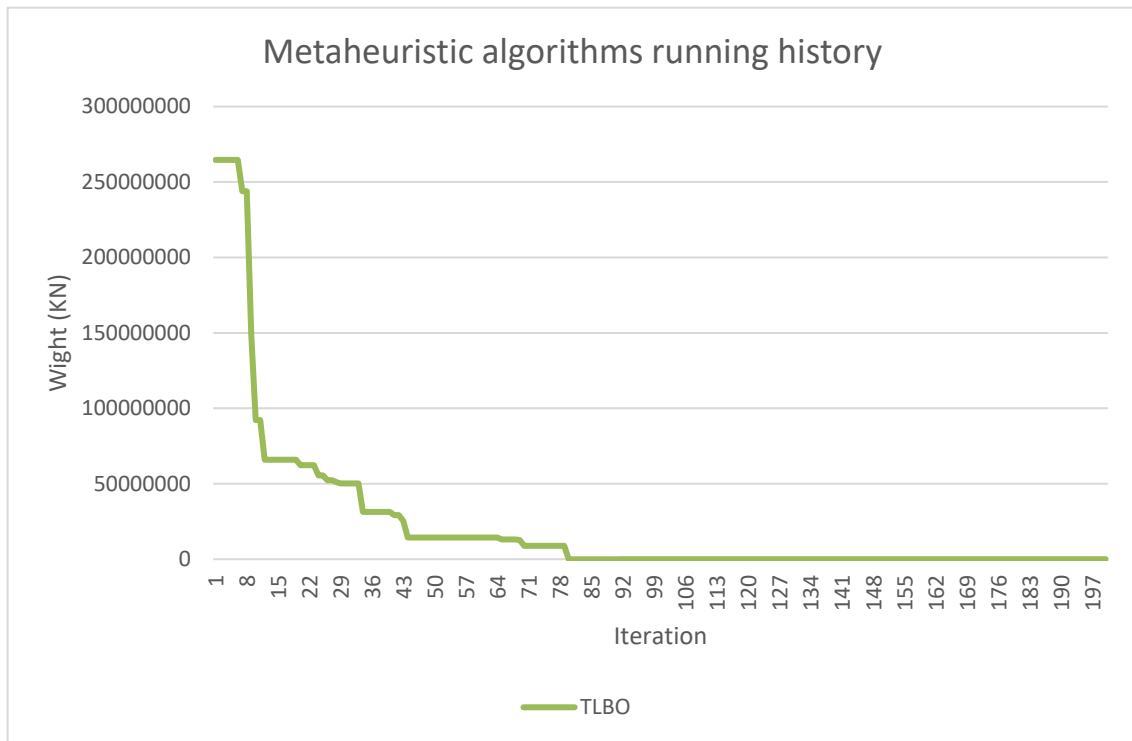


Figure 5.2 TLBO algorithm running history

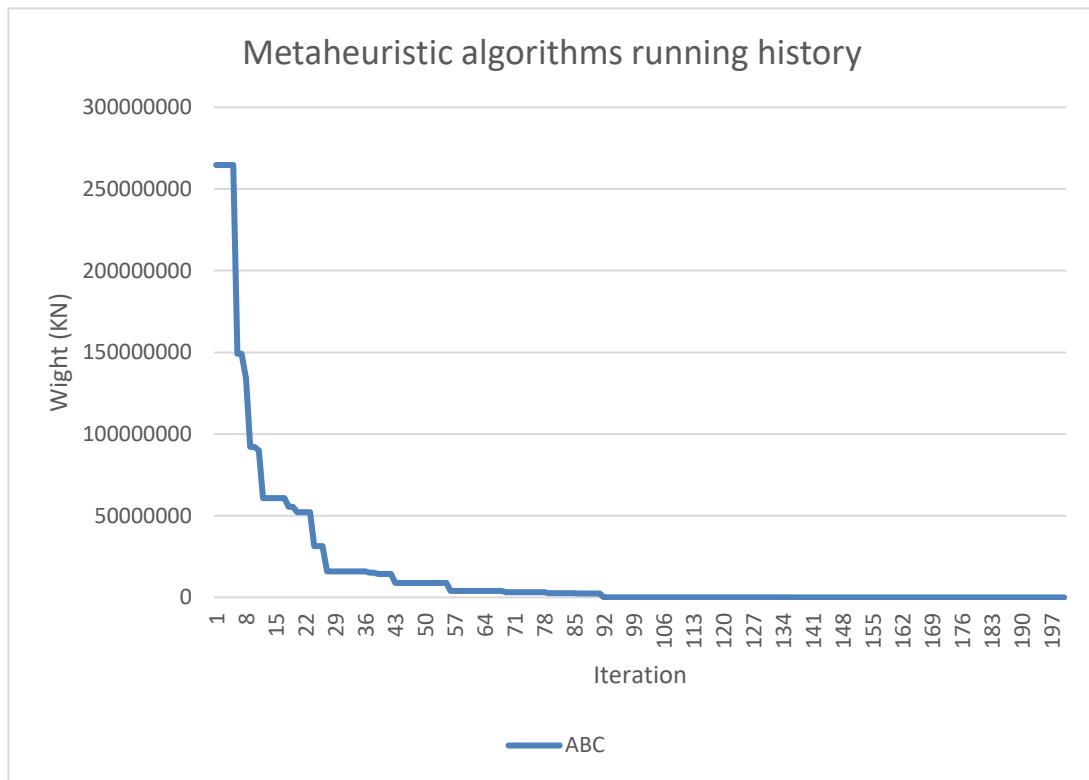


Figure 5.3 ABC algorithm running history

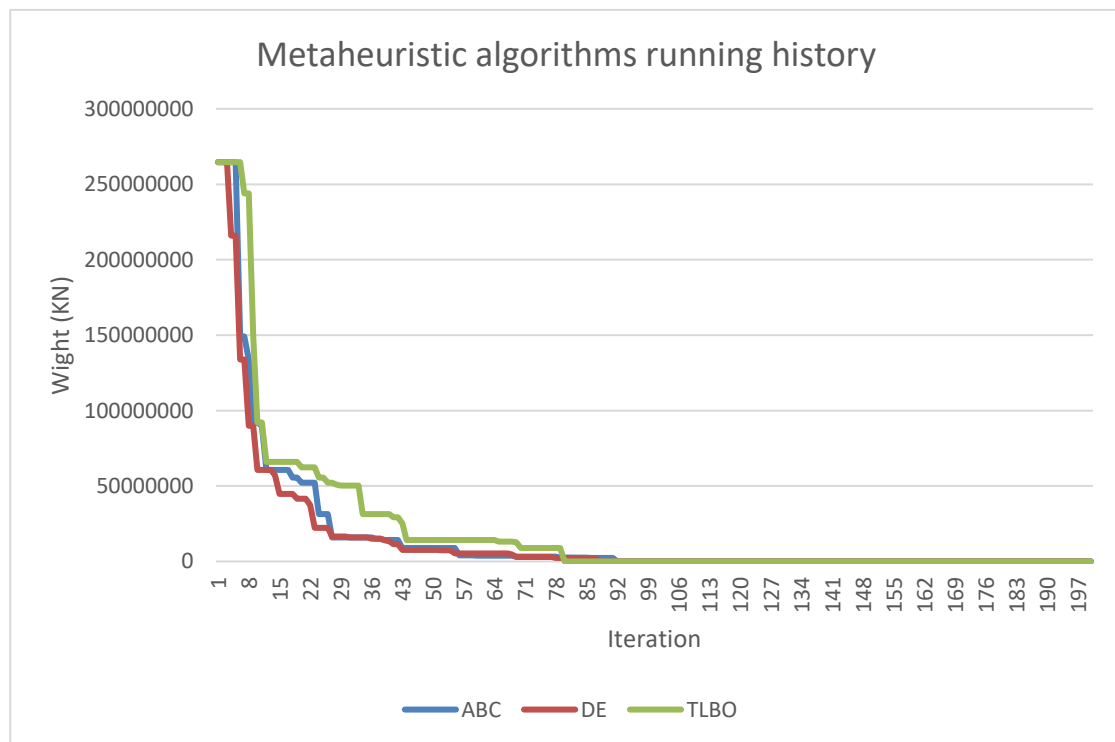


Figure 5.4 The three metaheuristic algorithms running history

Each of the three algorithms had an average running time of 19 days for ABC, 18 days for TLBO, and 10 days for De. It should be noted that all of the algorithms were run on the same computer.

Before reaching iteration number 175, the lightest optimum response for each of the two specified algorithms was discovered. For this design problem, 200 iterations were more than sufficient. It was discovered that TLBO converged faster than the other algorithms and found the best solution at the 140th iteration. DE and ABC achieved the best results at iterations 175 and 151, respectively.

Table 5.3, as well as the following Figure 5.5, depicts the outcomes of the design limitations. The table shows that the three constraints are dominating, and the numbers change until they reach zero, indicating that all of the identified constraints were the ruling variables in the design.

Table 5.3 Metaheuristic algorithms result for the composite column example

Constraint No.	Constraint Details	Equation No.	Maximum number	Minimum number
1	Strength Constrain	3.21 to 3.25	465.7042	0
2	Deflection Constrain	3.26	0.8006	0
3	Geometry Constraint	3.5,3.27	12	0

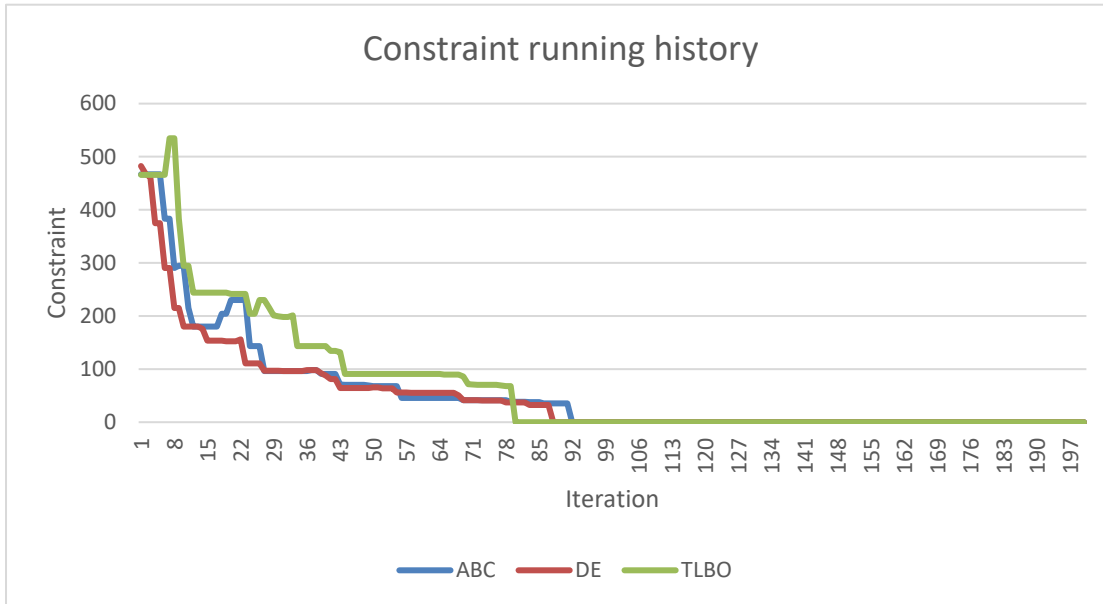


Figure 5.5 The three metaheuristic algorithms constraint running history

5.2 Hand Calculation

5.2.1 For the inner column on floor 1

Column section: HSS101.6x6.4

Mux max: 4.14 KN.m Muy max: 3.81 KN.m Pu max: 84.82 KN

$F'_c = 25 \text{ MPa}$, $f_y = 290 \text{ MPa}$, $E_s = 200000 \text{ MPa}$, $E_c = 4700 \sqrt{f'_c} = 23500 \text{ MPa}$

$\lambda_p = 0.09 E_s / f_y = 0.09 * 200000 / 290 = 62.07 \geq D/t = 101.6 / 6.4 = 15.875$

$\therefore$ Compact section,

For compact sections the compressive strength $P_{no} = P_p$

Where: $P_p = A_s f_y + C_2 f'_c A_c$

$C_2 = 0.95$ for round sections.

$$A_g = \frac{\pi (101.6)^2}{4} = 8107.31 \text{ mm}^2$$

$$A_c = \frac{\pi (101.6 - 6.4)^2}{4} = 7118.09 \text{ mm}^2$$

$$A_s = \frac{\pi (101.6)^2}{4} - \frac{\pi (101.6 - 6.4)^2}{4} = 989.22 \text{ mm}^2$$

$$P_p = P_{no} = 989.22 \times 290 + 0.95 \times 25 \times 7118.09 = 455.9 \text{ KN}$$

$$I_s = \frac{\pi (101.6^4 - (101.6 - 6.4)^4)}{64} = 1198544.64 \text{ mm}^4$$

$$I_s = \frac{\pi (101.6 - 6.4)^4}{64} = 4031973.715 \text{ mm}^4$$

$$C_3 = 0.45 + 3 (A_s/A_g) = 0.816 \leq 0.9$$

$$E_{leff} = E_s I_s + C_3 E_c I_c = 200000 \times 1198544.64 + 0.816 \times 23500 \times 4031973.715 \\ = 3.17 \times 10^{11} \text{ N.mm}^2$$

$$P_e = \frac{\pi^2 E_{leff}}{(KL)^2} = \frac{\pi^2 (3.17 \times 10^{11})}{(0.7 \times 5000)^2} = 255.4 \text{ KN}$$

$$\text{When } P_{no}/P_e (1.785) \leq 2.25$$

$$P_n = P_{no} (0.658^{(P_{no}/P_e)}) = 455.9 \times (0.658^{(1.785)}) = 215.97 \text{ KN}$$

$$P_u (84.82 \text{ KN}) \leq \phi P_n (0.75 \times 215.97 = 161.98 \text{ KN}) \therefore \text{compressive is ok}$$

For compact sections the flexural strength $M_n = M_p$

$$r_m = (d - t)/2 = 47.6 \text{ mm}$$

$$r_i = (d - 2t)/2 = 44.4 \text{ mm}$$

$$F_{CHS} = 0.1106$$

$$r_o = \pi/2 F_{CHS} = 0.1738$$

$$M_p = 4 \times 290 \times 6.4 \times 47.62 \cos(0.1738) + 0.95 (2/3) \times 25 \times 44.43 \cos 3(0.1738) \\ = 17.89 \text{ KN.m}$$

$$M_{ux} (4.14 \text{ KN.m}), M_{uy} (3.81 \text{ KN.m}) = \phi M_n (16.1 \text{ KN.m}) \therefore \text{flexural is ok}$$

Combined flexure and axial force (for compact sections)

$$\text{If } P_u/(\phi P_n) = (0.561) \geq 0.2$$

$$\frac{84.82}{161.98} + \frac{8}{9} \times \frac{4.14 + 3.81}{16.1} = 1 \leq 1 \therefore \text{Combined force is ok}$$

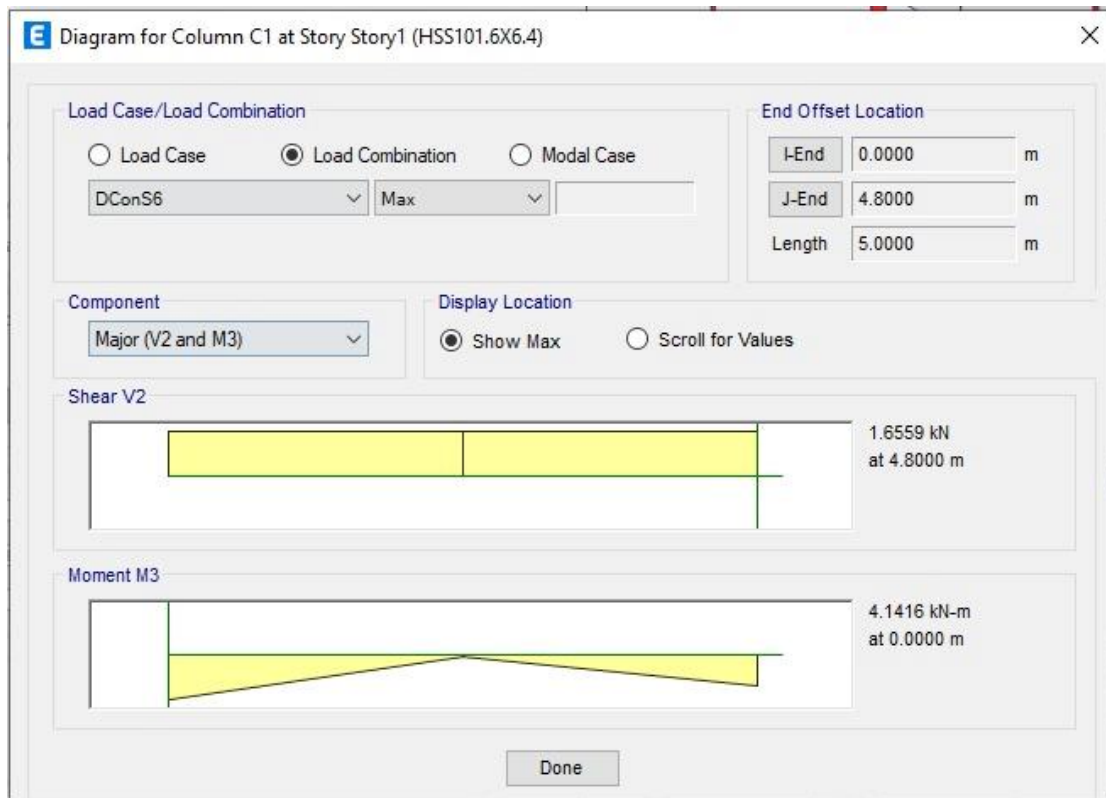


Figure 5.6 Inner column Mux max

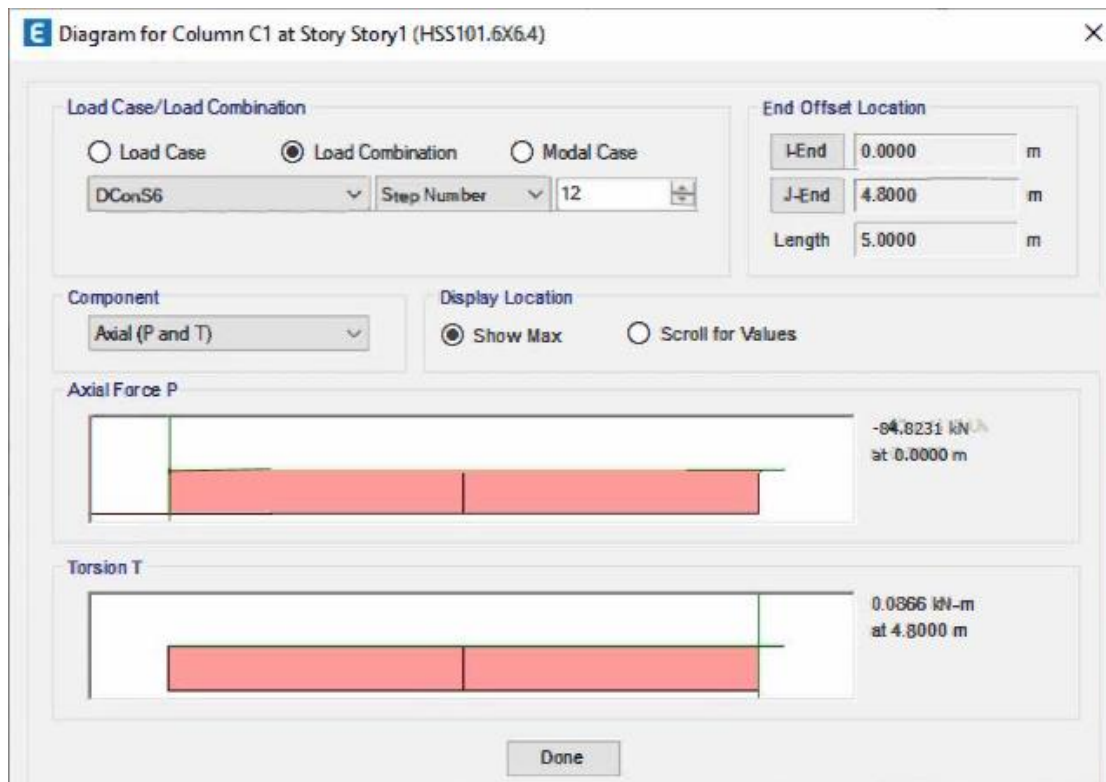


Figure 5.7 Inner column Pu max

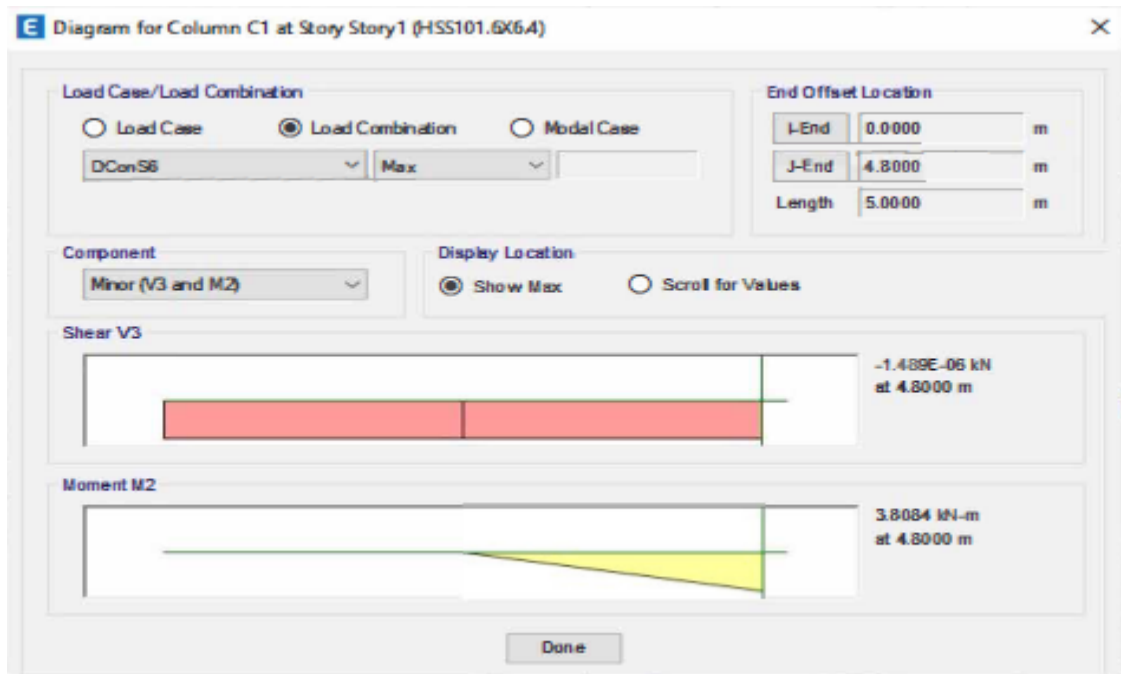


Figure 5.8 Inner column M_{uy} max

5.2.2 For the outer column on floor 1

Column section: HSS101.6x3.2

M_{ux} max: 3.33 kN.m M_{uy} max: 1.79 kN.m P_u max: 51 kN

$f'_c = 25 \text{ MPa}$, $f_y = 290 \text{ MPa}$, $E_s = 200000 \text{ MPa}$, $E_c = 4700\sqrt{f'_c} = 23500 \text{ MPa}$

$\lambda_p = 0.09E_s/f_y = 0.09 \times 200000/290 = 62.07 \geq D/t = 101.6/3.2 = 31.75$

$\therefore$ Compact section,

For compact sections the compressive strength $P_{no} = P_p$

Where: $P_p = A_s f_y + C_2 f'_c A_c$

$C_2 = 0.95$ for round sections.

$$A_g = \frac{\pi (101.6)^2}{4} = 8107.31 \text{ mm}^2$$

$$A_c = \frac{\pi (101.6 - 3.2)^2}{4} = 7604.66 \text{ mm}^2$$

$$A_s = \frac{\pi (101.6)^2}{4} - \frac{\pi (101.6 - 3.2)^2}{4} = 502.65 \text{ mm}^2$$

$$P_p = P_{no} = 502.65 \times 290 + 0.95 \times 25 \times 7604.66 = 326.4 \text{ kN}$$

$$I_s = \frac{\pi (101.6^4 - (101.6 - 3.2)^4)}{64} = 628479.38 \text{ mm}^4$$

$$I_s = \frac{\pi (101.6 - 3.2)^4}{64} = 4602038.975 \text{ mm}^4$$

$$C_3 = 0.45 + 3 (A_s/A_g) = 0.636 \leq 0.9$$

$$E I_{eff} = E_s I_s + C_3 E_c I_c = 200000 * 628479.38 + 0.636 * 23500 * 4602038.975 \\ = 1.94 \times 10^{11} \text{ N.mm}^2$$

$$P_e = \frac{\pi^2 E I_{eff}}{(KL)^2} = \frac{\pi^2 (1.94 * 10^{11})}{(0.7 * 5000)^2} = 156.69 \text{ KN}$$

$$\text{When } P_{no}/P_e (2.083) \leq 2.25$$

$$P_n = P_{no} (0.658^{(P_{no}/P_e)}) = 326.4 * (0.658^{(2.083)}) = 136.49 \text{ KN}$$

$$P_u (51 \text{ KN}) \leq \phi P_n (0.75 * 136.49 = 102.37 \text{ KN}) \therefore \text{compressive is ok}$$

For compact sections the flexural strength $M_n = M_p$

$$r_m = (d - t)/2 = 49.2 \text{ mm}$$

$$r_i = (d - 2t)/2 = 47.6 \text{ mm}$$

$$FCHS = 0.1893$$

$$r_o = \pi/2 FCHS = 0.2973$$

$$M_p = 4 * 290 * 3.2 * 49.22 \cos(0.2972) + 0.95(2/3) * 25 * 47.63 \cos^3(0.2973) \\ = 10.08 \text{ KN.m}$$

$$M_{ux} (3.33 \text{ KN.m}), M_{uy} (1.79 \text{ KN.m}) = \phi M_n (9.07 \text{ KN.m}) \therefore \text{flexural is ok}$$

Combined flexure and axial force (for compact sections)

$$\text{If } P_u/(\phi P_n) = (0.498) \geq 0.2$$

$$\frac{51}{102.37} + \frac{8}{9} * \frac{3.33 + 1.79}{9.07} = 1 \leq 1 \therefore \text{Combined force is ok}$$

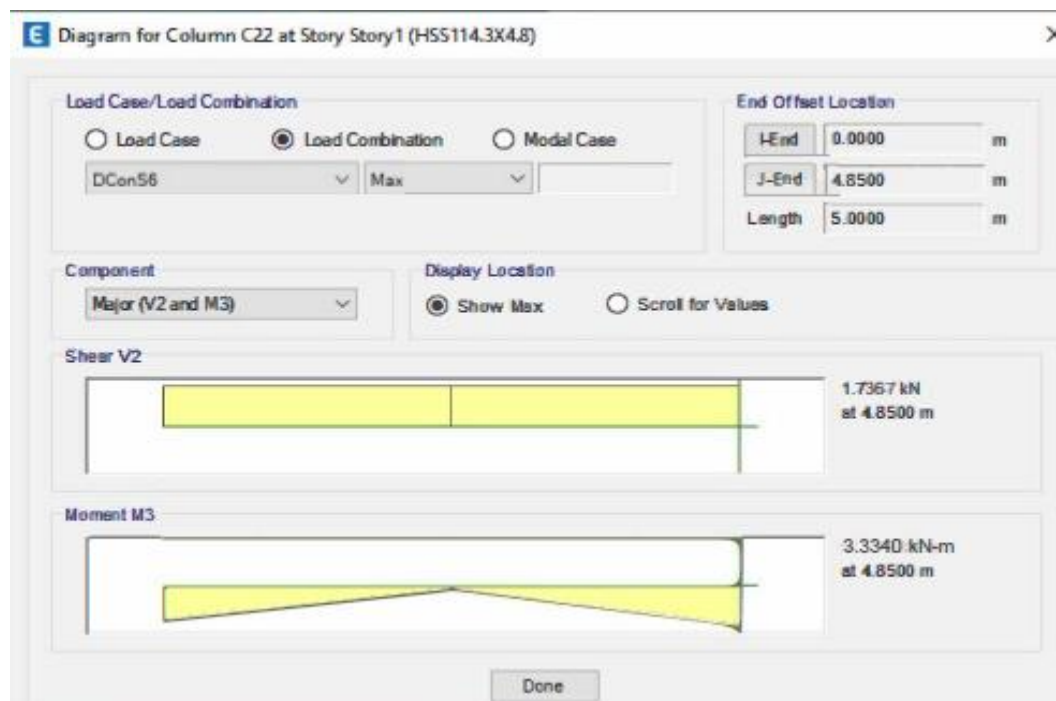


Figure 5.9 Outer column Mux max

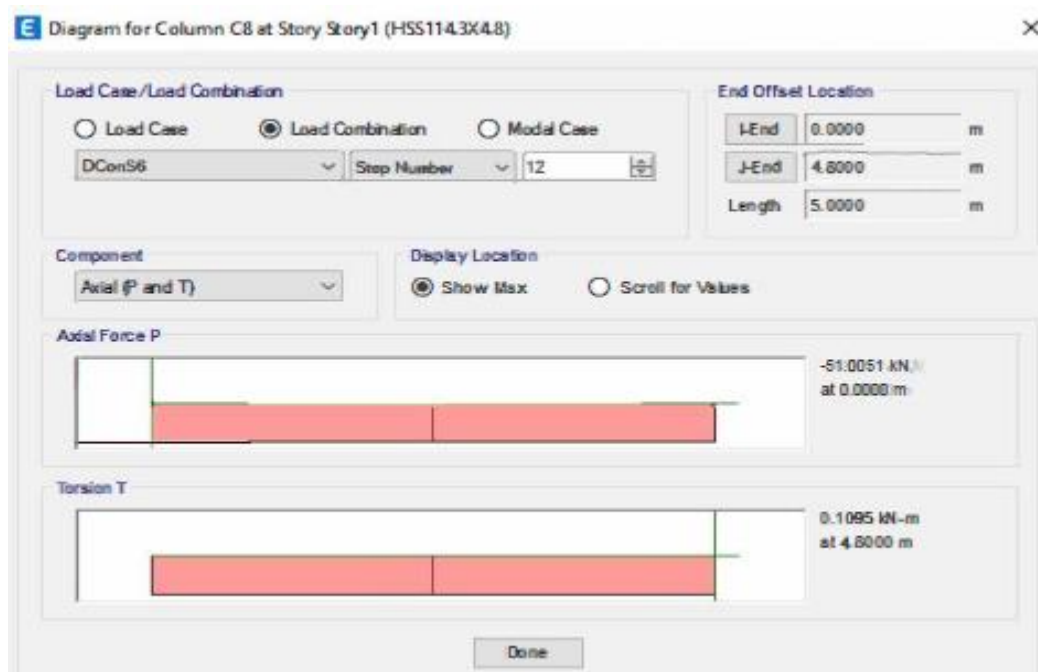


Figure 5.10 Outer column Pu max

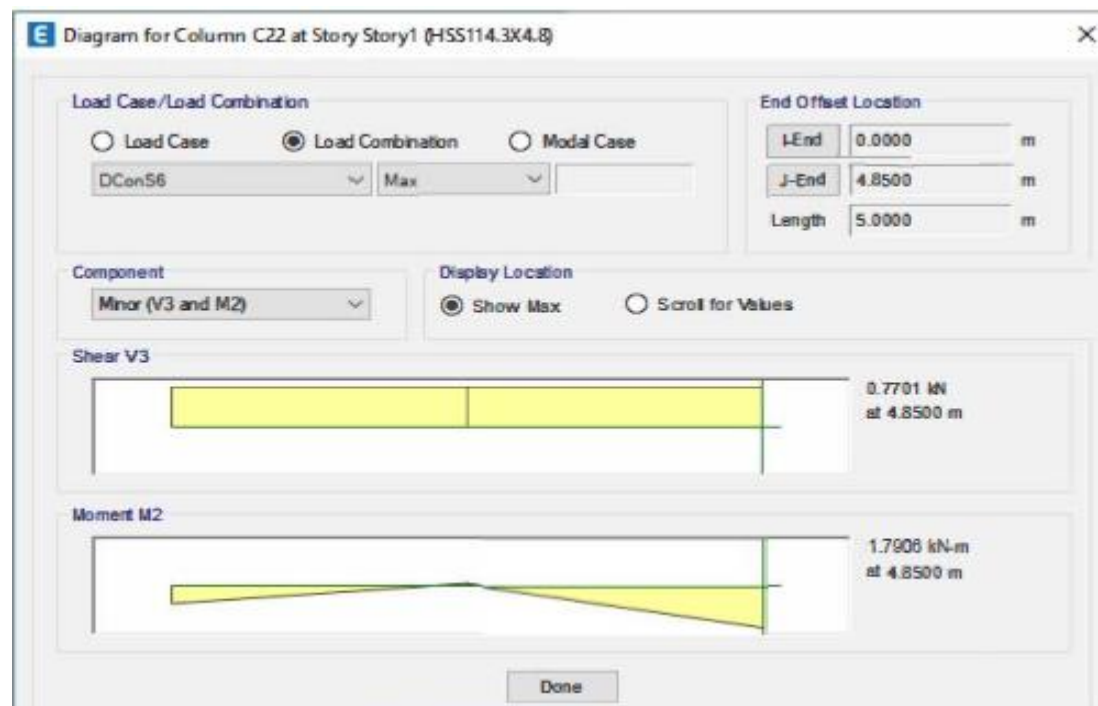


Figure 5.11 Outer column Muy max

5.3 Summary

This chapter discussed a specific form of optimization problem. The structural frame design was addressed using two variable cross-sections: a composite steel column filled with concrete and a steel I-section with concrete strength. The performance of these optimum design techniques is affected by the problem type, parameters, and metaheuristic algorithms used.

Chapter Six

Conclusion

6 Conclusions

This chapter is to describe the main findings of the research, as well as helpful recommendations for the upcoming and guidelines that have been followed in the research.

6.1 Main Findings

The Explanation of the main findings of this research will be through five achievements.

Objective 1

“To develop an optimum design algorithm for Steel Frames with Concrete Filled Columns that hasn’t been explained before in the lecturer”.

This research has been successfully to develop an optimum design algorithm for Steel Frames with concrete-filled columns following AISC slandered. These algorithms are able to improve the design problems by using updated metaheuristic algorithms by using interfacing between ETABS and MATLAB software. The developed algorithms are able to get the accurate dimensions and the properties of the optimum cross-section for Steel Frames with concrete-filled columns and I-steel sections. The mathematical modeling of such difficulties results in a discrete non-linear programming design problem with a high number of constraints. These types of problems were solved by employing several types of contemporary metaheuristic algorithms, and the outcomes were compared and analyzed. The structure was created as a three-dimensional model in ETABS. This sort of modeling accounts for the genuine influence of live loads. It also considers the unfavorable loading situation caused by the

various ways to load the design, which cannot be handled by two-dimensional modeling. Although the computational cost of constructing effect surfaces of structure activities under diverse loading circumstances is high, The ETABS-API capability provides an automated solution for this operation, allowing the design optimization algorithm to work with a single click and without additional user intervention.

Objective 2

“To obtain the optimum safe design in accordance with AISC, and ACI Composite Structure Design Specifications”

The optimal safe design of the structure elements was obtained in compliance with AISC, ACI Composite Structure Design Specifications. The following tables provide the ideal cross-sectional parameters for the composite steel column filled with concrete and steel I-beams:

Table 6.1 Optimum cross-sectional properties for the steel I-beams

Member Name	Sec. Name	Cross-section area (mm)	Depth (mm)	Width (mm)	Web thickness (mm)	Flange thickness (mm)
Inner Beam	W200X15	1910	200	100	4.32	5.21
Outer Beam	W150X13.5	1730	150	100	4.32	5.46

Table 6.2 Optimum cross-sectional properties for the composite columns

Member Name	Sec. Name	Diameter (mm)	wall thickness (mm)
Inner column First Floor	HSS101.6x6.4	101.6	6.4
Outer column First Floor	HSS101.6x4.8	101.6	4.8
Inner column Second Floor	HSS101.6x3.2	101.6	3.2
Outer column Second Floor	HSS101.6x3.2	101.6	3.2
Inner column Third Floor	HSS101.6x3.2	101.6	3.2
Outer column Third Floor	HSS101.6x3.2	101.6	3.2

Objective 3

“To achieve the minimum weight design for the selected frame problem”

The minimum weight found for the frame structure was 170KN.

Objective 4

“To choose some of the most modern metaheuristic algorithms that have not previously been employed in the literature on composite steel column design optimization and investigate their performance in achieving the best solution.”

Three metaheuristic algorithms were used to solve the example of the optimal design of a frame structure: ABC, TLBO, and DE. Except for the ABC, all of these algorithms worked admirably in terms of producing the best results. In terms of computational time, ABC was the worst, taking 19 days, and TLBO was taking 18 days, while DE was the best, taking 10 days. As a result, DE is suggested for this type of situation, followed by TLBO.

Objective 5

“To investigate the ability of these metaheuristic algorithms to solve high-dimensional issues”

Each metaheuristic algorithm's competence was evaluated by solving frame structure examples. To execute the load analysis, which necessitates the building of a high number of variables, frame structure design optimization problems necessitate computers with significant RAM and storage capacity. While solving the problem, the DE algorithm performs less structural analysis than TLBO and ABC. If more advanced computers were available, the usage of TLBO might be suitable. ABC should be tested further for a high number of spans and floor frame design optimization challenges if more powerful computers are available.

6.2 Recommendations for Future Work

From this thesis, the following recommendations for future work are suggested:

- Testing the performance of the updated EABC algorithm by testing more of different types of structure.
- Test the quality of DE in solving high-rise building structure design optimization problems such as the building Structure.
- Use a combination of metaheuristic algorithms to improve the quality of the outcomes and computation time such as TLBO and ABC.
- Upgrade the TLBO by reducing the calculating period to provide the result faster.
- Identify more variables for the Steel Frames with concrete-filled columns problem optimization, slab thickness, and steel yield strength (f_y).
- Solving structure design optimization for composite columns using both hollow square and hollow circular sections.
- Solve an optimization design problem for high rise buildings.
- The connection between the steel members in the structure has to be optimized.
- Total costs such as material costs, labor charges, and installation charges have to be considered as objective functions after the design variables.
- Material type and specification have to be considered as variables.

- Other frame structure types which are not mentioned in the literature review such as composite steel columns and beams covered with concrete.
- Optimization process has to be applied on some high-rise buildings in Bahrain or out in order to make a comparison between the outputs with the actual design to check if there are any changes in the saving cost.
- Optimization algorithms have to be developed for full steel structure frame problems, by including the design of superstructure and foundations.
- The initial inputs have to be constant by the user in all the algorithms to guarantee, that the outputs were coming through the same start points in all the algorithms, and by that, the comparison will be accurate, and we can decide which one is the most efficient for these kinds of problems.

6.3 Recommendations and Guidelines for this Research

Providing More guidelines and recommendations for the work that has been done in the research.

- The optimum design algorithm for Steel Frames with Concrete Filled Columns can be used on site directly.
- The frame structure's design method provides cross-section dimensions, which are utilized to create built-up sections that meet ACI and AISC criteria. The market's practical proportions could be chosen as the design pool. If the algorithm includes the design of the

connections, the optimum solution derived by the program can be easily utilized in real-life design.

- The research applied one type of structure and constant value of floor numbers and bay dimensions and quantity. This is because of the time and my try to get the highest quality of results. But at the same time, the theses can be applied to other numbers of floors and other bay's dimensions and quantity.
- The algorithm which developed needs the structure models to set up in ETABS. By using API models, the algorithm can be general for other types of steel structure frames. Also, it can be general for different span sizes and numbers.
- This development is limited to steel structure frame design problem optimization. But at the same time, optimization can be used in infrastructure.
- This development can be used with any steel structure frame design integrated with an analysis software that has used API commends. For example, instead of using ETABS, SAP2000 can be used.

References

- Akin, A., & Saka, M. P. (2015). Harmony search algorithm based optimum detailed design of reinforced concrete plane frames subject to ACI 318-05 provisions. *Computers & Structures*, 147, 79–95. <https://doi.org/10.1016/J.COMPSTRUC.2014.10.003>
- Asst. Prof. Awadh E Ajeel. (2021). *AASHTO LRFD Bridge Design Philosophy*. 26–30.
- Aydodu, I., & Saka, M. P. (2012). Ant colony optimization of irregular steel frames including elemental warping effect. *Advances in Engineering Software*, 44(1), 150–169. <https://doi.org/10.1016/j.advengsoft.2011.05.029>
- Aydodu, I., Akin, A., & Saka, M. P. (2016). Design optimization of real world steel space frames using artificial bee colony algorithm with Levy flight distribution. *Advances in Engineering Software*, 92, 1–14. <https://doi.org/10.1016/j.advengsoft.2015.10.013>
- Aviation Week Network (2021). *Bahrain International Airport Looks To New Terminal To Increase Traffic*. (n.d.). <https://Aviationweek.Com/Air-Transport/Airports-Networks/Bahrain-International-Airport-Looks-New-Terminal-Increase-Traffic>.
- Bekdaş, G., & Nigdeli, S. M. (2015). *Optimization of Reinforced Concrete Columns Subjected to Uniaxial Loading* (pp. 399–412). https://doi.org/10.1007/978-3-319-18320-6_21
- CSI Computer and Structures. (2020). *ETABS*.
- CSI Knowledge Base. (2020). *CSI Knowledge Base*.
- Construction Cost. <https://constructioncost.co/different-types-columns.html>
- Dubey, M., Kumar, V., Kaur, M., & Dao, T. P. (2021). A Systematic Review on Harmony Search Algorithm: Theory, Literature, and Applications. In *Mathematical Problems in Engineering* (Vol. 2021). Hindawi Limited. <https://doi.org/10.1155/2021/5594267>

- Eren, Y., Küçükdemiral, I. B., & Üstoğlu, I. (2017). Introduction to Optimization. *Optimization in Renewable Energy Systems: Recent Perspectives*, 27–74. <https://doi.org/10.1016/B978-0-08-101041-9.00002-8>
- Eurocode 4: Design of composite steel and concrete structures Part 1-1: General rules and rules for buildings*. (2004).
- G. Zhu and S. Kwong. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation*, Vol. 217, No. 7, Pp. 3166–3173.
- 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>
- Kaveh, A., Kamalinejad, M., Arzani, H., & Barzinpour, F. (2021). New enhanced colliding body optimization algorithm based on a novel strategy for exploration. *Journal of Building Engineering*, 43, 102553. <https://doi.org/10.1016/J.JOBE.2021.102553>
- Kaveh, A., & Mahdavi, V. R. (2019). Multi-objective colliding bodies optimization algorithm for design of trusses. *Journal of Computational Design and Engineering*, 6(1), 49–59. <https://doi.org/10.1016/J.JCDE.2018.04.001>
- Kim, J. H. (2016). Harmony Search Algorithm: A Unique Music-inspired Algorithm. *Procedia Engineering*, 154, 1401–1405. <https://doi.org/10.1016/J.PROENG.2016.07.510>
- Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598), 671–680. <https://doi.org/10.1126/science.220.4598.671>
- Kumar, A., Chatterjee, J. M., Payal, M., & Rathore, P. S. (2022). Revolutionizing the internet of things with swarm intelligence. *System Assurances: Modeling and Management*, 403–436. <https://doi.org/10.1016/B978-0-323-90240-3.00023-0>
- Lu, C., Yuan, H., & Zhang, N. (2023). Nanophotonic devices based on optimization algorithms. *Intelligent Nanotechnology: Merging*

- Nanoscience and Artificial Intelligence*, 71–111.
<https://doi.org/10.1016/B978-0-323-85796-3.00004-4>
- Lute, V., Upadhyay, A., & Singh, K. K. (2011). Genetic Algorithms-based Optimization of Cable Stayed Bridges. *Journal of Software Engineering and Applications*, 04(10), 571–578.
<https://doi.org/10.4236/jsea.2011.410066>
- M. Fahim (2022). *Composite Columns in Building Structure - A Analytical Study*, Material Science Research India.
- M.A.Latif (2019). *Design Optimization of Bridge Structures under Code Provisions using Metaheuristic Algorithms*.
- Marinaki, M., & Marinakis, Y. (2016). A Glowworm Swarm Optimization algorithm for the Vehicle Routing Problem with Stochastic Demands. *Expert Systems with Applications*, 46, 145–163.
<https://doi.org/10.1016/J.ESWA.2015.10.012>
- Marini, F., & Walczak, B. (2015). Particle swarm optimization (PSO). A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 149, 153–165. <https://doi.org/10.1016/J.CHEMOLAB.2015.08.020>
- MathWorks. (2019). *MATLAB*.
- Mohammad, F. A., & Seyan, D. A. (2016). *Optimum Design of Reinforced Concrete Rectangular Columns Subjected to Axial Compression and Biaxial Bending Moments*.
- Mostapha Kalami Heris. (2015a). *Artificial Bee Colony in MATLAB*. Yarpiz.
- Mostapha Kalami Heris. (2015b). *Differential Evolution (DE) in MATLAB*. Yarpiz.
- Mostapha Kalami Heris. (2015c). *Teaching-Learning-based Optimization in MATLAB*. Yarpiz.
- Optimum design of reinforced concrete columns employing teaching-learning based optimization. (2016). *Challenge Journal of Structural Mechanics*. <https://doi.org/10.20528/cjsmec.2016.11.030>
- Papanicolaou, C. G., & Papantoniou, I. C. (2016). Optimum Design of Textile-Reinforced Concrete as Integrated Formwork in Slabs.

Textile Fibre Composites in Civil Engineering, 245–274.

<https://doi.org/10.1016/B978-1-78242-446-8.00012-4>

Pires, S. de L., & Silva, M. C. A. T. da. (2021). Optimization of slender reinforced concrete columns subjected to biaxial bending using genetic algorithms. *Revista IBRACON de Estruturas e Materiais*, 14(6). <https://doi.org/10.1590/s1983-41952021000600010>

R. V. Rao, V. J. S. and D. P. V. (2011). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43, 303–315.

R. W. Furlong. (2000). Should I Use ACI 318 or LRFD for Designing Composite Columns? *Symposium Paper*, 196, 125–140.

Rafiq, M. Y., & Southcombe, C. (1998). Genetic algorithms in optimal design and detailing of reinforced concrete biaxial columns supported by a declarative approach for capacity checking. *Computers & Structures*, 69(4), 443–457.
[https://doi.org/10.1016/S0045-7949\(98\)00108-4](https://doi.org/10.1016/S0045-7949(98)00108-4)

Ramesh konchada, & Bruce Pelletier. (2023). *Types Of Columns Used In Construction*. DAILY CIVIL.

Rao, R. V., & Patel, V. (2013a). An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems. *Scientia Iranica*, 20(3), 710–720.
<https://doi.org/10.1016/J.SCIENT.2012.12.005>

Rao, R. V., & Patel, V. (2013b). An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems. *Scientia Iranica*, 20(3), 710–720.
<https://doi.org/10.1016/J.SCIENT.2012.12.005>

Saka, M. P., Doğan, E., & Aydogdu, I. (2013). Analysis of Swarm Intelligence–Based Algorithms for Constrained Optimization. *Swarm Intelligence and Bio-Inspired Computation*, 25–48.
<https://doi.org/10.1016/B978-0-12-405163-8.00002-8>

Saka, M. P., & Geem, Z. W. (2013). Mathematical and metaheuristic applications in design optimization of steel frame structures: An extensive review. In *Mathematical Problems in Engineering* (Vol. 2013). <https://doi.org/10.1155/2013/271031>

- Singh Gill, H., Singh Khehra, B., Singh, A., & Kaur, L. (2019). Teaching-learning-based optimization algorithm to minimize cross entropy for Selecting multilevel threshold values. *Egyptian Informatics Journal*, 20(1), 11–25.
<https://doi.org/10.1016/J.EIJ.2018.03.006>
- Singiresu S. Rao. (2019). *Engineering Optimization Theory and Practice* (Fifth Edition). John Wiley and Sons, Inc.
- The Treasury; Petra, Jordan*. (2008, December).
<https://edtrayes.com/2008/12/the-treasury-in-context-petra-jordan/>.
- Types of columns on the basis of shape, reinforcement, loading, slenderness ratio*. (n.d.). <https://constructioncost.co/different-types-columns.html>.
- Wu, D., Wang, S., Liu, Q., Abualigah, L., & Jia, H. (2022). An Improved Teaching-Learning-Based Optimization Algorithm with Reinforcement Learning Strategy for Solving Optimization Problems. *Computational Intelligence and Neuroscience*, 2022.
<https://doi.org/10.1155/2022/1535957>
- Xin-She Yang, & Suash Deb. (2009). *Cuckoo Search*.
- Yang, J., & Peng, Z. (2018). Improved abc algorithm optimizing the bridge sensor placement. *Sensors (Switzerland)*, 18(7).
<https://doi.org/10.3390/s18072240>
- Yang, X.-S. (2014). Analysis of Algorithms. *Nature-Inspired Optimization Algorithms*, 23–44. <https://doi.org/10.1016/B978-0-12-416743-8.00002-6>
- Yang, X.-She. (2010). *Nature-inspired metaheuristic algorithms*. Luniver Press.
- Z. W. Geem, J. H. Kim, & G. Loganathan. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2), 60–68.

Appendix A

Column Design

MATLAB Code

Appendix A Column Design MATLAB Code

This appendix includes the applications of API functions in the structure steel frames design optimization algorithm. These examples were extracted from the algorithms developed for the composite steel column filled with concrete.

A.1 Section Assigns, Analysis Run, Calculating the Constraints, and the Weight Function

```
function [Weight,c,sc,dc,gc] = WeightFunction(x,SapModel)
%% Weight Parameters
%Specific weight of the steel
gammaC=23.5631*10^-9;
gammaS=76.9729*10^-9;
%Total Length for each member
IC1=45000;
OC1=80000;
IC2=45000;
OC2=80000;
IC3=45000;
OC3=80000;
IB=360000;
OB=240000;
% Names of each group
fname=["IC1" "OC1" "IC2" "OC2" "IC3" "OC3" "IB" "OB"];
% Names of each load case
LCombo=["DCmpC1" "DCmpC2" "DCmpD1" "DCmpD2" "DCmpS1"
"DCmpS2" "DConS2" "DConS3" "DConS4" "DConS5" "DConS6"
"DConS7" "DConS8" "DConS9" "DConS10"];
c=0; % Constraints Summation
sc=0; % Strength Constraints
dc=0; % Deflection Constraints
gc=0; % Geometry Constraints

%% Defining the frame sections
ret = SapModel.SetPresentUnits(ETABSV1.eUnits.kN_mm_C);
ret = SapModel.SetModelIsLocked(false);
%Allowable Deflection
Dallow=5000/360;
% Defining Material
PropMaterial =
NET.explicitCast(SapModel.PropMaterial,'ETABSV1.cPropMaterial');
fc=((x(:,9)-1)*5+20);
fy=290;
Ec=4700*sqrt(fc);
Es=200000;
ret = PropMaterial.SetConcrete("4000Psi", fc/1000 , false,
0, 1, 2, 0.003, 0.004);
% Defining Sections
```

```

for j=1:8
    if j<7 %Defining section of Columns
        FrameObj
NET.explicitCast(SapModel.FrameObj, 'ETABSV1.cFrameObj');
        [d(j),th(j),n] = Csection(x(:,j));
        ret = FrameObj.SetSection(frname(j), n,
ETABSV1.eItemType.Group);
        ag(j)=pi()*d(j)^2/4;
        ar(j)=pi()*d(j)^2/4-pi()*(d(j)-th(j))^2/4;
        ar(j+8)=pi()*(d(j)-th(j))^2/4;
        %Defining sections of Beam
    else
        [ar(j),db,b,f,w] = Wsection(x(:,j));
        PropFrame
NET.explicitCast(SapModel.PropFrame, 'ETABSV1.cPropFrame');
        ret
PropFrame.SetISection(frname(j), 'A992Fy50', db,b,f,w,b,f);
        ret = FrameObj.SetSection(frname(j), frname(j),
ETABSV1.eItemType.Group);
        gc=gc+max(0,b/d(1)-1);
        gc=gc+max(0,b/d(2)-1);
        gc=gc+max(0,b/d(3)-1);
        gc=gc+max(0,b/d(4)-1);
        gc=gc+max(0,b/d(5)-1);
        gc=gc+max(0,b/d(6)-1);
    end
end

%% Deflection Results
Analyze
NET.explicitCast(SapModel.Analyze, 'ETABSV1.cAnalyze');
AnalysisResults
NET.explicitCast(SapModel.Results, 'ETABSV1.cAnalysisResults'
);
AnalysisResultsSetup
NET.explicitCast(AnalysisResults.Setup, 'ETABSV1.cAnalysisRes
ultsSetup');
    NumberResults = 0;
    Obj = NET.createArray('System.String',2);
    ObjSta = NET.createArray('System.Double',2);
    Elm = NET.createArray('System.String',2);
    ElmSta = NET.createArray('System.Double',2);
    ACase = NET.createArray('System.String',2);
    LoadCase = NET.createArray('System.String',2);
    StepType = NET.createArray('System.String',2);
    StepNum = NET.createArray('System.Double',2);
    U1 = NET.createArray('System.Double',2);
    U2 = NET.createArray('System.Double',2);
    U3 = NET.createArray('System.Double',2);
    R1 = NET.createArray('System.Double',2);
    R2 = NET.createArray('System.Double',2);
    R3 = NET.createArray('System.Double',2);
    P = NET.createArray('System.Double',2);
    V2 = NET.createArray('System.Double',2);
    V3 = NET.createArray('System.Double',2);
    T = NET.createArray('System.Double',2);
    M2 = NET.createArray('System.Double',2);
    M3 = NET.createArray('System.Double',2);
ret = Analyze.RunAnalysis();

```

```

for z=1:6
    ret = SapModel.SetPresentUnits(ETABSV1.eUnits.kN_mm_C);
    ret
AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput;
    ret
AnalysisResultsSetup.SetComboSelectedForOutput(LCombo(z));
    [ret, NumberResults, Obj, Elm, ACase, StepType, StepNum,
    U1,      U2,      U3,      R1,      R2,      R3] =
AnalysisResults.JointDisplAbs("ALL",
ETABSV1.eItemTypeElm.GroupElm, NumberResults, Obj, Elm,
ACase, StepType, StepNum, U1, U2, U3, R1, R2, R3);
    Dmaxv(z)=max(abs(double(U3)));
    Dmaxh(z)=max([-1*min(double(U1)),max(double(U1)),-
1*min(double(U2)),max(double(U2))]);
    for j=1:6
        if (d(j)/th(j))<=(0.09*Es/fy)
            Pno=(ar(j)*fy+0.95*fc*ar(j+8))/1000;
            Is=pi()* (d(j)^4-(d(j)-th(j))^4);
            Ic=pi()* ((d(j)-th(j))^4);
            if (0.45+3*(ar(j)/ag(j)))<=0.9
                C3=0.45+3*(ar(j)/ag(j));
            else
                C3=0.9;
            end
            EIEff=Es*Is+C3*Ec*Ic;
            Pe=(pi()^2*EIEff/(0.7*5000)^2)/1000;
            if Pno/Pe <= 2.25
                Pn=Pno*0.658^(Pno/Pe);
            else
                Pn=0.877*Pe;
            end
            Fchs=(1/8*0.95*fc/fy*(d(j)-
2*th(j))/th(j))/(1+1/4*0.95*fc/fy*(d(j)-2*th(j))/th(j));
            rm=(d(j)-th(j))/2;
            ri=(d(j)-2*th(j))/2;
            ro=pi()*Fchs/2;

Mn=(4*fy*th(j)*rm^2*cos(ro)+0.95*2/3*fc*ri^3*(cos(ro))^3)/10
00;

            [ret, NumberResults, Obj, ObjSta, Elm, ElmSta,
LoadCase, StepType, StepNum, P, V2, V3, T, M2, M3] =
AnalysisResults.FrameForce(frname(j),
ETABSV1.eItemTypeElm.GroupElm, NumberResults, Obj, ObjSta,
Elm, ElmSta, LoadCase, StepType, StepNum, P, V2, V3, T, M2,
M3);

            Pu=max(abs(double(P)));
            Mux=max(abs(double(M3)));
            Muy=max(abs(double(M2)));
            if (Pu/(0.75*Pn)) >= 0.2

sc=sc+max(0, (Pu/(0.75*Pn))+(8/9)*((Mux+Muy)/(0.9*Mn))-1);
            else

sc=sc+max(0, (Pu/(2*0.75*Pn))+((Mux+Muy)/(0.9*Mn))-1);
            end
            else
                gc=gc+1;
            end
        end
    end
end

```

```

end
    dc=dc+max(0,max(Dmaxv)/Dallow-1);

    dc=dc+max(0,max(Dmaxh)/Dallow-1);

%% Obtaining the Strength Constraints
ret
AnalysisResultsSetup.DeselectAllCasesAndCombosForOutput;
DesignSteel
NET.explicitCast(SapModel.DesignSteel,'ETABSv1.cDesignSteel'
);
ret = DesignSteel.StartDesign();
for j=7:8
ret = SapModel.SetPresentUnits(ETABSv1.eUnits.kN_mm_C);
NumberItems = 0;
FrameType = NET.createArray('System.String',2);
DesignSect = NET.createArray('System.String',2);
Status = NET.createArray('System.String',2);
PMMCombo = NET.createArray('System.String',2);
PMMRatio = NET.createArray('System.Double',2);
PRatio = NET.createArray('System.Double',2);
MMajRatio = NET.createArray('System.Double',2);
MMinRatio = NET.createArray('System.Double',2);
VMajCombo = NET.createArray('System.String',2);
VMajRatio = NET.createArray('System.Double',2);
VMinCombo = NET.createArray('System.String',2);
VMinRatio = NET.createArray('System.Double',2);
[ret,NumberItems, FrameType, DesignSect, Status, PMMCombo,
PMMRatio, PRatio, MMajRatio, MMinRatio, VMajCombo,
VMajRatio, VMinCombo, VMinRatio]=DesignSteel.GetSummaryResults_2(frname(j),NumberI
tems, FrameType , DesignSect, Status, PMMCombo, PMMRatio,
PRatio, MMajRatio, MMinRatio, VMajCombo, VMajRatio,
VMinCombo, VMinRatio, ETABSv1.eItemType.Group);
sc=sc+max(0,max(double(PMMRatio))-1);
sc=sc+max(0,max(double(VMajRatio))-1);
sc=sc+max(0,max(double(VMinRatio))-1);
end

%% Output
W=gammaS.*(IC1.*ar(1)+OC1.*ar(2)+IC2.*ar(3)+OC2.*ar(4)+IC3.*
ar(5)+OC3.*ar(6)+IB.*ar(7)+OB.*ar(8))+gammaC.*(IC1.*ar(9)+OC
1.*ar(10)+IC2.*ar(11)+OC2.*ar(12)+IC3.*ar(13)+OC3.*ar(14));
c=gc+sc+dc;
Weight=W.*(1+c).^2;
end

```

A.2 Column Section Function

```
function [d,t,n] = Csection(i)
%unit is mm

cname=["HSS42.2x3.6"      "HSS48.3x3"      "HSS48.3x3.7"
"HSS48.3x4.8"      "HSS60.3x3.2"      "HSS60.3x3.9"
"HSS60.3x4.8"      "HSS60.3x5.5"      "HSS60.3x6.4"
"HSS63.5x3.2"      "HSS63.5x4.8"      "HSS63.5x6.4"      "HSS73x3.2"
"HSS73x4.8"      "HSS73x5.2"      "HSS73x6.4"      "HSS76.2x3.2"
"HSS76.2x3.4"      "HSS76.2x3.9"      "HSS76.2x4.8"
"HSS76.2x5.2"      "HSS76.2x5.5"      "HSS76.2x6.4"
"HSS88.9x3.2"      "HSS88.9x4.8"      "HSS88.9x5.2"
"HSS88.9x5.5"      "HSS88.9x6.4"      "HSS88.9x7.6"      "HSS88.9x8"
"HSS101.6x3.2"      "HSS101.6x4.8"      "HSS101.6x5.6"
"HSS101.6x5.7"      "HSS101.6x6"      "HSS101.6x6.4"      "HSS101.6x8"
"HSS114.3x3.2"      "HSS114.3x4.8"      "HSS114.3x6"
"HSS114.3x8.6"      "HSS114.3x9.5"      "HSS127x3.2"      "HSS127x4.8"
"HSS127x6.4"      "HSS127x6.6"      "HSS127x7.9"      "HSS127x9.5"
"HSS127x12.7"      "HSS139.7x6.6"      "HSS139.7x9.5"
"HSS139.7x12.7"      "HSS141.3x3.4"      "HSS141.3x4.8"
"HSS141.3x6.6"      "HSS141.3x9.5"      "HSS141.3x12.7"      "C1"      "C2"
"C3"      "C4"      "C5"      "C6"      "C91"      "HSS152.4x3.2"      "HSS152.4x4.8"
"HSS152.4x6.4"      "HSS152.4x7.1"      "HSS152.4x7.9"
"HSS152.4x9.5"      "HSS152.4x12.7"      "HSS168.3x3.2"
"HSS168.3x4.8"      "HSS168.3x6.4"      "HSS168.3x7.1"
"HSS168.3x7.9"      "HSS168.3x9.5"      "HSS168.3x11"
"HSS168.3x12.7"      "HSS174.6x4.8"      "HSS174.6x6.4"
"HSS174.6x7.9"      "HSS174.6x9.5"      "HSS174.6x12.7"      "C7"      "C8"
"C9"      "C10"      "C11"      "C12"      "C92"      "HSS177.8x3.2"      "HSS177.8x4.8"
"HSS177.8x6.4"      "HSS177.8x7.9"      "HSS177.8x9.5"
"HSS177.8x12.7"      "HSS190.5x4.8"      "HSS190.5x6.4"
"HSS190.5x7.9"      "HSS190.5x9.5"      "HSS190.5x12.7"
"HSS193.7x8.3"      "HSS193.7x9.5"      "C13"      "C14"      "C15"      "C16"      "C17"
"C18"      "C93"      "HSS219.1x4.8"      "HSS219.1x6.4"
"HSS219.1x8.2"      "HSS219.1x9.5"      "HSS219.1x12.7"
"HSS219.1x15.9"      "C19"      "C20"      "C21"      "C22"      "C23"      "C24"      "C94"
"HSS244.5x4.8"      "HSS244.5x6.4"      "HSS244.5x7.9"
"HSS244.5x9.5"      "HSS244.5x12.7"      "C25"      "C26"      "C27"      "C28"
"C29"      "C30"      "C95"      "HSS254x4.8"      "HSS254x6.4"
"HSS254x7.9"      "HSS254x9.5"      "HSS254x12.7"
"HSS254x15.9"      "HSS273.1x6.4"      "HSS273.1x9.5"
"HSS273.1x12.7"      "C31"      "C32"      "C33"      "C34"      "C35"      "C36"      "C96"
"C37"      "C38"      "C39"      "C40"      "C41"      "C42"      "C97"      "HSS323.9x6.4"
"HSS323.9x9.5"      "HSS323.9x12.7"      "C43"      "C44"      "C45"      "C46"
"C47"      "C48"      "C98"      "C49"      "C50"      "C51"      "C52"      "C53"      "C54"      "C99"
"HSS355.6x6.4"      "HSS355.6x7.9"      "HSS355.6x9.5"
"HSS355.6x12.7"      "HSS355.6x15.9"      "C55"      "C56"      "C57"      "C58"
"C59"      "C60"      "C110"      "C61"      "C62"      "C63"      "C64"      "C65"      "C66"
"C101"      "HSS406.4x6.4"      "HSS406.4x7.9"      "HSS406.4x9.5"
"HSS406.4x11.1"      "HSS406.4x12.7"      "HSS406.4x15.9"      "C67"      "C68"
"C69"      "C70"      "C71"      "C72"      "C112"      "C73"      "C74"      "C75"      "C76"      "C77"
"C78"      "C113"      "HSS457.2x9.5"      "HSS457.2x12.7"      "C79"      "C80"
"C81"      "C82"      "C83"      "C84"      "C114"      "C85"      "C86"      "C87"      "C88"      "C89"
"C90"      "C115"      "HSS508x9.5"      "HSS508x12.7"];
cdiameter=[42.2 48.3      48.3      48.3      60.3      60.3      60.3
60.3      60.3      63.5      63.5      63.5      73      73      73      73      76.2
76.2      76.2      76.2      76.2      76.2      76.2      88.9      88.9
```

```

88.9      88.9      88.9      88.9      88.9      101.6      101.6
101.6      101.6      101.6      101.6      101.6      114.3      114.3
114.3      114.3      114.3      127 127 127 127 127 127 127 139.7
139.7      139.7      141.3      141.3      141.3      141.3      141.3      150
150 150 150 150 150 150 152.4      152.4      152.4      152.4
152.4      152.4      152.4      168.3      168.3      168.3      168.3
168.3      168.3      168.3      168.3      174.6      174.6      174.6
174.6      174.6      175 175 175 175 175 175 175 177.8      177.8
177.8      177.8      177.8      177.8      190.5      190.5      190.5
190.5      190.5      193.7      193.7      200 200 200 200 200 200 200
219.1      219.1      219.1      219.1      219.1      219.1      225 225 225
225 225 225 225 244.5      244.5      244.5      244.5      244.5      250
250 250 250 250 250 250 254 254 254 254 254 254 273.1
273.1      273.1      275 275 275 275 275 275 275 275 300 300 300 300
300 300 300 323.9      323.9      323.9      325 325 325 325 325 325
325 350 350 350 350 350 350 350 355.6      355.6      355.6
355.6      355.6      375 375 375 375 375 375 375 375 400 400 400 400
400 400 400 406.4      406.4      406.4      406.4      406.4      406.4
425 425 425 425 425 425 425 450 450 450 450 450 450 450 450
457.2      457.2      475 475 475 475 475 475 475 475 500 500 500 500
500 500 500 508 508];
cwthick=[3.6      3      3.7 4.8 3.2 3.9 4.8 5.5 6.4 3.2 4.8 6.4
3.2 4.8 5.2 6.4 3.2 3.4 3.9 4.8 5.2 5.5 6.4 3.2 4.8 5.2 5.5
6.4 7.6 9      3.2 4.8 5.6 5.7 6      6.4 8      3.2 4.8 6      8.6 9.5
3.2 4.8 6.4 6.6 7.9 9.5 12.7      6.6 9.5 12.7      3.4 4.8 6.6
9.5 12.7      10 12 14 16 18 20 22 3.2 4.8 6.4 7.1 7.9
9.5 12.7      3.2 4.8 6.4 7.1 7.9 9.5 11 12.7      4.8 6.4 7.9
9.5 12.7      10 12 14 16 18 20 22 3.2 4.8 6.4 7.9 9.5
12.7      4.8 6.4 7.9 9.5 12.7      8.3 9.5 10 12 14 16 18
20 22 4.8 6.4 8.2 9.5 12.7      15.9      10 12 14 16 18
20 22 4.8 6.4 7.9 9.5 12.7      10 12 14 16 18 20 22
4.8 6.4 7.9 9.5 12.7      15.9      6.4 9.5 12.7      10 12 14
16 18 20 22 10 12 14 16 18 20 22 6.4 9.5 12.7
10 12 14 16 18 20 22 10 12 14 16 18 20 22 6.4
7.9 9.5 12.7      15.9      10 12 14 16 18 20 22 10 12
14 16 18 20 22 6.4 7.9 9.5 11.1      12.7      15.9      10
12 14 16 18 20 22 10 12 14 16 18 20 22 9.5 12.7
10 12 14 16 18 20 22 10 12 14 16 18 20 22 9.5
12.7];

```

```

d=cdiameter(i);
t=cwthick(i);
n=cname(i);
end

```

A.3 W Section Function

```
function [a,d,b,f,w,n] = Wsection(i)
%unit is mm

warea=[1630 1730      1910      2280      2290      2470      2480
2680    2850    2860    2860    3040    3040    3060    3220
3390    3590    3590    3630    3790    3970    4180    4190
4190    4570    4740    4910    4940    4950    4960    5320
5670    5700    5710    5890    5890    6260    6450    6650
6650    6650    6840    7230    7420    7550    7550    7610
7610    8130    8390    8450    8580    8580    8710    9100
9100     9100     9290     9420     9480     9480     9480
10100    10100    10200    10500    10500    10500    10800
10800    11000    11000    11400    11400    11500    11700
11800    12300    12300    12600    12700    12800    12900
12900    13000    13500    13600    13900    14100    14400
14500    14600    14600    15000    15500    15700    15900
15900    16300    16500    16800    16900    17000    17100
17600    17800    17900    18200    18400    18700    18800
18900    19000    19200    19400    19500    19800    20100
20100    20500    20600    21000    21200    21700    22100
22200    22400    22600    22700    22800    23200    23500
24400    24700    24700    24900    25000    25000    25000
25700    25700    26800    27100    27500    27700    27900
27900    28100    28300    28600    28800    29000    29900
30100    30300    30700    30800    31500    31800    31900
32300    32300    32800    33200    33400    33400    33900
34400    34600    34600    36100    36200    36300    36500
36600    36800    36800    37900    38100    38300    39200
39900    39900    40000    40100    40200    40200    41000
41200    42100    42900    43400    43700    43700    43900
44000    44200    44300    44600    44700    44800    45900
46800    47400    47400    47600    47800    48600    48800
49000    49100    49700    49800    49800    49900    49900
52600    52800    52800    52800    53100    53500    53600
53700    53700    55100    55200    55500    55600    56300
57400    57700    57900    58200    59000    59100    60500
61500    61900    61900    62500    63000    63400    63500
63800    64000    65200    67100    67700    68400    68400
70300    70300    70300    71000    73500    74200    74800
74800    75500    75500    80600    81900    83900    86500
92300    94800    95500    101000    103000    105000    112000
115000    124000    125000    126000    137000    139000    152000
154000 162000 166000 175000];
wdepth=[148 150 200 251 153 106 203 302 254 206 152 305 127
160 257 207 131 310 259 157 210 312 259 348 201 162 262 310
399 353 205 312 267 351 404 203 247 356 450 318 206 404 358
252 210 302 455 406 348 526 307 409 257 460 216 523 351 254
310 528 457 414 353 307 257 599 528 460 536 417 310 222 462
259 353 602 533 467 307 414 229 264 536 356 602 470 312 538
361 462 607 419 269 315 363 544 678 612 467 318 274 427 749
356 549 683 617 323 472 754 361 282 432 544 688 622 612 475
328 757 363 546 290 693 762 617 836 483 333 368 551 767 701
841 490 622 770 554 373 904 340 846 495 376 627 696 561 780
970 912 348 851 500 381 914 701 635 572 980 859 919 356 772
508 640 386 706 991 922 577 366 780 518 648 394 927 714 983
856 584 653 932 373 399 1000      785 526 991 721 861 594 660
```

```

384 1090      943 927 406 536 1010      795 729 869 932 1000
668 602 391 951 417 546 737 803 1100      937 876 1020      1010
612 678 404 1020      942 744 556 424 1110      884 813 1030
1010      947 414 688 752 434 566 894 1020      823 1040      958
1040      699 1120      427 762 445 904 833 965 1030      772 711
455 1030      914 843 1060      975 1040      465 475 1050      988
483 998 498 1070      1010      826 513 1090      531 1040      1110
549 1060      569 1080      579 1090      599 1090];
width=[100 100 100 101 102 103 102 101 102 102 152 101 127
102 102 133 128 102 102 153 134 102 146 127 165 154 147 165
140 128 166 166 148 171 140 203 202 171 152 167 204 178 172
203 205 203 153 178 203 165 204 179 204 154 206 207 204 254
205 166 191 180 205 254 254 178 209 191 167 181 254 209 192
257 254 179 209 193 305 259 210 257 210 254 228 194 305 211
257 279 228 262 259 307 257 212 254 229 282 307 262 264 264
368 214 254 230 310 282 267 371 262 264 312 254 229 325 284
310 267 371 312 264 257 267 325 292 287 312 373 315 267 254
292 284 328 267 315 373 305 315 292 284 394 328 356 318 267
300 305 318 295 287 396 305 356 330 315 300 292 305 320 381
290 328 399 358 300 307 318 323 381 292 330 399 307 356 401
399 320 330 310 325 401 300 384 295 401 358 401 323 333 328
401 307 419 404 297 302 384 361 404 419 401 335 325 330 310
406 300 363 386 401 422 401 302 401 328 338 333 305 422 366
302 409 401 404 389 305 401 424 335 340 366 411 305 406 404
391 307 422 310 343 404 340 371 417 409 394 424 406 373 348
419 409 411 396 315 427 409 422 424 411 432 427 434 432 417
437 389 437 424 442 447 429 450 452 455 457 472 462 478
472];
webt=[4.32 4.32 4.32 4.83 5.84 7.11 5.84
5.08 5.84 6.22 5.84 5.59 6.10 6.60 6.10
5.84 6.86 5.97 6.35 6.60 6.35 6.60 6.10
5.84 6.22 8.13 6.60 5.84 6.35 6.48 7.24
6.60 7.62 6.86 6.99 7.24 7.37 7.24 7.62
7.62 7.87 7.49 7.87 8.00 9.14 7.49 8.00
7.75 7.75 8.89 8.51 8.76 8.89 9.14 10.2
8.89 8.64 8.64 9.40 9.65 9.02 9.65 9.40
8.76 9.40 10.0 9.53 9.91 10.3 10.9 9.14
13.0 10.5 10.7 9.53 10.9 10.2 11.4 9.91
10.0 14.5 11.9 10.9 10.5 10.5 12.6 10.9
11.6 11.4 10.8 11.2 11.6 13.5 11.9 13.0
13.1 11.7 11.9 12.2 13.1 15.4 13.3 11.9
11.2 14.7 12.4 13.1 14.0 13.6 13.2 12.3
17.3 14.9 12.7 13.1 14.0 12.7 15.0 15.5
13.8 13.3 14.0 19.2 14.5 14.4 14.0 14.0
16.6 18.0 15.0 15.2 14.9 15.5 14.7 17.0
15.4 15.6 16.5 16.4 15.2 20.1 15.4 18.5
17.3 16.5 15.4 18.3 16.5 16.0 15.9 22.1
16.1 20.6 18.9 16.5 16.8 17.9 19.1 16.5
17.0 17.3 24.4 16.6 22.6 19.1 21.1 18.4
16.5 18.4 21.1 26.9 18.0 24.4 20.6 22.6
19.4 19.1 16.5 18.2 23.1 22.1 21.1 30.0
24.9 19.1 19.7 26.9 16.5 21.1 19.7 25.4
24.4 32.8 18.0 22.1 19.3 27.2 29.5 21.1
21.1 23.1 21.1 20.3 19.1 26.4 27.9 35.6
24.4 30.0 32.5 24.9 23.6 19.9 21.3 22.1
24.4 21.1 31.0 29.5 38.9 26.2 22.5 26.9
35.6 32.8 22.0 24.4 25.9 26.9 23.6 24.0
41.4 32.0 29.5 35.8 38.6 26.4 25.4 29.0
30.0 25.9 31.0 35.1 26.2 45.2 32.0 39.1
29.5 31.5 28.4 28.4 35.1 38.6 42.2 29.5

```

32.0	34.5	36.1	31.0	31.0	45.0	47.8	34.0
34.5	51.3	38.1	55.6	39.1	40.9	50.0	60.5
45.5	66.0	50.0	50.0	71.9	55.1	78.0	60.5
95.0	64.0	100	76.7];				
wflanget=[4.95	5.46	5.21	5.33	7.11	8.76	6.48	
5.72	6.86	8.00	6.60	6.73	9.14	10.3	8.38
8.38	10.9	8.89	10.0	9.27	10.2	10.8	9.14
8.51	10.2	11.6	11.2	9.65	8.76	10.7	11.8
11.2	13.0	9.78	11.2	11.0	11.0	11.6	10.8
13.2	12.6	10.9	13.1	13.5	14.2	13.1	13.3
12.8	13.5	11.4	14.6	14.4	15.7	15.4	17.4
10.9	15.1	14.2	16.3	13.6	14.5	16.0	16.8
14.6	15.6	12.8	13.3	16.0	16.5	18.2	16.3
20.6	17.7	17.3	16.4	15.0	15.6	19.1	15.4
16.9	23.7	19.6	17.4	18.3	14.9	20.6	17.0
18.8	19.9	17.3	17.3	19.3	22.1	18.7	21.7
21.2	16.3	19.6	19.6	20.6	25.1	22.2	15.5
18.0	23.6	18.9	22.2	22.9	22.1	17.0	19.8
28.4	25.0	20.3	21.1	24.9	19.1	23.9	25.1
19.3	21.8	22.2	31.8	23.6	21.6	21.6	18.8
26.9	28.2	23.9	24.4	23.6	27.9	21.7	30.5
24.4	25.4	26.4	26.2	20.1	31.8	24.4	33.5
27.7	27.7	24.8	29.2	30.0	21.1	23.9	35.6
26.9	36.6	30.2	25.9	27.4	31.0	34.5	26.2
31.0	27.9	39.6	27.2	40.4	34.0	33.3	30.2
30.5	30.0	37.6	44.2	30.2	44.5	37.1	36.6
32.0	34.0	27.2	29.2	41.4	39.9	34.5	48.3
39.6	36.1	33.5	48.5	31.0	38.1	32.5	45.5
43.9	52.6	31.0	39.9	32.0	43.7	53.6	40.1
38.1	40.9	35.6	34.3	36.1	48.0	50.5	57.2
43.9	48.0	58.4	45.0	41.9	36.1	36.6	39.9
43.9	40.1	55.6	53.1	62.7	46.0	39.9	49.0
63.5	52.6	40.1	43.9	47.0	49.0	41.9	42.7
68.8	57.9	53.1	57.4	69.6	48.0	46.0	52.1
54.1	47.0	54.1	63.0	45.0	75.2	57.9	62.7
53.1	56.9	51.1	51.1	63.0	69.1	67.6	52.1
57.9	62.0	64.0	55.9	55.9	72.4	77.2	59.9
62.0	81.5	68.1	88.9	70.1	73.9	89.9	97.0
82.0	106	89.9	89.9	115	99.1	125	109
115];						130	115
115];						140	


```

wname=["W150X13"      "W150X13.5"  "W200X15"      "W250X17.9"
"W150X18"      "W100X19.3"  "W200X19.3"  "W310X21"      "W250X22.3"
"W200X22.5"    "W150X22.5"  "W310X23.8"  "W130X23.8"    "W150X24"
"W250X25.3"    "W200X26.6"  "W130X28.1"  "W310X28.3"    "W250X28.4"
"W150X29.8"    "W200X31.3"  "W310X32.7"  "W250X32.7"    "W360X32.9"
"W200X35.9"    "W150X37.1"  "W250X38.5"  "W310X38.7"    "W410X38.8"
"W360X39"      "W200X41.7"  "W310X44.5"  "W250X44.8"    "W360X44"
"W410X46.1"    "W200X46.1"  "W250X49.1"  "W360X51"      "W460X52"
"W310X52"      "W200X52"    "W410X53"    "W360X57.8"    "W250X58"
"W200X59"      "W310X60"    "W460X60"    "W410X60"      "W360X64"
"W530X66"      "W310X67"    "W410X67"    "W250X67"      "W460X68"
"W200X71"      "W530X72"    "W360X72"    "W250X73"      "W310X74"
"W530X74"      "W460X74"    "W410X75"    "W360X79"      "W310X79"
"W250X80"      "W610X82"    "W530X82"    "W460X82"      "W530X85"
"W410X85"      "W310X86"    "W200X86"    "W460X89"      "W250X89"
"W360X91"      "W610X92"    "W530X92"    "W460X97"      "W310X97"
"W410X100"     "W200X100"    "W250X101"   "W530X101"     "W360X101"
"W610X101"     "W460X106"    "W310X107"   "W530X109"     "W360X110"
"W460X113"     "W610X113"    "W410X114"   "W250X115"     "W310X117"
"W360X122"     "W530X123"    "W690X125"   "W610X125"     "W460X128"

```

```

"W310X129" "W250X131" "W410X132" "W760X134" "W360X134"
"W530X138" "W690X140" "W610X140" "W310X143" "W460X144"
"W760X147" "W360X147" "W250X149" "W410X149" "W530X150"
"W690X152" "W610X153" "W610X155" "W460X158" "W310X158"
"W760X161" "W360X162" "W530X165" "W250X167" "W690X170"
"W760X173" "W610X174" "W840X176" "W460X177" "W310X179"
"W360X179" "W530X182" "W760X185" "W690X192" "W840X193"
"W460X193" "W610X195" "W760X196" "W530X196" "W360X196"
"W920X201" "W310X202" "W840X210" "W460X213" "W360X216"
"W610X217" "W690X217" "W530X219" "W760X220" "W1000X222"
"W920X223" "W310X226" "W840X226" "W460X235" "W360X237"
"W920X238" "W690X240" "W610X241" "W530X248" "W1000X249"
"W840X251" "W920X253" "W310X253" "W760X257" "W460X260"
"W610X262" "W360X262" "W690X265" "W1000X272" "W920X271"
"W530X272" "W310X283" "W760X284" "W460X286" "W610X285"
"W360X287" "W920X289" "W690X289" "W1000X296" "W840X299"
"W530X300" "W610X307" "W920X313" "W310X313" "W360X314"
"W1000X314" "W760X314" "W460X315" "W1000X321" "W690X323"
"W840X329" "W530X332" "W610X341" "W310X342" "W1100X343"
"W920X345" "W920X344" "W360X347" "W460X349" "W1000X350"
"W760X350" "W690X350" "W840X359" "W920X368" "W1000X371"
"W610X372" "W530X369" "W310X375" "W920X381" "W360X382"
"W460X384" "W690X384" "W760X389" "W1100X390" "W920X390"
"W840X392" "W1000X393" "W1000X412" "W530X409" "W610X415"
"W310X415" "W1000X415" "W920X420" "W690X419" "W460X421"
"W360X421" "W1100X433" "W840X433" "W760X434" "W1000X438"
"W1000X443" "W920X449" "W310X454" "W610X455" "W690X457"
"W360X463" "W460X464" "W840X473" "W1000X483" "W760X484"
"W1000X486" "W920X491" "W1000X494" "W610X498" "W1100X499"
"W310X500" "W690X500" "W360X509" "W840X527" "W760X531"
"W920X537" "W1000X539" "W690X548" "W610X551" "W360X551"
"W1000X554" "W840X576" "W760X582" "W1000X584" "W920X588"
"W1000X591" "W360X592" "W360X634" "W1000X642" "W920X656"
"W360X677" "W920X725" "W360X744" "W1000X748" "W920X787"
"W690X802" "W360X818" "W1000X883" "W360X900" "W920X970"
"W1000X976" "W360X990" "W920X1077" "W360X1086" "W920X1194"
"W360X1202" "W920X1269" "W360X1299" "W920X1377"]];

```

```

a=warea(i);
d=wdepth(i);
b=width(i);
f=wflanget(i);
w=webt(i);
n=wname(i);

```

```
end
```

Appendix B

Algorithms and

API MATLAB

Code

Appendix B Algorithms and API MATLAB Code

B.1 ABC Algorithm

```
%% clean-up the workspace & command window

clear;
clc;

%%set the following flag to true to attach to an existing
instance of the program otherwise a new instance of the
program will be started
AttachToInstance = false(); % true(); %

%% set the following flag to true to manually specify the
path to ETABS.exe
%% this allows for a connection to a version of ETABS other
than the latest installation
%% otherwise the latest installed version of ETABS will be
launched
SpecifyPath = false(); % true(); %

%% if the above flag is set to true, specify the path to
ETABS below
ProgramPath      = 'C:\Program Files\Computers and
Structures\ETABS 20\ETABS.exe';

%% full path to API dll
%% set it to the installation folder
APIDLPath        = 'C:\Program Files\Computers and
Structures\ETABS 20\ETABSv1.dll';

%% full path to the model
%% set it to the desired path of your model
ModelDirectory = 'C:\CSI_ETABS_API_Example';
if ~exist(ModelDirectory, 'dir')
    mkdir(ModelDirectory);
end
ModelName = 'API_1-001.edb';
ModelPath = strcat(ModelDirectory, filesep, ModelName);

%% create API helper object
a = NET.addAssembly(APIDLPath);
helper = ETABSv1.Helper;
helper = NET.explicitCast(helper, 'ETABSv1.cHelper');

if AttachToInstance
    %% attach to a running instance of ETABS
    ETABSObject =
helper.GetObject('CSI.ETABS.API.ETABSObject');
    ETABSObject =
NET.explicitCast(ETABSObject, 'ETABSv1.cOAPI');
```

```

else
    if SpecifyPath
        %% create an instance of the ETABS object
        from the specified path
        ETABSObject
        helper.CreateObject(ProgramPath);
    else
        %% create an instance of the ETABS object
        from the latest installed ETABS
        ETABSObject
        helper.CreateObjectProgID('CSI.ETABS.API.ETABSObject');
    end
    ETABSObject
    NET.explicitCast(ETABSObject, 'ETABSV1.COAPI');

    %% start ETABS application
    ETABSObject.ApplicationStart;
end
helper = 0;

%% create SapModel object
SapModel
NET.explicitCast(ETABSObject.SapModel, 'ETABSV1.cSapModel');

%% initialize model
ret = SapModel.InitializeNewModel;

%% Open model
File = NET.explicitCast(SapModel.File, 'ETABSV1.cFile');
ret = File.OpenFile('C:\Users\Master 2\Desktop\ETABS
MODEL\EX1.edb');
%% Problem Definition
nVar = 9; % D Number of Unknown Variables
VarSize = [1 nVar]; % Unknown Variables Matrix Size
(NOT Change)

VarMin = [1 1 1 1 1 1 1 1 1]; % VarMin=[IC1
OC1 IC2 OC2 IC3 OC3 IB OB CS]
VarMax = [233 233 233 233 233 233 283 283 17]; % VarMax=[IC1
OC1 IC2 OC2 IC3 OC3 IB OB CS]
%% ABC Settings

MaxIt=200; % Maximum Number of Iterations

nPop=30; % Population Size (Colony Size)

nOnlooker=nPop; % Number of Onlooker Bees

L=round(0.5*nVar*nPop); % Abandonment Limit Parameter (Trial
Limit)

aa=1; % Acceleration Coefficient Upper
Bound

%% Initialization

```

```

% Empty Bee Structure
empty_bee.Var=[];
empty_bee.Weight=[];

% Initialize Population Array
pop= repmat(empty_bee,nPop,1);

% Initialize Best Solution Ever Found
BestSol.Weight=inf;

% Create Initial Population
for i=1:nPop
    pop(i).Var = ceil(unifrnd(VarMin, VarMax));
    [pop(i).Weight] = WeightFunction(pop(i).Var,SapModel);
    if pop(i).Weight < BestSol.Weight
        BestSol = pop(i);
    end
end

% Abandonment Counter
Ac=zeros(nPop,1);

% Array to Hold Best Cost Values
BestWeight=zeros(MaxIt,1);

%% ABC Main Loop

for it=1:MaxIt

    % Recruited Bees
    for i=1:nPop

        % Choose k randomly, not equal to i
        K=[1:i-1 i+1:nPop];
        k=K(randi([1 numel(K)]));

        % Define Acceleration Coeff.
        phi=aa*unifrnd(-1,+1,VarSize);

        % New Bee Position
        newbee.Var=ceil(pop(i).Var+phi.*(pop(i).Var-
pop(k).Var));

        % Clipping
        newbee.Var = max(newbee.Var, VarMin);
        newbee.Var = min(newbee.Var, VarMax);

        % Evaluation
        [newbee.Weight]
WeightFunction(newbee.Var,SapModel);

        % Comparision

```

```

        if newbee.Weight<=pop(i).Weight
            pop(i)=newbee;
        else
            Ac(i)=Ac(i)+1;
        end

    end

    % Calculate Fitness Values and Selection Probabilities
    F=zeros(nPop,1);
    MeanWeight = mean([pop.Weight]);
    for i=1:nPop
        F(i) = exp(-pop(i).Weight/MeanWeight); % Convert
Cost to Fitness
    end
    P=F/sum(F);

    % Onlooker Bees
    for m=1:nOnlooker

        % Select Source Site
        i=RouletteWheelSelection(P);

        % Choose k randomly, not equal to i
        K=[1:i-1 i+1:nPop];
        k=K(randi([1 numel(K)]));

        % Define Acceleration Coeff.
        phi=aa*unifrnd(-1,+1,VarSize);

        % New Bee Position
        newbee.Var=ceil(pop(i).Var+phi.*(pop(i).Var-
pop(k).Var));

        % Clipping
        newbee.Var = max(newbee.Var, VarMin);
        newbee.Var = min(newbee.Var, VarMax);

        % Evaluation
        [newbee.Weight] =
WeightFunction(newbee.Var,SapModel);

        % Comparision
        if newbee.Weight<=pop(i).Weight
            pop(i)=newbee;
        else
            Ac(i)=Ac(i)+1;
        end

    end

    % Scout Bees
    for i=1:nPop
        if Ac(i)>=L
            pop(i).Var = ceil(unifrnd(VarMin, VarMax));

```

```

        [pop(i).Weight]
WeightFunction(pop(i).Var,SapModel);
        Ac(i)=0;
    end
end

% Update Best Solution Ever Found
for i=1:nPop
    if pop(i).Weight<=BestSol.Weight
        BestSol=pop(i);
    end
end

% Store Best Cost Ever Found
BestWeight(it)=BestSol.Weight;

[WEIGHT,C,sc,dc,gc]
WeightFunction(BestSol.Var,SapModel);
% Display Iteration Information
disp(['Iteration ' num2str(it) ': Best Weight = '
num2str(BestWeight(it)) ': C = ' num2str(C) ', SC = '
num2str(sc) ', DC = ' num2str(dc) ', GC = ' num2str(gc)']);

end

%% Results & Display

figure;
%plot(BestWeights, 'LineWidth', 2);
semilogy(BestWeight, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best Weight');
grid on;
Csec=['C1" "C2" "C3" "C4" "C5" "C6" "C7" "C8" "C9" "C10"
"C11" "C12" "C13" "C14" "C15" "C16" "C17" "C18" "C19" "C20"
"C21" "C22" "C23" "C24" "C25" "C26" "C27" "C28" "C29" "C30"
"C31" "C32" "C33" "C34" "C35" "C36" "C37" "C38" "C39" "C40"
"C41" "C42" "C43" "C44" "C45" "C46" "C47" "C48" "C49" "C50"
"C51" "C52" "C53" "C54" "C55" "C56" "C57" "C58" "C59" "C60"
"C61" "C62" "C63" "C64" "C65" "C66" "C67" "C68" "C69" "C70"
"C71" "C72" "C73" "C74" "C75" "C76" "C77" "C78" "C79" "C80"
"C81" "C82" "C83" "C84" "C85" "C86" "C87" "C88" "C89" "C90"
"C91" "C92" "C93" "C94" "C95" "C96" "C97" "C98" "C99"
"C100"];
fname=['IC1" "OC1" "IC2" "OC2" "IC3" "OC3" "IB" "OB"];
BestSol

for j=1:9
    if j<7
        [d,th,n] = Csection(BestSol.Var(j));
        ret = FrameObj.SetSection(fname(j), n,
ETABSv1.eItemType.Group);
        ar(j)=pi()*d^2/4-pi()*(d-th)^2/4;
        ar(j+4)=pi()*(d-th)^2/4;
        disp([num2str(fname(j)) ' Section is ' num2str(n)
'=(' num2str(d) ', ' num2str(th) ', ' num2str(ar(j)) ', '
num2str(ar(j+8)) ' '])

```

```

        disp([' '])
    elseif j<5
        [a,d,b,f,w,n] = Wsection(BestSol.Var(j));
        disp([num2str(frname(j)) ' Section is ' num2str(n)
'=(' num2str(d) ' , ' num2str(b) ' , ' num2str(f) ' , '
num2str(w) ' , ' num2str(a) ')]])
        disp([' '])
    else
        disp(['Concrete Stringth is'
num2str((BestSol.Var(j)-1)*5+20) 'MPa'])
        disp([' '])
    end

end

%% close Sap2000
[Weight,C,sc,dc,gc] = WeightFunction(BestSol.Var,SapModel);
disp([': C = ' num2str(C) ', SC = ' num2str(sc) ', DC = '
num2str(dc) ', GC = ' num2str(gc)])
ret=File.Save('C:\Users\Master 2\Desktop\ETABS
MODEL\EX1.edb');
ret = ETABSObject.ApplicationExit(false());
File = 0;
PropMaterial = 0;
PropFrame = 0;
FrameObj = 0;
PointObj = 0;
View = 0;
LoadPatterns = 0;
Analyze = 0;
AnalysisResults = 0;
AnalysisResultsSetup = 0;
SapModel = 0;
ETABSObject = 0;

```

B.1.1 Roulette Wheel Selection Function

```

function i = RouletteWheelSelection(P)

    r = rand;

    C = cumsum(P);

    i = find(r <= C, 1, 'first');

end

```

B.2 TLBO Algorithm

```
% clean-up the workspace & command window

clear;
clc;

%%set the following flag to true to attach to an existing
instance of the program otherwise a new instance of the
program will be started
AttachToInstance = false(); % true(); %

%% set the following flag to true to manually specify the
path to ETABS.exe
%% this allows for a connection to a version of ETABS other
than the latest installation
%% otherwise the latest installed version of ETABS will be
launched
SpecifyPath = false(); % true(); %

%% if the above flag is set to true, specify the path to
ETABS below
ProgramPath      = 'C:\Program Files\Computers and
Structures\ETABS 20\ETABS.exe';

%% full path to API dll
%% set it to the installation folder
APIDLLPath      = 'C:\Program Files\Computers and
Structures\ETABS 20\ETABSV1.dll';

%% full path to the model
%% set it to the desired path of your model
ModelDirectory = 'C:\CSI_ETABS_API_Example';
if ~exist(ModelDirectory, 'dir')
    mkdir(ModelDirectory);
end
ModelName = 'API_1-001.edb';
ModelPath = strcat(ModelDirectory, filesep, ModelName);

%% create API helper object
a = NET.addAssembly(APIDLLPath);
helper = ETABSV1.Helper;
helper = NET.explicitCast(helper, 'ETABSV1.cHelper');

if AttachToInstance
    %% attach to a running instance of ETABS
    ETABSObject =
helper.GetObject('CSI.ETABS.API.ETABSObject');
    ETABSObject =
NET.explicitCast(ETABSObject, 'ETABSV1.COAPI');
else
    if SpecifyPath
        %% create an instance of the ETABS object
        from the specified path
    end
end
```

```

                                ETABSObject                                =
helper.CreateObject(ProgramPath);
    else
        %% create an instance of the ETABS object
        from the latest installed ETABS
                                ETABSObject                                =
helper.CreateObjectProgID('CSI.ETABS.API.ETABSObject');
    end
                                ETABSObject                                =
NET.explicitCast(ETABSObject, 'ETABSV1.COAPI');

    %% start ETABS application
    ETABSObject.ApplicationStart;
end
helper = 0;

%% create SapModel object
SapModel
NET.explicitCast(ETABSObject.SapModel, 'ETABSV1.cSapModel');

%% initialize model
ret = SapModel.InitializeNewModel;

%% Open model
File = NET.explicitCast(SapModel.File, 'ETABSV1.cFile');
ret = File.OpenFile('C:\Users\User\Desktop\ETABS
MODEL\EX1.edb');
%% Problem Definition
nVar = 9;    % D Number of Unknown Variables
VarSize = [1 nVar];    % Unknown Variables Matrix Size
(NOT Change)

VarMin =[1 1 1 1 1 1 1 1 1];    % VarMin=[IC1
OC1 IC2 OC2 IC3 OC3 IB OB CS]
VarMax =[233 233 233 233 233 233 283 283 17];    % VarMax=[IC1
OC1 IC2 OC2 IC3 OC3 IB OB CS]
%% DE Parameters

MaxIt=200;    % Maximum Number of Iterations

nPop=30;    % Population Size

rng(1);
%% Initialization

% Empty Structure for Individuals
empty_individual.Var = [];
empty_individual.Weight = [];

% Initialize Population Array
pop = repmat(empty_individual, nPop, 1);

% Initialize Best Solution
BestSol.Weight = inf;

```

```

% Initialize Population Members
for i=1:nPop

    pop(i).Var = ceil(unifrnd(VarMin, VarMax));
    [pop(i).Weight] = WeightFunction(pop(i).Var, SapModel);

    if pop(i).Weight < BestSol.Weight
        BestSol = pop(i);
    end
end

% Initialize Best Cost Record
BestWeight = zeros(MaxIt,1);

%% TLBO Main Loop

for it=1:MaxIt

    % Calculate Population Mean
    Mean = 0;
    for i=1:nPop
        Mean = Mean + pop(i).Var;
    end
    Mean = Mean/nPop;

    % Select Teacher
    Teacher = pop(1);
    for i=2:nPop
        if pop(i).Weight < Teacher.Weight
            Teacher = pop(i);
        end
    end

    % Teacher Phase
    for i=1:nPop
        % Create Empty Solution
        newsol = empty_individual;

        % Teaching Factor
        TF = randi([1 2]);

        % Teaching (moving towards teacher)
        newsol.Var = ceil(
            rand(VarSize).*(Teacher.Var - TF*Mean)) +
            pop(i).Var;

        % Clipping
        newsol.Var = max(newsol.Var, VarMin);
        newsol.Var = min(newsol.Var, VarMax);

        % Evaluation
        newsol.Weight = WeightFunction(newsol.Var, SapModel);

        % Comparision
        if newsol.Weight < pop(i).Weight
            pop(i) = newsol;
        end
    end
end

```

```

        if pop(i).Weight < BestSol.Weight
            BestSol = pop(i);
        end
    end
end

% Learner Phase
for i=1:nPop

    A = 1:nPop;
    A(i)=[];
    j = A(randi(nPop-1));

    Step = pop(i).Var - pop(j).Var;
    if pop(j).Weight < pop(i).Weight
        Step = -Step;
    end

    % Create Empty Solution
    newsol = empty_individual;

    % Teaching (moving towards teacher)
    newsol.Var = ceil( (pop(i).Var +
rand(VarSize).*Step);

    % Clipping
    newsol.Var = max(newsol.Var, VarMin);
    newsol.Var = min(newsol.Var, VarMax);

    % Evaluation
    newsol.Weight = WeightFunction(newsol.Var,SapModel);

    % Comparision
    if newsol.Weight < pop(i).Weight
        pop(i) = newsol;
        if pop(i).Weight < BestSol.Weight
            BestSol = pop(i);
        end
    end
end

% Store Record for Current Iteration
BestWeight(it) = BestSol.Weight;
[WEIGHT,C,sc,dc,gc] =
WeightFunction(BestSol.Var,SapModel);
% Show Iteration Information
disp(['Iteration ' num2str(it) ': Best Weight = '
num2str(BestWeight(it)) ': C = ' num2str(C) ', SC = '
num2str(sc) ', DC = ' num2str(dc) ', GC = ' num2str(gc)']);

end

%% Results & Display

figure;

```

```

%plot(BestWeights, 'LineWidth', 2);
semilogy(BestWeight, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best Weight');
grid on;
frname=["IC1" "OC1" "IC2" "OC2" "IC3" "OC3" "IB" "OB"];
BestSol

for j=1:9
    if j<7
        [d(j),th(j),n] = Csection(BestSol.Var(j));
        ret = FrameObj.SetSection(frname(j), n,
ETABSv1.eItemType.Group);
        ar(j)=pi()*d(j)^2/4-pi()*(d(j)-th(j))^2/4;
        ar(j+8)=pi()*(d(j)-th(j))^2/4;
        disp([num2str(frname(j)) ' Section is ' num2str(n)
'=(' num2str(d(j)) ', ' num2str(th(j)) ', ' num2str(ar(j))
', ' num2str(ar(j+8)) ')'])
        disp([' '])
    elseif j<5
        [a,d,b,f,w,n] = Wsection(BestSol.Var(j));
        disp([num2str(frname(i)) ' Section is ' num2str(n)
'=(' num2str(d) ', ' num2str(b) ', ' num2str(f) ', '
num2str(w) ', ' num2str(a) ')'])
        disp([' '])
    else
        disp(['Concrete Stringth is'
num2str((BestSol.Var(j)-1)*5+20) 'MPa'])
        disp([' '])
    end
end

end

%% close Sap2000
[Weight,C,sc,dc,gc] = WeightFunction(BestSol.Var,SapModel);
disp([': C = ' num2str(C) ', SC = ' num2str(sc) ', DC = '
num2str(dc) ', GC = ' num2str(gc)])
ret=File.Save('C:\Users\User\Desktop\ETABS MODEL\EX1.edb');
ret = ETABSObject.ApplicationExit(false());
File = 0;
PropMaterial = 0;
PropFrame = 0;
FrameObj = 0;
PointObj = 0;
View = 0;
LoadPatterns = 0;
Analyze = 0;
AnalysisResults = 0;
AnalysisResultsSetup = 0;
SapModel = 0;
ETABSObject = 0;

```

B.3 DE Algorithm

```
%% clean-up the workspace & command window

clear;
clc;

%%set the following flag to true to attach to an existing
instance of the program otherwise a new instance of the
program will be started
AttachToInstance = false(); % true(); %

%% set the following flag to true to manually specify the
path to ETABS.exe
%% this allows for a connection to a version of ETABS other
than the latest installation
%% otherwise the latest installed version of ETABS will be
launched
SpecifyPath = false(); % true(); %

%% if the above flag is set to true, specify the path to
ETABS below
ProgramPath      = 'C:\Program Files\Computers and
Structures\ETABS 20\ETABS.exe';

%% full path to API dll
%% set it to the installation folder
APIDLLPath      = 'C:\Program Files\Computers and
Structures\ETABS 20\ETABSV1.dll';

%% full path to the model
%% set it to the desired path of your model
ModelDirectory = 'C:\CSI_ETABS_API_Example';
if ~exist(ModelDirectory, 'dir')
    mkdir(ModelDirectory);
end
ModelName = 'API_1-001.edb';
ModelPath = strcat(ModelDirectory, filesep, ModelName);

%% create API helper object
a = NET.addAssembly(APIDLLPath);
helper = ETABSV1.Helper;
helper = NET.explicitCast(helper, 'ETABSV1.cHelper');

if AttachToInstance
    %% attach to a running instance of ETABS
    ETABSObject =
helper.GetObject('CSI.ETABS.API.ETABSObject');
    ETABSObject =
NET.explicitCast(ETABSObject, 'ETABSV1.COAPI');
else
    if SpecifyPath
        %% create an instance of the ETABS object
        from the specified path
        ETABSObject =
helper.CreateObject(ProgramPath);
```

```

else
    %% create an instance of the ETABS object
    from the latest installed ETABS
    ETABSObject
    helper.CreateObjectProgID('CSI.ETABS.API.ETABSObject');
end
ETABSObject
NET.explicitCast(ETABSObject, 'ETABSV1.COAPI');

    %% start ETABS application
    ETABSObject.ApplicationStart;
end
helper = 0;

%% create SapModel object
SapModel
NET.explicitCast(ETABSObject.SapModel, 'ETABSV1.cSapModel');

%% initialize model
ret = SapModel.InitializeNewModel;

%% Open model
File = NET.explicitCast(SapModel.File, 'ETABSV1.cFile');
ret = File.OpenFile('C:\Users\Master DE\Desktop\ETABS
MODEL\EX1.edb');
%% Problem Definition
nVar = 9;    % D Number of Unknown Variables
VarSize = [1 nVar];    % Unknown Variables Matrix Size
(NOT Change)

VarMin =[1 1 1 1 1 1 1 1 1];    % VarMin=[IC1
OC1 IC2 OC2 IC3 OC3 IB OB CS]
VarMax =[233 233 233 233 233 233 283 283 17];    % VarMax=[IC1
OC1 IC2 OC2 IC3 OC3 IB OB CS]
%% DE Parameters

MaxIt=200;    % Maximum Number of Iterations

nPop=30;    % Population Size

beta_min=0.2;    % Lower Bound of Scaling Factor
beta_max=0.8;    % Upper Bound of Scaling Factor

pCR=0.2;    % Crossover Probability

rng(1);
%% Initialization

empty_individual.Var=[];
empty_individual.Weight=[];

BestSol.Weight=inf;

pop= repmat(empty_individual, nPop, 1);

```

```

for i=1:nPop

    pop(i).Var = ceil(unifrnd(VarMin, VarMax));
    [pop(i).Weight] = WeightFunction(pop(i).Var, SapModel);

    if pop(i).Weight < BestSol.Weight
        BestSol = pop(i);
    end
end

BestWeight=zeros(MaxIt,1);

%% DE Main Loop

for it=1:MaxIt

    for i=1:nPop

        x=pop(i).Var;

        A=randperm(nPop);

        A(A==i)=[];

        a=A(1);
        b=A(2);
        c=A(3);

        % Mutation
        %beta=unifrnd(beta_min,beta_max);
        beta=unifrnd(beta_min,beta_max,VarSize);
        y=ceil(pop(a).Var+beta.*(pop(b).Var-pop(c).Var));
        y = max(y, VarMin);
        y = min(y, VarMax);

        % Crossover
        z=zeros(size(x));
        j0=randi([1 numel(x)]);
        for j=1:numel(x)
            if j==j0 || rand<=pCR
                z(j)=y(j);
            else
                z(j)=x(j);
            end
        end

        NewSol.Var=z;
        NewSol.Weight=WeightFunction(NewSol.Var, SapModel);

        if NewSol.Weight < pop(i).Weight
            pop(i)=NewSol;

            if pop(i).Weight < BestSol.Weight

```

```

        BestSol=pop(i);
    end
end

end

% Update Best Cost
BestWeight(it)=BestSol.Weight;

[WEIGHT,C,sc,dc,gc] =
WeightFunction(BestSol.Var,SapModel);
% Show Iteration Information
disp(['Iteration ' num2str(it) ': Best Weight = '
num2str(BestWeight(it)) ': C = ' num2str(C) ', SC = '
num2str(sc) ', DC = ' num2str(dc) ', GC = ' num2str(gc)']);

end

%% Results & Display

figure;
%plot(BestWeights, 'LineWidth', 2);
semilogy(BestWeight, 'LineWidth', 2);
xlabel('Iteration');
ylabel('Best Weight');
grid on;
fname=['IC1" "OC1" "IC2" "OC2" "IC3" "OC3" "IB" "OB"];
BestSol

for j=1:9
    if j<7
        [d(j),th(j),n] = Csection(BestSol.Var(j));
        ret = FrameObj.SetSection(fname(j), n,
ETABSv1.eItemType.Group);
        ar(j)=pi()*d(j)^2/4-pi()*(d(j)-th(j))^2/4;
        ar(j+8)=pi()*(d(j)-th(j))^2/4;
        disp([num2str(fname(j)) ' Section is ' num2str(n)
'=(' num2str(d(j)) ', ' num2str(th(j)) ', ' num2str(ar(j))
', ' num2str(ar(j+8)) ')'])
        disp([' '])
    elseif j<5
        [a,d,b,f,w,n] = Wsection(BestSol.Var(j));
        disp([num2str(fname(j)) ' Section is ' num2str(n)
'=(' num2str(d) ', ' num2str(b) ', ' num2str(f) ', '
num2str(w) ', ' num2str(a) ')'])
        disp([' '])
    else
        disp(['Concrete Stringth is'
num2str((BestSol.Var(j)-1)*5+20) 'MPa'])
        disp([' '])
    end
end

end

%% close Sap2000

```

```

[Weight,C,sc,dc,gc] = WeightFunction(BestSol.Var,SapModel);
disp([': C = ' num2str(C) ', SC = ' num2str(sc) ', DC = '
num2str(dc) ', GC = ' num2str(gc)'])
ret=File.Save('C:\Users\Master          DE\Desktop\ETABS
MODEL\EX1.edb');
ret = ETABSObject.ApplicationExit(false());
File = 0;
PropMaterial = 0;
PropFrame = 0;
FrameObj = 0;
PointObj = 0;
View = 0;
LoadPatterns = 0;
Analyze = 0;
AnalysisResults = 0;
AnalysisResultsSetup = 0;
SapModel = 0;
ETABSObject = 0;

```

الملخص

الهدف من التحسين هو تحقيق التصميم الأكثر ملاءمة مع الأخذ في الاعتبار مجموعة من المعايير أو القيود ذات الأولوية. في عالم التقدم الهندسي، من المهم للمهندسين أن يستكشفوا ما هو أبعد من الأساليب التقليدية. في حين أن الأساليب التقليدية قد توفر الراحة والألفة، يجب على المصممين إدراك الأهمية المتزايدة لأساليب التحسين. تلعب هذه الأساليب دورًا حاسمًا في توفير الوقت والمال مع تقليل الأخطاء في نفس الوقت. يمثل هذا البحث خطوة متواضعة نحو فهم المزايا التي يمكن أن يجلبها التحسين إلى التصميم الهيكلي. ويهدف إلى المساهمة في أحد الأهداف الأساسية لكل مهندس: تصميم هياكل ليست آمنة فحسب، بل أيضًا مستدامة اقتصاديًا وبيئيًا.

ولتحقيق ذلك، تم استخدام خوارزميات تحسين التصميم مثل ABC و TLBO و DE لتحديد أبعاد المقطع العرضي الأمثل للأعمدة وفقًا لأحكام تصميم AISC-LRFD. تم استخدام الأداة البرمجية ETABS للحصول على استجابة الإطار للأحمال الخارجية، وتم تنفيذ تكرارات التحسين باستخدام MATLAB. يؤدي تغيير خصائص المقطع العرضي للأعضاء في الإطار الفولاذي ثلاثي الأبعاد إلى تغييرات في أدائها تحت الأحمال الخارجية، مما يستلزم إجراء تحليل جديد. ويتم تسهيل ذلك من خلال استخدام ميزة واجهة برمجة التطبيقات (API) الخاصة بـ ETABS، وإنشاء رابط بين برنامج MATLAB المشفر و ETABS لتبادل البيانات ثنائي الاتجاه. تستمر التكرارات حتى يتم استيفاء معايير التقارب.

وقد وجد أن خوارزميات التصميم الأمثل المستخدمة كانت قادرة على تحسين الأوزان وتقليلها إلى الحد الأدنى من الأقسام الممكنة. وقد وجد أيضًا أن TLBO و DE المقترحين فعالان وأفضل من ABC القياسي من حيث تاريخ التقارب وتقديم أفضل النتائج.

جامعة البحرين

قسم الهندسة المدنية

كلية الهندسة



التصميم الأمثل للمنشآت الفولاذية مع الأعمدة المملوءة بالخرسانة باستخدام خوارزميات الأدلة العليا

أطروحة مقدمة كجزء من متطلبات الحصول على درجة

الماجستير في الهندسة المدنية

إعداد

زيد بشار مصطفى ملكاوي

الرقم الجامعي: 20121435

إشراف

د. محمد أجمل

جامعة البحرين

مملكة البحرين

فبراير، 2024