AMERICAN UNIVERSITY OF BEIRUT

A COMPUTATIONALLY EFFICIENT FRAMEWORK FOR OPTIMIZING THE DESIGNS OF REINFORCED CONCRETE SPECIAL MOMENT RESISTING FRAMES

by
HADI AL KHANSA

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Civil and Environmental Engineering
of the Maroun Semaan Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
September 2020

AMERICAN UNIVERSITY OF BEIRUT

A COMPUTATIONALLY EFFICIENT FRAMEWORK FOR OPTIMIZING THE DESIGNS OF REINFORCED CONCRETE SPECIAL MOMENT RESISTING FRAMES

by
HADI AL KHANSA

Approved by:

	[illegible]
Dr. George Saad, Associate Professor Civil and Environmental Engineering	Advisor
	[illegible]
Dr. Elie Hantouche, Associate Professor Civil and Environmental Engineering	Member of Committee
	[illegible]
[Dr. Mayssa Dabaghi, Assistant Professor] Civil and Environmental Engineering	Member of Committee

Date of thesis/dissertation defense: [September 3, 2020]

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: AL KHANSA HADI MOHAMMAD

Last	First	Middle
AL KHANSA	HADI	MOHAMMAD

[✓ ● Master's Thesis] ○ Master's Project ○ Doctoral Dissertation

[✓ X] I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☐ I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes
after:

One —— year from the date of submission of my thesis, dissertation, or project.

Two —— years from the date of submission of my thesis, dissertation, or project.

Three —— years from the date of submission of my thesis, dissertation, or project.

HK September 10th 2020

Signature Date

ACKNOWLEDGMENTS

Special thanks for Dr. George Saad for his knowledge, continuous support, guidance, patience, and thought provoking questions throughout this research. Thank you for Dr. Elie Hantouche and Dr. Mayssa Dabaghi for being part of my thesis committee.

AN ABSTRACT OF THE THESIS OF

Hadi Al Khansa for Master of Engineering
Major: Civil Engineering

Title: A Computationally Efficient Framework for Optimizing the Designs of Reinforced Concrete Special Moment Resisting Frames

Structural design of reinforced concrete special moment resisting frames requires analyzing multiple design alternatives with the goal of obtaining the most economic, safe, and constructible solution. The purpose of this research is to reduce the computational burden of structural analysis and to automate the iterative design process of reinforced concrete special moment resisting frames.

To reduce the computational burden of structural analysis Artificial Neural Networks (ANNs) are first investigated, and a novel algorithm known as the Weight Matrix method is then explored. Artificial neural networks are computationally efficient mathematical models that can be trained to simulate computationally costly processes. For this research, ANNs are used as complete\partial replacements for the structural analysis of reinforced concrete 2D frames. The results show that using ANNs as surrogates for structural analysis is impractical since the training time grows exponentially with the size of the frame. On the other hand, the Weight Matrix method relies on linear algebra, vector operations, and sparse matrix algorithms to produce a framework that allows for efficiently updating the global stiffness matrices of plane frame structures when needed. Results show that the Weight Matrix method is more computationally efficient than the conventional method of constructing the global stiffness matrices of 2D frames.

Furthermore, the genetic algorithm (GA) was used to simulate the iterative design process. GAs are population based optimization methods founded on the concept of natural selection. In each iteration, the structural responses and costs are evaluated for a population of design alternatives, and the designs with the most favorable costs and structural responses are selected and combined to produce a new population for the next iteration. However, like many population-based optimization algorithms, GAs suffer from the issue of premature convergence. During premature convergence, the population collapses to copies of a single individual, and the algorithm becomes trapped. To improve the convergence of the GA, this research explored population diversity maintaining mechanisms found in the GA literature. Finally, this research will investigate the constructability of the solutions obtained from GAs by considering the effect of buildability factors (e.g. formwork surface area, quantity of reinforcement, concrete volume …) on the total cost of the design alternatives.

ACKNOLEGMENTS……………………………………………..……v

ABSTRACT……………………………………………….………vi

ILLUSTRATIONS……………………………………..……….x

TABLES…………………………………………..…………….…..xiv

Chapter

1. INTRODUCTION 1

1.1. Background...1

2. LITERATURE REVIEW..6

2.1. Common Efficient Structural Reanalysis Methods 6

2.2. Efficient Methods for Constructing the Global Stiffness Matrix 8

2.3. Artificial Neural Networks..13

2.4. Neural Networks in Structural Engineering 16

2.5. Heuristic Optimization Methods..18

2.6. Uses of Heuristics in the Optimization of Reinforced Concrete Frames..19

2.7. The Genetic Algorithms 21

2.8. Objective Function for the Cost Optimization of Reinforced Concrete Frames...23

2.9. Premature Convergence of the Genetic Algorithm and Population Diversity...25

3. METHODOLOGY 30

3.1. Training ANNs to Predict the Structural Response of Frames................30

3.2. Novel Method for Constructing the Global Stiffness Matrix of 2D Frames using Neural Networks 31

3.3. Novel Weight Matrix Method for Efficient Updating the Global Stiffness Matrices of Structures..33

3.4. Design Dead, Live, and Seismic Loads and Load Combinations............56

3.5. Integrating the Genetic Algorithm into Structural Design 56

3.6. Choosing the Diversity Maintaining Mechanism 83

3.7. Assessing Buildability and the Productivity of Workers 85

3.8. Assessing the Effect of Buildability on the Cost Optimization Process ...88

3.9. Unit Material Costs of the Frames...89

3.10. Frames Considered In to Study The Diversity Maintaining Mechanisms and Buildability Factors..89

4. RESULTS AND DISCUSSIONS 91

4.1. ANN Surrogate for Symmetric Two Bay Frame Subjected to Gravity Load 91

4.2. Results for Two Story, Three Story, and Four Story Two Bay Frames....95

4.3. Constructing the Stiffness Matrix of a Frame Using a Neural Network...97

4.4. Comparing the Performance of Neural Network Approaches and the Vectorized Method for Constructing the Stiffness Matrix of a Frame..........101

4.5. Comparison of the Conventional, Vectorized, and Weight Matrix Methods 102

4.6. Comparing the Performance of the Diversity Maintaining Mechanisms on the Penalized Cost..105

4.7. Comparing the Performance of the Diversity Maintaining Mechanisms on the Convergence Time 111

4.8. Comparing Tournament Selection and Rank Selection 116

4.9. The Time Saved by The Weight Matrix Method on GA Optimizations.122

4.10. Study of the Influence of Buildability Factors on the Total Man Hours and Material Costs of the Frames 123

5. CONCLUSIONS AND FUTURE WORK 127

5.1. Conclusions 127

5.2. Future Work...129

REFERENCES 131

ILLUSTRATIONS

Figure Page

1. Constant strain triangle with three degrees of freedom (DOFs) 10

2. Architecture of a single artificial neuron (Haykin, 2008)..14

3. A neural network with two hidden layers (Haykin, 2008) 14

4. Pareto front of Cost versus Displacement (Babaei & Mollayi, 2016)......................19

5. Flowchart of the genetic algorithm process (Zhou, 2012).......................................21

6. Binary and Real Encoding 22

7. Crossover example 23

8. A random number is sampled from the uniform distribution F = [fmin, fmax], and the individual with the nearest fitness value is selected (Hutter & Legg, 2006) 29

9. Architecture of ANN global stiffness matrix assembler..33

10. Line element with four degrees of freedom 34

11. A structure composed of two line elements and having six degrees of freedom......38

12. A MATLAB code for building the weight matrix of a plane frame made of Bernoulli elements 55

13. Reinforcement configuration in Beams 60

14. Configuration of reinforcement in a typical column section 61

15. Free body diagram used to calculate the shear induced when the probable moment capacities are reached (Moehle, 2014) 64

16. Interaction diagram with expected capacity at point A and applied load at point B (Kaveh & Sabzi, 2012) 68

17. Free body diagrams of the joints subjected to the nominal moment capacities of the beams and columns (Moehle, 2014)..69

18. Free body diagram for calculating the columns' shear when the beams reach their probable moment capacities (Moehle, 2014) 71

19. Free body diagram used for calculating the joint shear induced when the beams' reinforcements reach their probable capacities (Moehle, 2014) 72

20. Adjustment of the beams longitudinal reinforcements' positions to avoid clashes with the columns reinforcement inside the beam-column joint 74

21. Reduction of the available spacing Sai to ensure the beam's primary reinforcements' spacing requirements are satisfied when the secondary reinforcement bars are placed 76

22. Sorting the columns' dimensions to satisfy connectivity requirements 77

23. Probable moment diagrams for (a) Reversing plastic hinges (b) non reversing plastic hinges due to larger spans or heavier gravity loads (Moehle, 2014) 78

24. The moment diagram from structural analysis (grey), and the moment diagram resulting from satisfying EQ. (98) (black), the positive probable moment is chosen such that the shear is positive 79

25. Typical beam and column cross sections used when creating the beam, column, and joint groupings 82

26. Flow chart summary of integrating the genetic algorithm into structural design 83

27. Frames considered in this study 90

28. Inputs of two bay-one story frame neural network surrogate 91

29. Outputs of two bay-one story frame neural network surrogate 92

30. Two bay frame surrogate neural network architecture 93

31. Mean squared error for training and validation data for a neural network with 20 neurons per hidden layer, and trained on 300 training example of a two bay one story frame. 94

32. Mean squared error for training and validation data for a neural network with 20 neurons per hidden layer, and trained on 1000 training example of a two bay one story frame 94

33. Comparison of the performance of ANNs to the Direct Stiffness Method for frames of different sizes 96

34. The time needed to training neural networks to simulate the analysis of frames of different sizes 97

35. Neural network training time as a function of the number of DOFs....................98

36. Neural network prediction time in comparison to the standard method of assembling the stiffness matrix 99

37. Extrapolation of the Standard and Neural network processing times shows a decrease in the utility of using neural networks as the number of DOFs increases..99

38. Comparison of processing time of neural networks with frame subdivisions to the previous methods....................101

39. Comparison of the performance of the vectorized stiffness algorithm to other methods of evaluating the stiffness matrix 102

40. Performance of the Conventional, Vectorized, and Weight Matrix stiffness matrix assembly methods as a function of the number of degrees of freedom of the plane frame structures....................103

41. Comparison between the times needed to perform linear static structural analysis using the conventional method, neural networks, vectorized stiffness method, the weight matrix method....................105

42. The variation of the average penalized cost of a two story frame as a function of the population size....................106

43. The variation of the average penalized cost of a five story frame as a function of the population size....................108

44. The variation of the average penalized cost of a ten story frame as a function of the population size....................109

45. Mean of the average penalized cost of all of the tested diversity maintaining selection algorithms....................111

46. The variation of the convergence time of a two story frame as a function of the population size....................112

47. The variation of the convergence time of a five story frame as a function of the population size....113

48. The variation of the convergence time of a ten story frame as a function of the population size....114

49. Mean of the convergence times of all of the tested diversity maintaining selection algorithms 116

50. The average penalized cost as a function of the population size for a two story frame for rank and tournament selection 117

51. The average penalized cost as a function of the population size for a five story frame for rank and tournament selection 118

52. The average penalized cost as a function of the population size for a ten story frame for rank and tournament selection 119

53. five bay-fifteen story frame....120

54. The average penalized cost as a function of the population size for a fifteen story frame for rank and tournament selection 120

55. Comparison between the time needed to complete one GA iteration using the Conventional and Weight Matrix Method to construct the stiffness matrix....123

56. Comparison of man-hours for frames optimized without considering buildability and frames optimized considering buildability with a labor cost of $50 per man hour 124

57. Comparison of man-hours for frames optimized without considering buildability and frames optimized considering buildability with a labor cost of $100 per man hour 125

58. Comparison of material costs for frames optimized without considering buildability and frames optimized considering buildability with a labor cost of $50 per man hour 125

59. Comparison of material costs for frames optimized without considering buildability and frames optimized considering buildability with a labor cost of $100 per man hour....126

TABLES

Table Page

1. Beam design variables 58
2. Column design variables 61
3. Input ranges for frame structural analysis surrogate neural networks 92
4. Comparison of the performance of different ANNs trained on a different amount of training data 94
5. Time needed by the direct stiffness method versus the neural network to predict the internal forces in a two bay frame 95
6. Summary of the speedups of the Vectorized and Weight Matrix methods relative to the Conventional method of assembling the global stiffness matrix 103
7. Average Penalized Cost at Plateau using Rank and Tournament Selection 121
8. Optimization Time for Last Increment of Population Size using Rank and Tournament Selection 121
9. Cumulative Optimization Time of different frames using Rank and Tournament Selection 122

CHAPTER 1

INTRODUCTION

1.1. Background

Reinforced Concrete special moment resisting frames are part of the lateral load resisting systems of buildings designed to sustain seismic loads. The beams, columns, and joints are designed to resist the flexural, axial, and shear loads resulting from the building swaying as a result of the strong ground shaking. Proportioning and detailing the members of such frames must adhere to special requirements found in building codes (Moehle, 2014).

The structural design process of special moment resisting frames entails analyzing numerous design alternatives with the goal of finding the most economic and safe design. Traditional design suffers from two main issues. First, the traditional structural analysis methods, like the direct stiffness method, become computationally expensive for large structures (Waszczysznk, 1999). Second, the traditional design process relies on prior experiences, which may push the designer towards sub-optimal designs (Paya-Zaforteza, Yepes, Hospitaler, & González-Vidosa, 2009). Therefore, the objective of this research is to produce a computationally efficient framework that is capable of automating the structural design process.

To improve the efficiency of structural analysis, researchers have developed reanalysis techniques based on Tylor series expansions and sensitivity analysis to quantify the change in the structural response due to local changes in the material and geometric properties of an initial design (Balling, 1991; Sun, Liu, Xu, Zhang, & Zuo, 2014; Zuo, Bai, & Yu, 2016). The main limitation of such methods is that the predictions of the structural response become unreliable when the material and geometric properties deviate significantly from the initial design.

Furthermore, researchers have relied on various surrogate model approaches to replace computationally costly procedures. Some commonly used surrogate models are the Response Surface Methodology (RMS), Kriging, and Artificial Neural Networks (ANNs) (Heap, 2013; Loja, Barbosa, & Mota Soares, 2014; Pilkington, Preston, & Gomes, 2014; Vicario, Craparotta, & Pistone, 2016). More recently, ANNs are being used excessively in various engineering disciplines due to their ability to employ learning algorithms, their parallelism, and their ability to model highly complex relationships (Haykin, 2008; Pilkington et al., 2014). In structural engineering, researchers have used ANNs to predict the response of various structures including small truss structures, beams and columns, beam-column joints, and existing buildings subjected to variable seismic loads (Afaq Ahmad, 2017; Kao & Yeh, 2016; Kotsovou, Cotsovos, & Lagaros, 2017; Perez-Ramirez et al., 2019; Wu & Jahanshahi, 2019). This research suggests investigating the use of ANNs as replacements for the structural analysis of reinforced concrete 2D frames where the geometric properties, material properties, and the loads are input variables.

One limitation of using ANNs is that training neural networks with variable input and output sizes is still a field of research that is applicable to only some special cases (Baldi, 2018; Ghosh, Das, & Nasipuri, 2019; Vinyals, Fortunato, & Jaitly, 2015). This is an issue when creating surrogates for the structural analysis of frames because frames of different sizes have a different number of geometric properties as input variables and a different number of internal forces as output variables. As a result, different neural networks will need to be trained for frames having a different number of geometric properties and internal forces. This limitation makes quantifying the growth of the neural network’s training time as a function of the size of the frames critical for determining the feasibility of using neural networks as surrogate models for the structural analysis of frames.

Therefore, methods that allow reusing neural networks trained on small structures to predict the response of larger structures are interesting to investigate.

To allow for the reusability of neural networks, this research also proposes a novel method for training neural networks to predict the global stiffness matrices of 2D frames. Neural networks benefit from the power of parallel computing which could accelerate the assembly of the stiffness matrices of 2D frame structures. Furthermore, the direct stiffness method allows for constructing the stiffness matrix of a more complex structure from the stiffness matrices of simpler structures (Kassimali, 2011). Therefore, neural networks trained on simpler frames can be used to build the global stiffness matrix of more complex frames.

Then, this research will investigate the use of neural networks as surrogate models for the analysis of 2D frame structures and as tools that accelerate the assembly of the global stiffness matrix of 2D frame structures.

Apart from relying on ANNs to improve the computational efficiency of structural analysis, vectorized methods have been adopted in the literature to accelerate the computational efficiency of constructing the global stiffness matrices of frames. Vectorized methods increase computational efficiency by replacing programming loops with multithreaded vector operations. A novel vector based method known as the Weight Matrix method is developed and tested in this thesis. The Weight Matrix method relies on linear algebra, vector operations, and sparse matrix algorithms to produce a framework that allows for efficiently updating the global stiffness matrices of plane frame structures when needed by the designer.

To minimize the human involvement in the design process of reinforced concrete frames, researchers have opted to using various optimization algorithms to minimize the costs of frame structures (Slobbe, 2015). Optimization is the procedure of finding the best alternative from a set of available solutions (Hoos & Stutzle, 2004). Optimization algorithms are either gradient based or heuristic. Gradient based optimization requires that the problem can be expressed as a continuous and differentiable function (Leng, 2016). However, in many real life problems, an analytical form of the gradient is not guaranteed. To resolve this issue, numerous derivative free heuristic algorithms have been developed to find near optimal solutions for real life problems (F.-S. Wang & Chen, 2013). Examples of heuristic optimization algorithms that exist in the literature include simulated annealing, particle swarm intelligence, and the genetic algorithm (Hoos & Stutzle, 2004; Reyes & Steidley, 1998; Rozenberg, Back, & Kok, 2012; D. Wang, Tan, & Liu, 2018).

The genetic algorithm (GA) is one of the most commonly used heuristic techniques because its flexibility and ease of implementation make it suitable for a wide range of problems (Rozenberg et al., 2012; Slobbe, 2015; Zhou, 2012). GA is inspired by the theory of evolution where species improve through the survival of the fittest (Hoos & Stutzle, 2004; Rozenberg et al., 2012; Zhou, 2012). In a GA, a cost function is evaluated for a population of design alternatives, and then the designs with the best performances on the cost function are selected to produce a new population.

Research on the use of GAs in structural design shows promising results; however, the structural engineering literature rarely addresses two important issues. First, the genetic algorithm suffers from the problem of premature convergence where the population collapses to copies of a single design trapping the algorithm in a small region of the design space (Sudholt, 2018). Premature convergence occurs due to the loss of diversity in the population, so diversity maintaining methods available in the GA literature will be investigated to address this issue. Second, the constructability of the designs obtained from GAs has not been properly explored. Constructability will be addressed by modifying the cost function to include the effects of various buildability factors (e.g. formwork surface area, quantity of reinforcement, concrete volume …) on the total cost of the design alternatives.

CHAPTER 2

LITERATURE REVIEW

2.1. Common Efficient Structural Reanalysis Methods

Reanalysis methods use the response of an initial structure to evaluate the response for a modified structure without the need to resolve the system of linear equations (Chipperfield, Vance, & Fischer, 2006). Commonly used reanalysis techniques include the Taylor series expansion, and a method known as Combined Approximation (Chipperfield et al., 2006).

The Taylor series approximation is the most commonly used reanalysis technique (Chipperfield et al., 2006). Balling (1991) used the Taylor series expansion to approximate the structural response of steel frames in order to reduce the time needed by the simulated annealing algorithm to find the lightest steel frame design. The results showed that the Taylor series only resulted in an acceptable approximation when the changes in the geometry from the initial design to the final optimal deign were small. Chipperfield, Vance, & Fischer (2006) used the Taylor series expansion and the initial response of a cantilever beam to predict the responses of beams with similar cross sections but different lengths. The results also showed that the Taylor series is only useful when the change in the length of the beam was small relative to the initial design.

Another widely used reanalysis technique is the combined approximation (CA) method (Chipperfield et al., 2006; Li, Jia, Yu, & Li, 2018; Sun et al., 2014; Zuo et al., 2016). In the CA method, a binomial series is used to generate a set of basis vectors (r_i) , and then Reduced Basis Approximation (RBA) is used to estimate the response (d) of a modified structure. According to RBA, the response (d) of the modified structure can be approximated as a linear combination of the basis vectors such that $= \sum_{1}^{s} w_i r_i$, where w_i are coefficients to be determined, and s is the number of basis vectors specified by the user. The idea behind this method is that for a reasonable change in the global stiffness matrix, the number of basis needed (s) would be much less than the number of degrees of freedom (n). Therefore, a smaller $s \times s$ system would need to be solved instead of the large $n \times n$ force-displacement system. Li et al. (2018) used the CA method to perform nonlinear time history analysis on a frame structure. Their results showed that the CA method is an order of magnitude faster than performing LDL^T factorization of the stiffness matrix. Sun et al. (2014) provide a method to determine the number of basis needed to get a high quality approximation from CA, and they showed that the CA method is more efficient than factorizing the stiffness matrix only when the number of basis needed is less than 10% of the number of degrees of freedom. The CA method suffers some significant drawbacks. First, the basis vectors generated by the binomial are not guaranteed to be linearly independent resulting in an ill conditioned problem (Sun et al., 2014). Therefore, the quality of the solution may not improve regardless of the number of basis used. Second, the CA method has been known to be equivalent to a preconditioned conjugate gradient with a preconditioner equal to the inverse of the stiffness matrix of the initial structure (Chipperfield et al., 2006; Sun et al., 2014).

2.2. Efficient Methods for Constructing the Global Stiffness Matrix

2.2.1. The Conventional Method for Constructing the Sparse Global Stiffness Matrix

The conventional methods for constructing the global stiffness matrices of frame structures are sequential in nature and consist of the following steps (Kassimali, 2011; McGuire, Gallagher, & Saunders, 1982). First, the geometric and material properties of the beam or column element are used to assemble the element stiffness matrix in the local coordinate system of the structural elements. Second, a transformation matrix is applied to transform the element's stiffness matrix from local coordinates to global coordinates. Finally, the element stiffness matrix is assembled into the global stiffness matrix. This procedure is repeated for all the beam and column elements of the frame structure.

Although the size of the global stiffness matrix of a structure with N degrees of freedom is $N \times N$, the sparse nature of the stiffness matrix implies that most entries are zeros. Chen (2011), Davis (2011), and (Cuvelier, Japhet, & Scarella, 2016) discuss using sparse matrix storage formats and algorithms to reduce the computation time and memory requirements for sparse stiffness matrices by avoiding operations on zero entries.

The *coordinate format* is one of the most intuitive sparse matrix storage schemes (Chen, 2011). In this storage scheme, the nonzero entries of the sparse matrix, their row indices, and their column indices are stored in separate vectors. Consider as an example the sparse 6×4 matrix A below.

$$A = \begin{bmatrix} 0 & 2 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad g = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}, \quad i = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 3 \end{bmatrix}, \quad j = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 3 \end{bmatrix} \qquad \text{EQ.(1)}$$

The nonzero entries of A are stored in the vector g, the row indices of those elements are stored in i, and their column indices are stored in j.

All programming languages have sparse matrix libraries that allow the user to efficiently store sparse matrices. Most implementations require the user to provide the *coordinate format* as input to a “Sparse_Builder” function, which has a definition similar to that in EQ.(2) (Chen, 2011). The “Sparse_Builder” functions use the *coordinate format* to create less intuitive but more compressed storage schemes such as the *Compressed Sparse Column* (CSC) storage scheme.

$$A = Sparse\_Builder \ \ (i, j, g) \qquad \text{EQ.(2)}$$

The “Sparse_Builder” function requires that g, i, and j vectors have the same length, but duplicate i and j pairs are allowed to exist. In cases where i and j pairs are repeated more than once, the “Sparse_Builder” function sums their corresponding g values (Chen, 2011). This summation feature is useful for constructing the stiffness matrices of structures since there are many cases where the stiffness at one node is influenced by the stiffness of several elements (Chen, 2011).

Therefore, to efficiently construct the global stiffness matrix of a structure, Chen (2011), Davis (2011), and (Cuvelier et al., 2016) recommend the following. Define G as the vector that holds all the nonzeros of the global stiffness matrix, define I as the vector of row indices of G in the global stiffness matrix, and define J as the vector of column indices of G in the global stiffness matrix. For each structural element, find the element stiffness matrix in global coordinates. Then, determine the vector of nonzero values g of the element stiffness matrix, the row position vector i of the nonzeros in the global stiffness matrix, and the column position vector j of the nonzeros in the global stiffness matrix. Using the proper transformation operator, insert the values of g into the vector G, insert the row indices i into the vector I, and insert the column indices j into the vector J. Finally, use the "Sparse_Builder" function on $I, J,$ and G to store the global stiffness matrix in a sparse format.

2.2.2. Vectorized Stiffness Matrix Approach to Constructing the Global Stiffness Matrix

The vectorized method by Cuvelier et al. (2013) was hardcoded in MATLAB specifically for meshes made of scalar constant stain triangular elements, but the method can be extended to meshes made from different kinds of finite elements. Consider the triangular element m as shown in Figure 1. The element has three nodes $n1, n2,$ and $n3$. Since this is a scalar triangular element, the degrees of freedom associated with this element are $n1, n2,$ and $n3$.

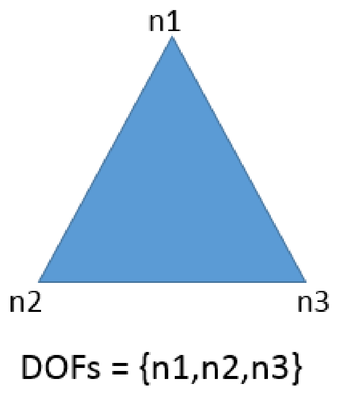


Figure 1 Constant strain triangle with three degrees of freedom (DOFs)

The stiffness matrix of an element m is of the form shown in EQ.(3)

$$K_e^m = \begin{array}{c c} \begin{matrix} n1 & n2 & n3 \end{matrix} & \\ \begin{bmatrix} k_1^m & k_4^m & k_7^m \\ k_2^m & k_5^m & k_8^m \\ k_3^m & k_6^m & k_9^m \end{bmatrix} & \begin{matrix} n1 \\ n2 \\ n3 \end{matrix} \end{array} \qquad \text{EQ.(3)}$$

Using the *coordinate format,* the nonzeros of the element stiffness matrix and their locations in the global stiffness matrix can be stored as shown in EQ.(4)

$$k_{nz}^m = \begin{bmatrix} k_1^m \\ k_2^m \\ k_3^m \\ k_4^m \\ k_5^m \\ k_6^m \\ k_7^m \\ k_8^m \\ k_9^m \end{bmatrix}, \quad i^m = \begin{bmatrix} n1 \\ n2 \\ n3 \\ n1 \\ n2 \\ n3 \\ n1 \\ n2 \\ n3 \end{bmatrix}, \quad j^m = \begin{bmatrix} n1 \\ n1 \\ n1 \\ n2 \\ n2 \\ n2 \\ n3 \\ n3 \\ n3 \end{bmatrix} \qquad \text{EQ.(4)}$$

k_{nz}^m is the vector of nonzero values of K_e^m, and i^m and j^m are the row and column indices of k_{nz}^m in the global stiffness matrix of the structure.

For a mesh made up of n_{elem} elements, the goal of Cuvelier et al. (2013) was to find the most efficient way to construct the vectors K, I, and J defined in EQ.(5).

$$K = \begin{bmatrix} k_{nz}^1 \\ k_{nz}^2 \\ k_{nz}^3 \\ \vdots \\ k_{nz}^{n_{elem}} \end{bmatrix}, \quad I = \begin{bmatrix} i^1 \\ i^2 \\ i^3 \\ \vdots \\ i^{n_{elem}} \end{bmatrix}, \quad J = \begin{bmatrix} j^1 \\ j^2 \\ j^3 \\ \vdots \\ j^{n_{elem}} \end{bmatrix} \qquad \text{EQ.(5)}$$

The trivial solution is loop on all n_{elem} of the mesh in a sequential manner, find the k_{nz}^m, i^m and j^m vectors, and then insert these vectors into K, I, and J

respectively. This is the process followed by the conventional method of assembling the stiffness matrix discussed in Section 2.2.1.

Instead, Cuvelier et al. (2013) opt with a hardcoded efficient solution. They introduce the matrices K_g, I_g, and J_g as defined in EQ.(6).

$$K_g = [k_{nz}^1 \quad k_{nz}^2 \quad k_{nz}^3 \quad \dots \quad k_{nz}^{n_{elem}}]$$

$$I_g = [i^1 \quad i^2 \quad i^3 \quad \dots \quad i^{n_{elem}}] \qquad \text{EQ.(6)}$$

$$J_g = [j^1 \quad j^2 \quad j^3 \quad \dots \quad j^{n_{elem}}]$$

Each row l of K_g can be expressed as shown in EQ.(7).

$$K_g(l,:) = [k_{nz}^1(l) \quad k_{nz}^2(l) \quad k_{nz}^3(l) \quad \dots \quad k_{nz}^{n_{elem}}(l)] \qquad \text{EQ.(7)}$$

$K_g(l,:)$ contains the l^{th} nonzero element of the element stiffness matrices of all n_{elem}. By virtue of occupying the same location in the local stiffness matrix, all elements of $K_g(l,:)$ share the same symbolic form. Therefore, for each value of $l = \{1, 2, 3 \dots 9\}$, $K_g(l,:)$ can be calculated using element-wise matrix operations.

Element-wise matrix operations perform the same mathematical operation on all elements of the matrix. For example, consider the vectors

$$B = [b_1, b_2], \qquad C = [c_1, c_2], \qquad D = [d_1, d_2]$$

The element-wise operation $\frac{B*C}{D} + B$ will produce the vector

$$\left[\frac{b_1 * c_1}{d_1} + b_1, \frac{b_2 * c_2}{d_2} + b_1\right]$$

The advantage of using element-wise matrix operations is that they are multithreaded in vector based languages such as MATLAB. Furthermore, only six operations are needed to construct K_g if symmetry of the element stiffness matrices is considered.

I_g and J_g can be constructed similarly to K_g, and then K, I, and J can be recovered.

2.3. Artificial Neural Networks

2.3.1. Definition and Basics Architecture

Neural networks are function approximators that have been proven to be capable of approximating any function to any desired degree of accuracy for a given range of the inputs (Haykin, 2008). Neural networks learn by being exposed to training examples of the inputs and outputs of the function they need to approximate, and then they are able to generalize and give reliable predictions on examples not encountered during training (Haykin, 2008). However, their effectiveness in generalization is limited to interpolation in the range of the inputs they were trained on (Haykin, 2008). Furthermore, ANNs process information in parallel making them good candidates to replace computationally costly conventional methods (Rafiq, Bugmann, & Easterbrook, 2001).

Any neural network consists of basic computing units called artificial neurons. Figure 2 shows the architecture of a single artificial neuron. Neurons transform a set of inputs x_k's into a response z_k. Each input x_k is multiplied by a weight w_{kj}, the weighted sum is obtained and added to a bias term b_{kj} , and then the result is passed through an activation function φ to limit the range of the output z_{kj}. (Haykin, 2008)

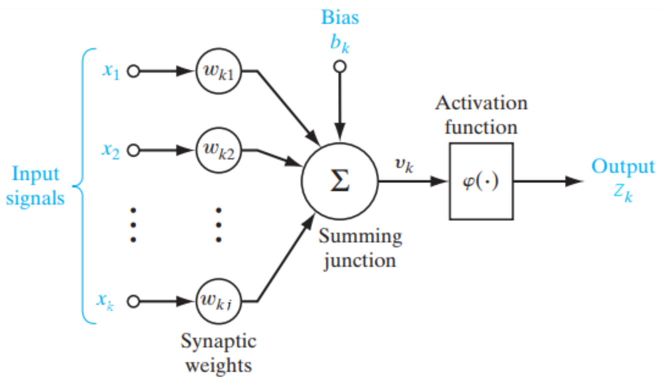


Figure 2 Architecture of a single artificial neuron (Haykin, 2008)

Neural networks are organized as layers each containing a specific number of neurons (Haykin, 2008). Figure 3 shows a basic example of a neural network. The inputs are presented at the input layer, the hidden layers process the inputs, and the predictions of the neural network are presented at the output layer (Haykin, 2008). The number of hidden layers and the number of neurons per hidden layer are hyper parameters determined by the user and specific to each problem (Haykin, 2008). Neural networks with two hidden layers have demonstrated the best performance in past experiments (Haykin, 2008).

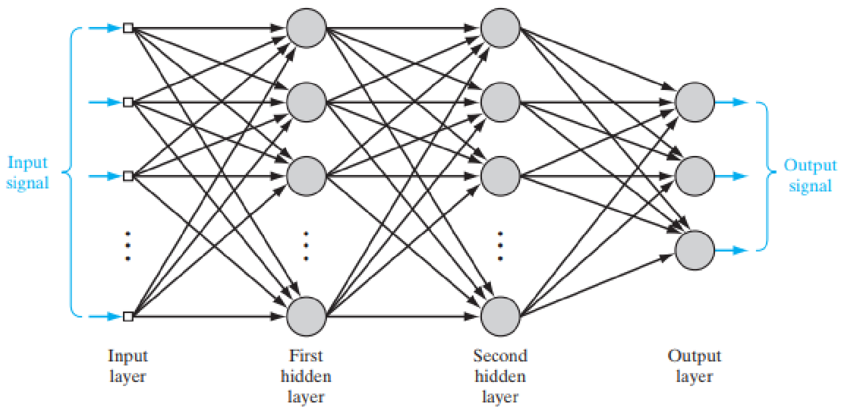


Figure 3 A neural network with two hidden layers (Haykin, 2008)

2.3.2. Training Artificial Neural Networks

The training process of a neural network involves adjusting the weights and biases of the neurons to reduce the prediction error (Haykin, 2008). The most commonly used error function is the mean-squared error because it is continuous and differentiable over the whole domain (Haykin, 2008). To steer the training algorithm in the right direction, information about the sensitivity of the prediction error to changes in the weights is needed (Haykin, 2008). Back-propagation is the most accepted method for obtaining the effect of local changes in weights on the prediction error (Haykin, 2008).

In this research, the training of the neural networks was performed in MATLAB’s Neural Network Toolbox. This toolbox provides a broad variety of training algorithms for performing Back-propagation. The MATLAB documentation encourages using the Scaled Conjugate Gradient algorithm since it has a similar convergence rate to second order optimization algorithms such as the Levenberg-Marquardt algorithm, but requires significantly less computational requirements.

2.3.3. Division of Data Set, Overfitting, and Underfitting

The data is divided into training, validation, and testing sets (Basheer & Hajmeer, 2000). The training set and validation set are used during the training phase of the neural network (Basheer & Hajmeer, 2000). The training set is used in the process of back propagation to update the weights and biases in order to minimize the error (Basheer & Hajmeer, 2000; Haykin, 2008). After the weights have been updated, the prediction error is calculated on the validation set. When the training process is complete, the performance of the neural network is evaluated on a testing set of examples not encountered in the training and validation sets (Basheer & Hajmeer, 2000; Haykin, 2008).

While training, if the prediction error on the validation set starts to increase past the prediction error on the training set, then the neural network is overfitting (Haykin, 2008). Overfitting is when the neural network memorizes the training set and fails to generalize leading to poor predictions on problems not encountered during training (Haykin, 2008). The solution to overfitting is providing more data or decreasing the size of the neural network (Haykin, 2008). As a result, training is conducted on neural networks of different architectures and using training sets with varying sizes.

If the training process shows no sign of overfitting, but the errors calculated on the training, validation, and testing sets show a poor performance, then the neural network is underfitting. Underfitting occurs when the size of the neural network is small resulting in an insufficient number of weights and biases to approximate the function. The solution is to the problem of under fitting is increasing the size of the neural network.

2.4. Neural Networks in Structural Engineering

Kao & Yeh (2016) proposed integrating feed forward neural networks into the optimization of structures as a less computationally expensive substitute to structural analysis. However, their work only investigated the integration of neural networks for the analysis of trusses with ten varying cross sectional areas as input variables. Their methodology consisted of the following steps. First, design alternatives are generated randomly. Each design alternative has a unique set of design variables X (e.g. different cross section sizes). Second, structural analysis is employed to analyze all the generated design alternatives to obtain their internal forces and displacements. The internal forces and the displacements are the response variables Y. Third, a neural network is trained to fit the relationship Y = f(X) between the input and response variables. Finally, the trained neural network model is used in an optimization algorithm to reach the most cost optimal design.

Afaq Ahmad (2017) used feed forward neural networks to simulate the nonlinear behavior of reinforced concrete rectangular beams, T-beams, columns, and slab panels. He used test data obtained from experiments on a wide range of structural elements. The geometric and material properties were used as inputs, and the ultimate capacity and mode of failure are the response variables.

Kotsovou, Cotsovos, & Lagaros (2017) relied on feed forward neural networks to predict the mode of failure and the load bearing capacity of reinforced concrete beam-column joints. The data used in training the neural networks is from over 150 experiments on RC beam-column joints. The input variables considered are the concrete compressive strength, the steel yield stress, the dimensions of the beam and column forming the joint, the amount of reinforcement in the beam and column, and the reinforcement in the joint.

Perez-Ramirez et al. (2019) used a recurrent neural network to predict the acceleration response of an existing building. Acceleration response data was collected from the 25th, 30th, and 34th floor of a thirty-eight story scaled model of a building. The data from the 25th and 34th floors was used as inputs to the recurrent neural network and the data from the 30th floor was the output. The data was split into a training set used to train the neural network and a validation set used to verify the reliability of the model.

The model performed accurately on both sets of data showing that once a recurrent neural network is trained, it can be used to replace a sensor in a building.

Wu & Jahanshahi (2019) compared the performance of feed forward neural networks and convolutional neural networks in predicting the dynamic response of existing structures. The input data is composed of vectors containing values of the velocity and acceleration at one hundred instances of time. The output is a vector containing the displacements of the structure at one hundred instances of time. The results showed that convolutional neural networks performed better than feed forward networks when the data contained noise. This is because the convolutional filters are capable of removing the noise while extracting the important trends in the data.

2.5. Heuristic Optimization Methods

Heuristic optimization techniques are derivative free computational procedures that seek optimal solutions by iteratively improving candidate solutions with respect to a certain metric (F.-S. Wang & Chen, 2013). Heuristic optimization techniques make few or no assumptions about the problem being optimized and search large design spaces to achieve a close to optimal solution with a reasonable computation time. The most prominent heuristic optimization methods found in the literature are the simulated annealing algorithm, particle swarm intelligence, and the genetic algorithm (Hoos & Stutzle, 2004; Reyes & Steidley, 1998; Rozenberg et al., 2012; D. Wang et al., 2018).

2.6. Uses of Heuristics in the Optimization of Reinforced Concrete Frames

Esfandiari, Urgessa, Sheikholarefin, & Dehghan Manshadi (2018) utilize the particle swarm algorithm to optimize the cost of 2D reinforced concrete frames subjected to time-history loadings. The authors integrated a decision-making module into the optimization process to bias the particle swarm algorithm towards choosing larger or more heavily reinforced sections when the demand to capacity ratio of a structural member is more than one. The results show that the decision-making module improves the convergence of the particle swarm algorithm.

Babaei & Mollayi (2016) used a multi-objective genetic algorithm to achieve a trade-off between the cost and lateral displacements of reinforced concrete 2D frames. They used the multi-objective algorithm to plot the pareto front of the displacement versus total cost as shown in Figure 4. The pareto front represents the set of solutions where improvements to the cost will harm the performance on displacement and vice versa. The results show that an interval exists where investing money in larger sections leads to significant improvements in displacement. However, beyond this interval, the displacement becomes insensitive to cost.

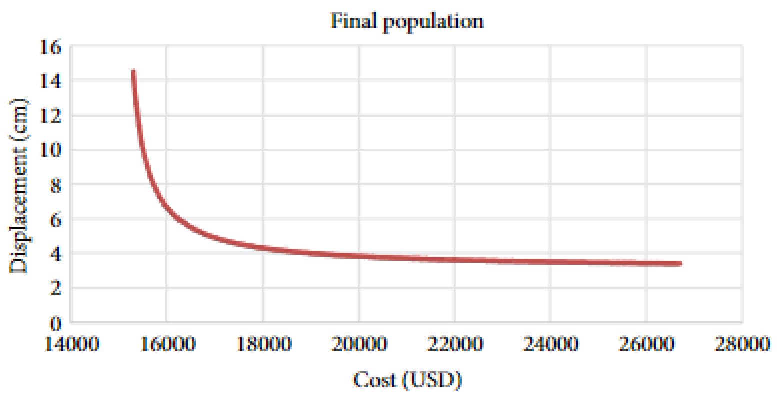


Figure 4 Pareto front of Cost versus Displacement (Babaei & Mollayi, 2016)

Paya, Yepes, González-Vidosa, and Hospitaler, (2008) used a multi-objective simulated annealing algorithm to find designs of reinforced concrete 2D frames that balanced between cost, carbon dioxide footprint, and the constructability. The main issue with this study is that constructability was determined in a simplistic manner. The authors assumed that constructability is represented by the number of longitudinal bars neglecting the effects of repeated members, the area of formwork, and the volume of concrete among other factors. Their study showed no significant change in the cost of reinforced concrete frames when taking the carbon dioxide footprint and the constructability into consideration.

In 2009, Paya-Zaforteza, Yepes, Hospitaler, and González-Vidosa used the simulated annealing algorithm to determine the relationship between a building's imbedded CO2 and total cost. For this purpose, they optimized a set of buildings once for imbedded CO2 and another time for total cost. They plotted the imbedded CO2 levels against the total cost, and their results showed that imbedded CO2 and total cost are closely related and optimizing for cost results in an environmentally friendly building.

Lee & Ahn (2003) and Kwak and Kim (2008) used the genetic algorithm with a predetermined sections database to optimize the cost of reinforced concrete 2D frames. The authors formulate a database containing all possible cross sections and sort them in order of their moment resisting capacity. The advantages of this method is that the predetermined sections already satisfy requirements of reinforcement spacing and quantity, and the sections' capacities do not need to be computed every iteration.

2.7. The Genetic Algorithms

The genetic algorithm is a population based optimization algorithm built on the concepts of evolution where species evolve through the survival of the fittest individuals. The genetic algorithm consist of the following steps: encode, define the objective function being optimized, initialize the population, evaluate the objective function, select the most fit individuals for the next generation, perform crossover reproduction and mutation. The main steps of the genetic algorithm are shown in Figure 5 (Hoos & Stutzle, 2004; Rozenberg et al., 2012; Zhou, 2012).

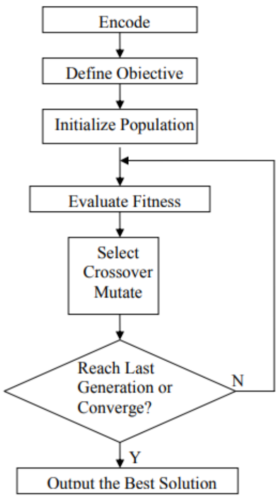


Figure 5 Flowchart of the genetic algorithm process (Zhou, 2012)

2.7.1. Encoding:

Encoding is the process of representing the design variables (e.g. height, width, amount of reinforcing…) of potential solutions as strings of code called chromosomes. A variable can be coded using either the binary code or the real code formats. Figure 6 shows an example of a potential solution with variables x1, x2, and x3 coded using both code formats. (Herrera, Lozano, & Verdegay, 1998; Rozenberg et al., 2012; Zhou, 2012)

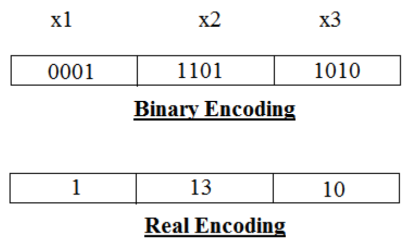


Figure 6 Binary and Real Encoding

2.7.2. Objective Function:

The objective function is used to evaluate the performance of potential solutions. The genetic algorithm can handle problems where a mathematical representation of the objective function is too complicated to formulate, so the values of the objective function can be obtained through computer simulations. (Rozenberg et al., 2012; Zhou, 2012)

2.7.3. Selection:

The selection operator is used to create copies of chromosomes that will be used later for crossover reproduction and mutation. Selection results in a new population of design alternatives with more favorable values of the objective function. The most commonly used selection method is proportional selection where the number of copies of each individual is proportional to its relative performance. EQ. (8) is used calculate the probability of selecting an individual during proportional selection, where f_i is the fitness of individual i. (Rozenberg et al., 2012; Zhou, 2012)

$$P = \frac{f_i}{\sum f_i} \qquad \text{EQ. (8)}$$

2.7.4. Crossover:

Crossover aims to combine the features of two parents with the possibility of good chromosomes producing better ones. The parent chromosomes exchange parts of their code to create new individuals which replace the parents. Figure 7 demonstrates an example of crossover. (Rozenberg et al., 2012; Zhou, 2012)

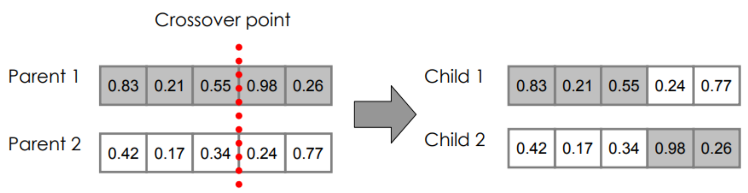


Figure 7 Crossover example

2.7.5. Mutation:

Mutation is the process that randomly alters a gene within a chromosome to ensure that the probability of reaching any point in the search space is not zero. (Herrera et al., 1998; Rozenberg et al., 2012; Zhou, 2012)

2.8. Objective Function for the Cost Optimization of Reinforced Concrete Frames

Most research has relied on the simple objective function in EQ. (9) to evaluate the quality of the Frame design alternatives (Camp, 2007; Esfandiari, Urgessa, Sheikholarefin, & Manshadi, 2018; Govindaraj & Ramasamy, 2005; Jahjouh, Arafa, & Alqedra, 2013; Kaveh & Sabzi, 2012). The objective accounts for the material costs of concrete ($C_{concrete}$), reinforcement steel bars (C_{steel}), and formwork ($C_{formwork}$) as presented in EQ. (10). Then, the material cost is penalized by the sum of violated constraints (G) obtained from EQ. (11), EQ. (12), and EQ. (13) .

$$C_{penalized} = C_{material}(1 + G)$$ EQ. (9)

$$C_{material} = C_{concrete} + C_{steel} + C_{formwork}$$ EQ. (10)

$$G = \sum \max\left(g_i^{upper}, 0\right) + \sum \max(g_i^{lower}, 0)$$ EQ. (11)

$$g_i^{upper} = \frac{(z_i - c_{upper})}{c_{upper}}$$ EQ. (12)

$$g_i^{lower} = \frac{(c_{lower} - z_i)}{c_{lower}}$$ EQ. (13)

In EQ. (12) and EQ. (13) z_i is a variable with an upper bound constraint c_{upper} and a lower bound constraint c_{lower}. g_i^{upper} and g_i^{lower} are constraint functions of the upper bound and lower bound respectively. These constraint functions are violated when they are strictly positive. EQ. (12) and EQ. (13) show that g_i^{upper} is positive when upper bound is violated and g_i^{lower} is positive when the lower bound is violated. The penalty G must not be negative because a negative penalty rewards the design alternative for satisfying the constraints at the expense of $C_{material}$. To ensure that the penalty G is always positive, $\max\left(g_i^{upper}, 0\right)$ and $\max(g_i^{lower}, 0)$ are used in EQ. (11).

2.9. Premature Convergence of the Genetic Algorithm and Population Diversity

2.9.1. The Problem of Premature Convergence

Most optimization problems such as the cost optimization of frame structures lead to multimodal solution spaces that require the genetic algorithm to identify multiple local optima (Sareni & Krähenbühl, 1998). A major issue with the genetic algorithm is premature convergence (Sareni & Krähenbühl, 1998). In the case of premature convergence, the population collapses to copies of a single individual leading the evolutionary algorithm to be trapped in a small region of the solution space (Sudholt, 2018). Having parents with diverse genetic material is necessary for crossover to create individuals that are different from their predecessors (Sudholt, 2018). Therefore, maintaining population diversity is essential for the success of any evolutionary algorithm.

2.9.2. Methods for Maintaining Population Diversity

2.9.2.1. <u>Tournament Selection</u>

Tournament Selection is a popular selection method capable of balancing between convergence speed and maintaining population diversity (Xie & Zhang, 2013) . In tournament selection, n individuals are selected from the large population (Miller & Goldberg, 1995). Those n individuals compete and the one with the highest fitness participates in the next generation (Miller & Goldberg, 1995). The number of individuals competing against each other is known as the tournament size (Miller & Goldberg, 1995). In Tournament Selection, the selection pressure towards fit individuals is determined by the tournament size (Xie & Zhang, 2013). Small tournaments maintain diversity but degrade the convergence speed, and large tournaments lead to loss in diversity and premature convergence. Therefore, in Tournament Selection, the tournament size needs to be carefully calibrated (Xie & Zhang, 2013).

2.9.2.2. <u>Rank Selection</u>

In Rank Selection, each chromosome is ranked based on its fitness from worst to best, and then the ranks are used to calculate the probability of selecting each chromosome. The probability of selecting a chromosome is calculated as shown in EQ. (14), where r_i is the rank of chromosome i. Rank selection is known to have a slower convergence than proportional selection, but it allows for maintaining the diversity of the population. (Lowen & Verschoren, 2008)

$$P = \frac{r_i}{\sum ri} \qquad \text{EQ. (14)}$$

2.9.2.3. <u>Rank-Space Method</u>

The Rank-Space method is another selection scheme developed for the purpose of maintaining population diversity (Winston, 1992). In this scheme, the objective function as well as the Euclidian distance between chromosome code strings are used to decide which individuals should be selected for the next generation. The Rank-Space procedure is summarized as follows:

1. Select the individual in the current population with the highest fitness, and add it to the new population.
2. Remove the selected individual from the old population.

3. For the rest of the selections, perform the following steps.
    a. Rank the individuals of the old population by their fitness. The highest ranking is given to the fittest individual.
    b. For each individual in the old population, find the sum of the Euclidian distances to all of the individuals in the new population.
    c. Rank the individuals of the old population by the sum of Euclidian distances. Individuals with the largest distance are given the highest rank since they are the most different from the individuals of the new population.
    d. For each individual in the old population, sum the Fitness rank and Euclidian distance rank to produce a combined rank. Sort the individuals in a descending order of combined ranks.
    e. Loop over the individuals in the old population. During each iteration of the loop, a random number is generated. If the random number is greater than a user defined probability of section Pc, then the individual is not selected, and the loop moves to the next individual. If the random number is less than Pc, then the individual is selected. The selected individual is added to the new population, it is removed from the old population, and the algorithm returns to step 3.a.

The downside of the Rank Space method is its O (n^3) time complexity with respect to the population size, so as the population size increases, the selection process becomes computationally expensive.

2.9.2.4. Fitness Uniform Sampling

Fitness Uniform Selection Strategy (FUSS) was developed by Hunter (2002) to address the difficulty evolutionary algorithms have in escaping the local optimum. Rather than having a selection process that favors the most fit individual, Hunter (2002) proposes the following scheme. Before selection, all individuals of the population are added to an archive so that no information is ever discarded. Then, fitness uniform selection takes place as follows. The archive is sorted by fitness from f_{min} to f_{max} and a fitness value is selected from the uniform random distribution $F = [f_{min}, f_{max}]$. The individual with a fitness value nearest to F is selected for the next generation. This selection process continues until the desired number of individuals is selected for the next generation. At first glance, there seems to be no selection pressure towards the more fit individuals while using FUSS. However, since FUSS selects uniformly, it can be shown that individuals with higher fitness are favored (Hutter, 2002). In the early stages of a typical evolutionary algorithm, there are many unfit individuals and a few fit ones making the high fitness levels less populated. The lower populated fitness levels lead to large fitness gaps as shown in Figure 8. Furthermore, the probability of F lying in a fitness gap is proportional to the size of the gap, so F is more likely to be in the fitness gaps created by the highly fit individuals. Therefore, the rare highly fit individuals are more likely to be selected. The main disadvantage of FUSS is the need to store and sort an indefinite number of individuals. As the size of the population archive increases, the sorting and selection processes becomes computationally expensive.

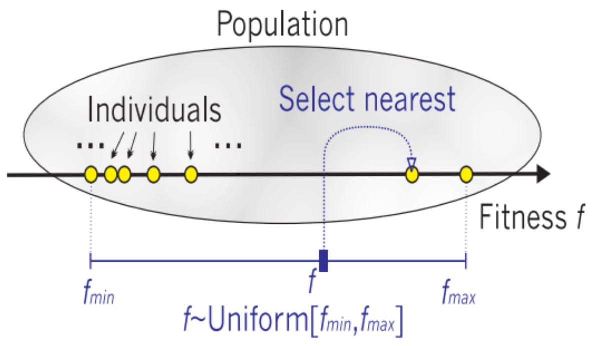


Figure 8 A random number is sampled from the uniform distribution F = [fmin, fmax], and the individual with the nearest fitness value is selected (Hutter & Legg, 2006)

CHAPTER 3

METHODOLOGY

3.1. Training ANNs to Predict the Structural Response of Frames

The following methodology is used to train neural networks to simulate the direct stiffness method for frames subjected to gravity loads.

a. A database of alternative designs is generated at random.

b. The direct stiffness method is applied on the generated alternatives to obtain the internal forces and moments.

c. The geometric and material properties of the beam and column elements (cross section Area, moment of inertia, and modulus of elasticity) are labeled as input variables of the neural network.

d. The internal forces and moments of the beams and columns are added to the database and labeled as output variables.

e. The database is divided to training, validation, and testing sets.

f. MATLAB's Neural Network Toolbox is used to create an ANN with two hidden layers. Start with a small number of neurons per hidden layer (10 to 20).

g. Select a training algorithm and start the training process.

h. If overfitting is observed, generate more random design alternatives and re-train.

i. If underfitting is observed add more neurons to the hidden layers and re-train.

j. Alter the size of the neural network and the size of the database until the performance of the neural network on the testing set is within the desired range. In order for the accuracy of neural networks to be competitive with exact methods, the neural networks will be required to make predictions with less than 5% percentage error on the testing set.

3.2. Novel Method for Constructing the Global Stiffness Matrix of 2D Frames using Neural Networks

Processing information in sequence makes traditional methods inefficient. Neural networks can process information in parallel making them good candidates for increasing the computational efficiency of the process of assembling the global stiffness matrix. To train neural networks for this task, a few realizations must be made about the nature of the global stiffness matrix of a frame.

The non-zero values in the global stiffness matrix of a 2D frame are linear combinations of values in the stiffness matrices of the beams and columns making the frame.

The stiffness matrix of a beam\column, n, can be reconstructed by knowing its four basis: $\frac{A_nE_n}{L_n}, \frac{E_nI_n}{L_n}, \frac{E_nI_n}{L_n^2}, \frac{E_nI_n}{L_n^3}$. Where A_n is the cross sectional area of element n, I_n is the moment of inertia of element n, E_n is the modulus of elasticity of element n, and L_n is the length of element n,

The relationship between the basis $\frac{A_nE_n}{L_n}, \frac{E_nI_n}{L_n}, \frac{E_nI_n}{L_n^2}, \frac{E_nI_n}{L_n^3}$ and the geometric and material properties of the beam A_n, E_n, I_n, L_n can be further simplified to linear combinations as shown below.

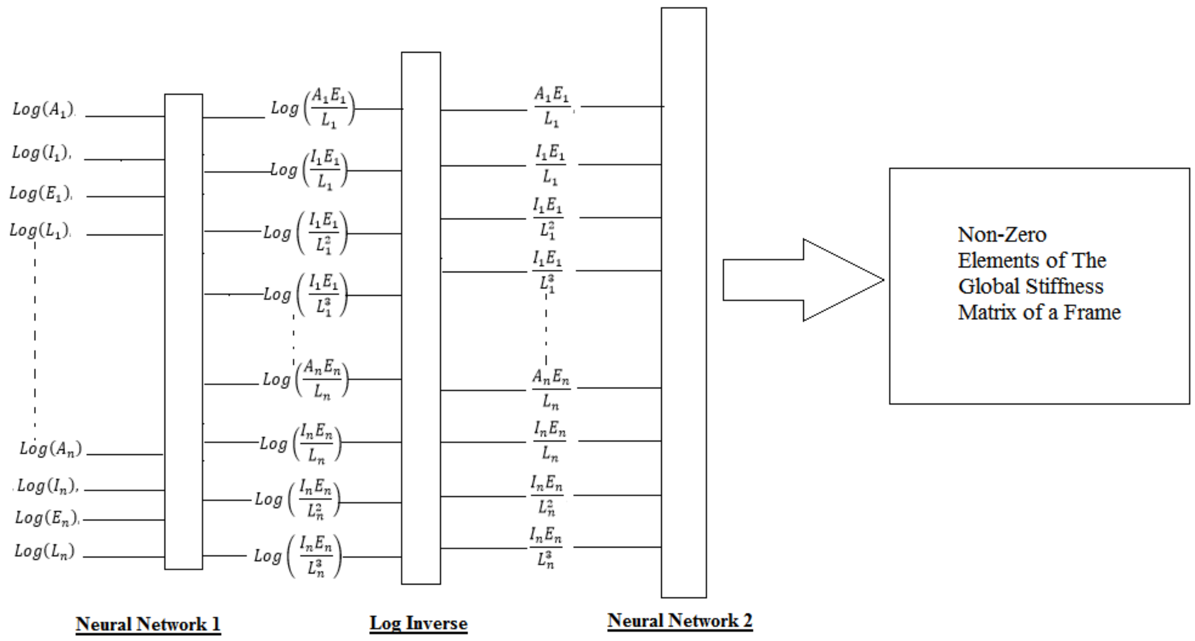
$$\log\left(\frac{A_n E_n}{L_n}\right) = \log(A_n) + \log(E_n) - \log(L_n)$$ EQ. (15)

$$\log\left(\frac{E_n I_n}{L_n}\right) = \log(E_n) + \log(I_n) - \log(L_n)$$ EQ. (16)

$$\log\left(\frac{E_n I_n}{L_n^2}\right) = \log(E_n) + \log(I_n) - 2\log(L_n)$$ EQ. (17)

$$\log\left(\frac{E_n I_n}{L_n^3}\right) = \log(E_n) + \log(I_n) - 3\log(L_n)$$ EQ. (18)

Based on the realizations presented above, the architecture in Figure 9 is suggested to predict the non-zero elements of the global stiffness matrix of a 2D frame. First, neural network 1, will be trained to replicate the linear relationships in EQ. (15), EQ. (16), EQ. (17), and EQ. (18). Then the results of neural network 1 will be used to recover the basis by applying the inverse of log. Finally, neural network 2 will be trained to find the relationship between the basis of all beam\column elements of the frame and the non-zero values in the global stiffness matrix of the frame.



$Log(A_1)$
$Log(I_1)$
$Log(E_1)$
$Log(L_1)$
$Log(A_n)$
$Log(I_n)$
$Log(E_n)$
$Log(L_n)$
$Log\left(\frac{A_1E_1}{L_1}\right)$
$Log\left(\frac{I_1E_1}{L_1}\right)$
$Log\left(\frac{I_1E_1}{L_1^2}\right)$
$Log\left(\frac{I_1E_1}{L_1^3}\right)$
$Log\left(\frac{A_nE_n}{L_n}\right)$
$Log\left(\frac{I_nE_n}{L_n}\right)$
$Log\left(\frac{I_nE_n}{L_n^2}\right)$
$Log\left(\frac{I_nE_n}{L_n^3}\right)$
$\frac{A_1E_1}{L_1}$
$\frac{I_1E_1}{L_1}$
$\frac{I_1E_1}{L_1^2}$
$\frac{I_1E_1}{L_1^3}$
$\frac{A_nE_n}{L_n}$
$\frac{I_nE_n}{L_n}$
$\frac{I_nE_n}{L_n^2}$
$\frac{I_nE_n}{L_n^3}$
Non-Zero Elements of The Global Stiffness Matrix of a Frame
Neural Network 1
Log Inverse
Neural Network 2

Figure 9 Architecture of ANN global stiffness matrix assembler

3.3. Novel Weight Matrix Method for Efficient Updating the Global Stiffness Matrices of Structures

The purpose of the Weight Matrix Method is to provide a computationally efficient way of updating the stiffness matrix of a structure that is subjected to changes in its material and geometric properties. To explain the weight matrix method, the method will be demonstrated on two examples, and then the general approach will be presented. The first example is a trivial example of a structure consisting of a 1D element, and its purpose is to introduce the main concepts of this method. The second example is on a structure made of two 1D elements, and its purpose is to show how this method is extended to structures with multiple elements.

3.3.1. Example 1: 1D Element with Four Uncoupled Degrees of Freedom

First, consider the 1D element shown in Figure 10. The 1D element has two nodes 1 and 2, and four degrees of freedom 1, 2, 3, and 4. For simplicity, assume that the stiffness matrix of this element is of the form presented in EQ.(19). For this example, the element stiffness matrix will act as the global stiffness matrix of the structure.

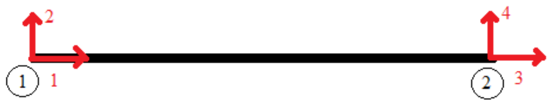


Figure 10 Line element with four degrees of freedom

$$K_{global} = K_e = \begin{bmatrix} k1 & 0 & -k1 & 0 \\ 0 & k2 & 0 & -k2 \\ -k1 & 0 & k1 & 0 \\ 0 & -k2 & 0 & k2 \end{bmatrix} \qquad \text{EQ.(19)}$$

Similar to the Conventional method, define G as the vector that holds all the nonzeros of the global stiffness matrix, define I as the vector of row indices of G in the global stiffness matrix, and define J as the vector of column indices of G in the global stiffness matrix. Using the coordinate format, K_e can be written as shown in EQ.(20).

$$G = \begin{bmatrix} k1 \\ -k1 \\ k2 \\ -k2 \\ -k1 \\ k1 \\ -k2 \\ k2 \end{bmatrix}, \quad I = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \\ 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}, \quad J = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 4 \\ 4 \end{bmatrix} \qquad \text{EQ.(20)}$$

One can notice that multiple entries of this stiffness matrix are similar. Both $k1$ and $k2$ are observed four times each in the element stiffness matrix, but the coefficients vary between negative and positive ones.

The following procedure will take advantage of such repetitions to construct the vector G.

Consider a vector b_e, which holds the elements $k1$ and $k2$ as shown in EQ.(21).

$$b_e = \begin{bmatrix} k1 \\ k2 \end{bmatrix} \qquad \text{EQ.(21)}$$

The idea of the Weight Matrix method is to define a matrix W that allows for calculating G as $G = W * b_e$. In this example, W will be defined as $W = Sparse\_Builder(v, ib, c_e)$, where c_e are coefficients (-1 or 1), v contains the row indices of the coefficients of W that range from 1 to 4, and ib contains the column indices of the coefficients of W that range from 1 to 2. The final form of W is presented in EQ.(25).

Define the vector ib that contains for each element of G, an index that points to an element of b_e such that, $|G(l)| = b_e(ib(l))$. Where l is an index that ranges between one and the number of elements of G. For this element, ib is presented in EQ.(22).

$$ib = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix} \qquad \text{EQ.(22)}$$

Also, define a vector c_e that carries the coefficients of the elements of G, such that $G(l) = c_e(l) * b_e(ib(l))$. For this element, c_e is presented in EQ.(23).

$$c_e = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \qquad \text{EQ.(23)}$$

Define a vector v, which contains for each unique I and J pair an identification number that ranges between one and the number of nonzero elements of K_e. v is shown in EQ.(24).

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{bmatrix} \qquad \text{EQ.(24)}$$

Using the Sparse_Builder function on $v, ib,$ and c_e will result in the weight matrix W shown in EQ.(25).

$$W = Sparse\_Builder(v, ib, c_e) = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \qquad \text{EQ.(25)}$$

G can easily be recovered as the dot product of W and b_e as shown in EQ.(26).

$$G = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \\ -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} k1 \\ k2 \end{bmatrix} = \begin{bmatrix} k1 \\ -k1 \\ k2 \\ -k2 \\ -k1 \\ k1 \\ -k2 \\ k2 \end{bmatrix} \qquad \text{EQ.(26)}$$

When the analysis needs to be repeated for several iterations, W needs to be constructed just once in the first iteration. For later iterations, only the vector b_e needs to be updated, and then the vector G of non-zeros can be quickly calculated using the sparse matrix- vector multiplication as in EQ.(26).

The benefit of W cannot be appreciated for the element stiffness matrix in this example because G is trivial to assemble. This method is advantageous when the structure is made up of several elements and the analysis needs to be repeated several times with different geometric or material properties. For structures with multiple elements, constructing the vector of nonzeros requires determining the influence of each element's stiffness on the stiffness of each degrees of freedom. The conventional method does so by building the local stiffness matrices, performing the proper rotation transformations, and then assembling the global stiffness matrix. The Weight Matrix method circumvents the conventional procedure by constructing a weight matrix W that allows for retrieving the nonzeros vector G of the global stiffness matrix by a simple matrix multiplication similar to that shown in EQ.(26).

3.3.2. Example 2: Structure Made of Two 1D Elements

To demonstrate the construction of W for a structure made up of more than one element, consider the system in Figure 11. The system is composed of two of the 1D elements shown in Figure 10. The first element is horizontal, but the second element is rotated 90 degrees. Each element has two degrees of freedom, and the full system has six degrees of freedom.

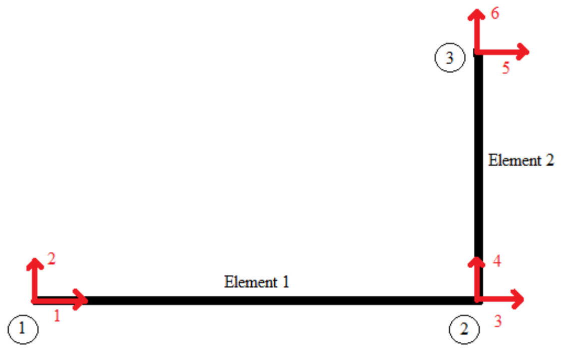


Figure 11 A structure composed of two line elements and having six degrees of freedom

Denote the local stiffness matrix of element m as K_e^m. The local stiffness matrix of the first element is shown in EQ.(27).

$$K_e^1 = \begin{bmatrix} k1^1 & 0 & -k1^1 & 0 \\ 0 & k2^1 & 0 & -k2^1 \\ -k1^1 & 0 & k1^1 & 0 \\ 0 & -k2^1 & 0 & k2^1 \end{bmatrix} \qquad \text{EQ.(27)}$$

The local stiffness matrix of the second element is shown in EQ.(28).

$$K_e^2 = \begin{bmatrix} k1^2 & 0 & -k1^2 & 0 \\ 0 & k2^2 & 0 & -k2^2 \\ -k1^2 & 0 & k1^2 & 0 \\ 0 & -k2^2 & 0 & k2^2 \end{bmatrix} \qquad \text{EQ.(28)}$$

To assemble the global stiffness matrix, the second elements local stiffness matrix needs to be rotated by 90 degrees to obtain the transformed matrix is shown in EQ.(29).

$$K_{e\,Trasform}^2 = \begin{bmatrix} k2^2 & 0 & -k2^2 & 0 \\ 0 & k1^2 & 0 & -k1^2 \\ -k2^2 & 0 & k2^2 & 0 \\ 0 & -k1^2 & 0 & k1^2 \end{bmatrix} \qquad \text{EQ.(29)}$$

Denote g_e^m as the vector of nonzeros of K_e^m, i_e^m as the row indices of the elements of g_e^m in K_e^m, and j_e^m as the column indices of the elements of g_e^m in K_e^m. The coordinate formats of K_e^1 and $K_{e\,Trasform}^2$ are as shown in EQ.(30) and EQ.(31).

$$g_e^1 = \begin{bmatrix} k1^1 \\ -k1^1 \\ k2^1 \\ -k2^1 \\ -k1^1 \\ k1^1 \\ -k2^1 \\ k2^1 \end{bmatrix}, \quad i_e^1 = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \\ 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}, \quad j_e^1 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 4 \\ 4 \end{bmatrix} \qquad \text{EQ.(30)}$$

$$g_e^2 = \begin{bmatrix} k2^2 \\ -k2^2 \\ k1^2 \\ -k1^2 \\ -k2^2 \\ k2^2 \\ -k1^2 \\ k1^2 \end{bmatrix}, \quad i_e^2 = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \\ 1 \\ 3 \\ 2 \\ 4 \end{bmatrix}, \quad j_e^2 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 4 \\ 4 \end{bmatrix} \qquad \text{EQ.(31)}$$

The assembled global stiffness matrix should be as presented in EQ.(32).

$$K_{global} = \begin{bmatrix} k1^1 & 0 & -k1^1 & 0 & 0 & 0 \\ 0 & k2^1 & 0 & -k2^1 & 0 & 0 \\ -k1^1 & 0 & k1^1 + k2^2 & 0 & -k2^2 & 0 \\ 0 & -k2^1 & 0 & k2^1 + k1^2 & 0 & -k1^2 \\ 0 & 0 & -k2^2 & 0 & -k2^2 & 0 \\ 0 & 0 & 0 & -k1^2 & 0 & k1^2 \end{bmatrix} \qquad \text{EQ.(32)}$$

The coordinate format of K_{global} is as shown in EQ.(33).

$$G = \begin{bmatrix} k1^1 \\ -k1^1 \\ k2^1 \\ -k2^1 \\ -k1^1 \\ k1^1 + k2^2 \\ -k2^2 \\ -k2^1 \\ k2^1 + k1^2 \\ -k1^2 \\ -k2^2 \\ -k2^2 \\ -k1^2 \\ k1^2 \end{bmatrix}, \quad I = \begin{bmatrix} 1 \\ 3 \\ 2 \\ 4 \\ 1 \\ 3 \\ 5 \\ 2 \\ 4 \\ 6 \\ 3 \\ 5 \\ 4 \\ 6 \end{bmatrix}, \quad J = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 3 \\ 4 \\ 4 \\ 4 \\ 5 \\ 5 \\ 6 \\ 6 \end{bmatrix} \qquad \text{EQ.(33)}$$

When the structure is made up of more than one element, I and J are not readily available. To construct I and J, i_e^m and j_e^m will be transformed as follows.

Define DOF^m as the vector that contains the degrees of freedom associated with element m. DOF^m is presented in EQ.(34) and EQ.(35) for the first and second elements

$$DOF^1 = [1\ 2\ 3\ 4] \qquad \text{EQ.(34)}$$

$$DOF^2 = [3\ 4\ 5\ 6] \qquad \text{EQ.(35)}$$

In EQ.(36) and EQ.(37), define i_{dof}^m and j_{dof}^m as the indices of the elements of g_e^m in the global stiffness matrix K_{global}.

$$i_{dof}^m(l) = DOF^m\big(i^m\,(l)\big) \qquad \text{EQ.(36)}$$

$$j_{dof}^m(l) = DOF^m\big(j^m\,(l)\big) \qquad \text{EQ.(37)}$$

Where l is an index that ranges from one to the length of g^m. For the elements 1 and 2, $i_{dof}^1, j_{dof}^1, i_{dof}^2$, and j_{dof}^2 are as shown in EQ.(38) and EQ.(39).

$$i_{dof}^1 = \begin{bmatrix}1\\3\\2\\4\\1\\3\\2\\4\end{bmatrix}, \qquad j_{dof}^1 = \begin{bmatrix}1\\1\\2\\2\\3\\3\\4\\4\end{bmatrix} \qquad \text{EQ.(38)}$$

$$i_{dof}^{2} = \begin{bmatrix}3\\5\\4\\6\\3\\5\\4\\6\end{bmatrix}, \qquad j_{dof}^{2} = \begin{bmatrix}3\\3\\4\\4\\5\\5\\6\\6\end{bmatrix}$$ EQ.(39)

In EQ.(40), Define I_{con} as the vector where all i_{dof}^{m} are vertically concatenated and J_{con} as the vector where all j_{dof}^{m} are vertically concatenated.

$$I_{con} = \begin{bmatrix}1\\3\\2\\4\\1\\3\\2\\4\\3\\5\\4\\6\\3\\5\\4\\6\end{bmatrix}, \qquad J_{con} = \begin{bmatrix}1\\1\\2\\2\\3\\3\\4\\4\\3\\3\\4\\4\\5\\5\\6\\6\end{bmatrix}$$ EQ.(40)

Already, I_{con} and J_{con} resemble I and J in EQ.(33), but some I_{con} - J_{con} pairs are repetitions. [3,3] and [4,4] are repeated. I and J can be created by removing the repeated elements. Depending on the connectivity and the numbering of the elements in a given problems, the order of the elements of I and J calculated in this step may not be identical to that in EQ.(33). This will be resolved in later steps by sorting.

Define b_e^m as the vector that holds the repeated elements in the stiffness matrix of element m. b_e^1 and b_e^2 are presented in EQ.(41) and EQ.(42).

$$b_e^1 = \begin{bmatrix}k1^1\\k2^1\end{bmatrix}$$ EQ.(41)

$$b_e^2 = \begin{bmatrix} k1^2 \\ k2^2 \end{bmatrix} \quad \text{EQ.(42)}$$

In this example, W will be defined as $W = Sparse\_Builder(v, iB_{con}, c_{con})$. Unlike the previous trivial example, $iB_{con}, v,$ and c_{con} concatenate information from more than one element. The following procedure demonstrates how $iB_{con}, v,$ and c_{con} are determined.

Define the vector ib^m that contains indices that points to elements of b_e^m such that, $|g^m(l)| = b_e^m(ib(l))$. Where l is an index that ranges between one and the number of elements of g_e^m. ib^1 and ib^2 are presented in EQ.(43) and EQ.(44).

$$ib^1 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix} \quad \text{EQ.(43)}$$

$$ib^2 = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \end{bmatrix} \quad \text{EQ.(44)}$$

For each element m, ib^mpoints to elements of b_e^m. It would be easier to concatenate all ib^m into one vector B_e and define vectors iB^m that point to the proper elements of B_e.

In EQ.(45), define B_e as the vertical concatenation of all b_e^m vectors

$$B_e = \begin{bmatrix} b_e^1 \\ b_e^2 \end{bmatrix} = \begin{bmatrix} k1^1 \\ k2^1 \\ k1^2 \\ k2^2 \end{bmatrix} \quad \text{EQ.(45)}$$

Denote $length_{b_e}$ as the length of b_e^m. Then, the iB^m vectors, which point to the elements of B_e, can be defined as shown in EQ.(46).

$$iB^m = ib^m + length_{b_e} * (m-1) \qquad \text{EQ.(46)}$$

The horizontal element has an identification number $m = 1$, so iB^1 is calculated as shown in EQ.(47).

$$iB^1 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix} + 2(1-1) = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix} \qquad \text{EQ.(47)}$$

The vertical element has an identification number $m = 2$, so iB^2 is calculated as shown in EQ.(48).

$$iB^2 = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \end{bmatrix} + 2(2-1) = \begin{bmatrix} 4 \\ 4 \\ 3 \\ 3 \\ 4 \\ 4 \\ 3 \\ 3 \end{bmatrix} \qquad \text{EQ.(48)}$$

Then, define iB_{con} as the vertical concatenation of all iB^m as shown in

$$iB_{con} = \begin{bmatrix} iB^1 \\ iB^2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \\ 2 \\ 2 \\ 4 \\ 4 \\ 3 \\ 3 \\ 4 \\ 4 \\ 3 \\ 3 \end{bmatrix} \quad \text{EQ.(49)}$$

In EQ.(50), define a vector c_e^m that carries the coefficients of the elements of g_e^m, such that $g(l) = c_e^m(l) * \ b_e^m(ib^m(l))$.

$$c_e^1 = c_e^2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \quad \text{EQ.(50)}$$

Then, define c_{con} as the vertical concatenation of all c_e^m as shown in

$$c_{con} = \begin{bmatrix} c_e^1 \\ c_e^2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix} \quad \text{EQ.(51)}$$

Define a vector v, which contains for each unique I_{con} - J_{con} pair an identification number as shown in EQ.(52). Give the identification numbers in ascending order. When a repeated I_{con} - J_{con} pair is encountered repeat the identification number as well.

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 6 \\ 9 \\ 8 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \end{bmatrix} \qquad \text{EQ.(52)}$$

Using the Sparse_Builder function on v, iB_{con}, and c_{con} will result in the weight matrix W shown in EQ.(53).

$$W = Sparse\_Builder(v, iB_{con}, c_{con}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{EQ.(53)}$$

G can easily be recovered as the dot product of W and B_e as shown in EQ.(54).

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} k1^1 \\ k2^1 \\ k1^2 \\ k2^2 \end{bmatrix} = \begin{bmatrix} k1^1 \\ -k1^1 \\ k2^1 \\ -k2^1 \\ -k1^1 \\ k1^1 + k2^2 \\ -k2^2 \\ -k2^1 \\ k2^1 + k1^2 \\ -k1^2 \\ -k2^2 \\ -k2^2 \\ -k1^2 \\ k1^2 \end{bmatrix} \qquad \text{EQ.(54)}$$

Then the global stiffness matrix can be built as $K_{Global} = Sparse\_Builder(I, J, G)$. In order to improve the efficiency of the Sparse_Builder function I and J need to be properly sorted. Sort the rows of W, I, and J in an ascending order of the values of J, and then sort the rows of W, I, and J that correspond to the same value of J in an ascending order of I.

Whenever the global stiffness matrix needs to be updated, only B_e needs to be updated.

3.3.3. General Weight Matrix Procedure

The following steps are needed to implement the Weight Matrix Procedure

1. Define the vectors b_e^m that contain the repeated entries of the transformed stiffness matrices of each element m.
2. Define $B_e = \begin{bmatrix} b_e^1 \\ \vdots \\ b_e^m \end{bmatrix}$.

3. Define ib^m that contains indices that point to elements of b_e^m such that, $|g^m(l)| = b_e^m(ib(l))$, where g^m is the vector that holds the nonzero entries of the transformed stiffness matrices of each element m.
4. Define iB^m as the vector that points to the elements of b_e^m inside B_e.
5. Define $iB_{con} = \begin{bmatrix} iB^1 \\ \vdots \\ iB^m \end{bmatrix}$.
6. Define i_e^m as a vector that contains the row indices of the nonzero elements of the local stiffness matrix of element m, and define and j_e^m as the vector that contains the column indices of the nonzero elements of the local stiffness matrix of element m.
7. Use the connectivity matrix to determine the degrees of freedom DOF^m associated with each element.
8. Use the element's F^m, i_e^m, and j_e^m to construct i_{dof}^m and j_{dof}^m as shown in EQ.(36) and EQ.(37). i_{dof}^m and j_{dof}^m point to the location of the nonzero elements of the local stiffness matrix in the global stiffness matrix.
9. Use i_{dof}^m and j_{dof}^m to construct $I_{con} = \begin{bmatrix} i_{dof}^1 \\ \vdots \\ i_{dof}^m \end{bmatrix}$ and $J_{con} = \begin{bmatrix} j_{dof}^1 \\ \vdots \\ j_{dof}^m \end{bmatrix}$.
10. Define I and J as the row and column indices of the nonzero elements of the global stiffness matrix. Obtain unsorted I and J by removing the repeated $I_{con} - J_{con}$ pairs. I and J will be sorted in the final step.
11. Define a vector v, which contains for each unique I_{con} - J_{con} pair an identification number. Give the identification numbers in ascending order. When a repeated I_{con} - J_{con} pair is encountered repeat the identification number as well.

12. Define a vector c_e^m that carries the coefficients of the elements of g_e^m, such that $g_e^m(l) = c_e^m(l) * \ b_e^m(ib^m(l))$.
13. Define $c_{con} = \begin{bmatrix} c_e^1 \\ \vdots \\ c_e^m \end{bmatrix}$.
14. Use , iB_{con}, c_{con}, and the Sparse_Builder function to determine the weight matrix $W = Sparse\_Builder(v, iB_{con}, c_{con})$.
15. Sort the rows of W, I, and J in an ascending order of the values of J, and then for each value of J, sort the rows of W, I, and J in an ascending order of I.
16. The nonzero elements of the global stiffness matrix can now be determined as $G = \ W * B_e$, and the global stiffness matrix can be determined as $K_{Global} = Sparse\_Builder(I, J, G)$.
17. When the global stiffness matrix needs to be updated, only B_e needs to be updated.

3.3.4. Implementation of the Weight Matrix Method for Bernoulli Beam/Column Elements

In order to implement the Weight Matrix method, the stiffness matrices of the Bernoulli beam/column elements need to be decomposed.

The stiffness matrix of a Bernoulli beam element is as shown in EQ.(55).

$$K_{Beam}=\begin{bmatrix}\frac{AE}{L} & 0 & 0 & -\frac{AE}{L} & 0 & 0\\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2}\\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L}\\ -\frac{AE}{L} & 0 & 0 & \frac{AE}{L} & 0 & 0\\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2}\\ 0 & \frac{6EI}{L^2} & \frac{2EI}{L} & 0 & \frac{-6EI}{L^2} & \frac{4EI}{L}\end{bmatrix} \qquad \text{EQ.(55)}$$

The row and column indices, $(i_e)_{Beam}$ and $(j_e)_{Beam}$, of the nonzero elements of K_{Beam} are as shown in EQ.(56).

$$(i_e)_{Beam}=\begin{bmatrix}1\\4\\2\\3\\5\\6\\2\\3\\5\\6\\1\\4\\2\\3\\5\\6\\2\\3\\5\\6\end{bmatrix},\qquad (j_e)_{Beam}=\begin{bmatrix}1\\1\\2\\2\\2\\2\\3\\3\\3\\3\\4\\4\\5\\5\\5\\5\\6\\6\\6\\6\end{bmatrix} \qquad \text{EQ.(56)}$$

K_{Beam}'s vector of repeated elements is $(b_e)_{Beam}$ as shown in EQ.(57).

$$(b_e)_{Beam} = \begin{bmatrix} \frac{4EI}{L} \\ \frac{6EI}{L^2} \\ \frac{12EI}{L^3} \\ \frac{AE}{L} \end{bmatrix} \qquad \text{EQ.(57)}$$

The vector of indices $(ib_e)_{Beam}$ pointing to elements of $(b_e)_{Beam}$ is defined in EQ.(58), and the vector of coefficients $(c_e)_{Beam}$ is defined in EQ.(59).

$$(ib_e)_{Beam} = \begin{bmatrix} 4 \\ 4 \\ 3 \\ 2 \\ 3 \\ 2 \\ 2 \\ 1 \\ 2 \\ 1 \\ 4 \\ 4 \\ 3 \\ 2 \\ 3 \\ 2 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} \qquad \text{EQ.(58)}$$

$$(c_e)_{Beam} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ \frac{1}{2} \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ -1 \\ 1 \\ \frac{1}{2} \\ -1 \\ 1 \end{bmatrix} \qquad \text{EQ.(59)}$$

The Bernoulli Column is obtained by rotating the Bernoulli Beam 90 degrees. The stiffness matrix of a Bernoulli Column element is shown in EQ.(60).

$$K_{Column} = \begin{bmatrix} \frac{12EI}{L^3} & 0 & -\frac{6EI}{L^2} & -\frac{12EI}{L^3} & 0 & -\frac{6EI}{L^2} \\ 0 & \frac{AE}{L} & 0 & 0 & -\frac{AE}{L} & 0 \\ -\frac{6EI}{L^2} & 0 & \frac{4EI}{L} & \frac{6EI}{L^2} & 0 & \frac{2EI}{L} \\ -\frac{12EI}{L^3} & 0 & \frac{6EI}{L^2} & \frac{12EI}{L^3} & 0 & \frac{6EI}{L^2} \\ 0 & -\frac{AE}{L} & 0 & 0 & \frac{AE}{L} & 0 \\ -\frac{6EI}{L^2} & 0 & \frac{2EI}{L} & \frac{6EI}{L^2} & 0 & \frac{4EI}{L} \end{bmatrix} \qquad \text{EQ.(60)}$$

The row and column indices, $(i_e)_{Column}$ and $(j_e)_{Column}$, of the nonzero elements of K_{Column} are as shown in EQ.(61).

$$(i_e)_{Column} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 6 \\ 2 \\ 5 \\ 1 \\ 3 \\ 4 \\ 6 \\ 1 \\ 3 \\ 4 \\ 6 \\ 2 \\ 5 \\ 1 \\ 3 \\ 4 \\ 6 \end{bmatrix}, \qquad (j_e)_{Column} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 3 \\ 3 \\ 4 \\ 4 \\ 4 \\ 4 \\ 5 \\ 5 \\ 6 \\ 6 \\ 6 \\ 6 \end{bmatrix} \qquad \text{EQ.(61)}$$

K_{Column}'s vector of repeated elements is $(b_e)_{Column}$ as shown in EQ.(62).

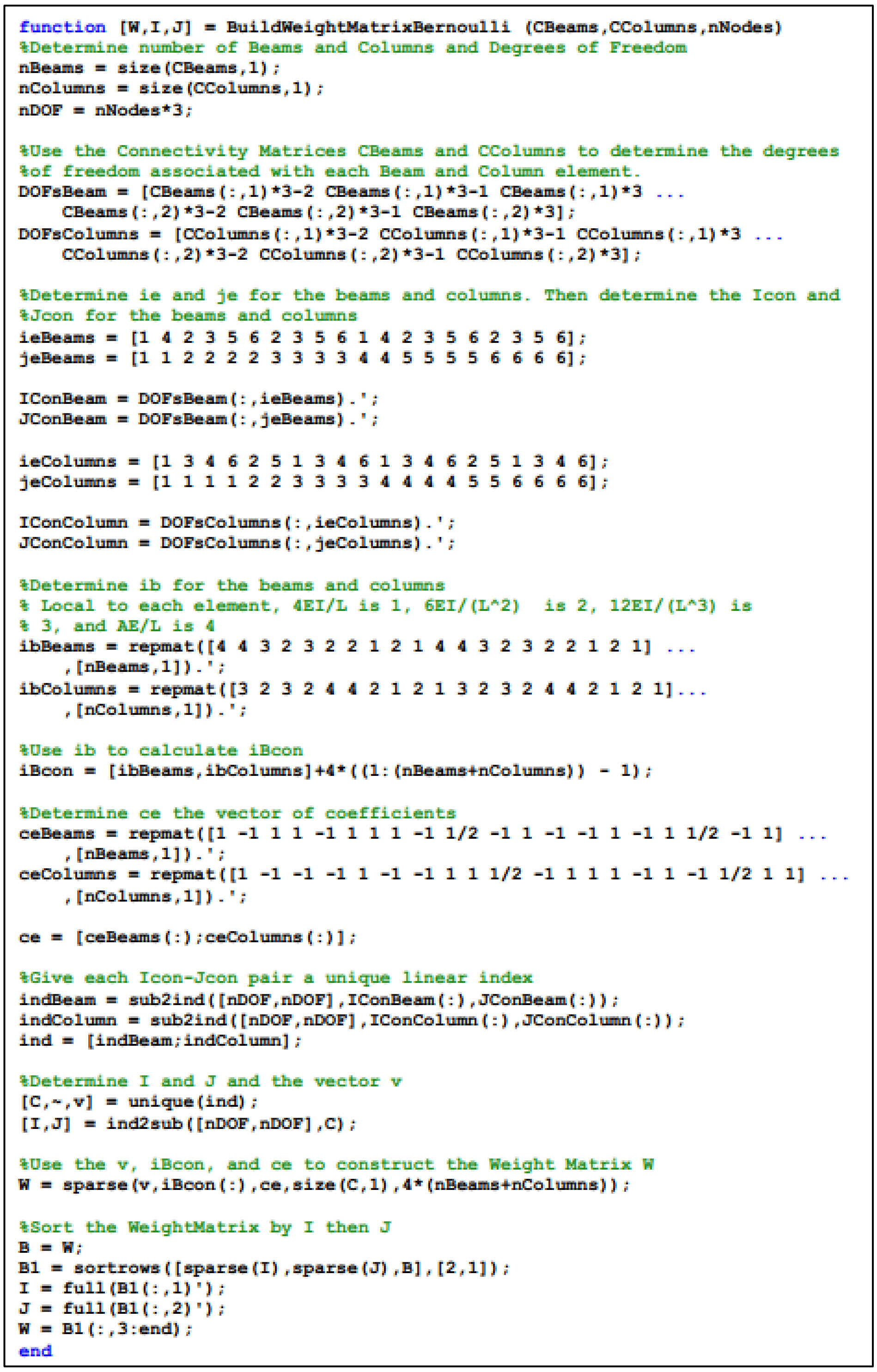
$$(b_e)_{Column} = \begin{bmatrix} \frac{4EI}{L} \\ \frac{6EI}{L^2} \\ \frac{12EI}{L} \\ \frac{AE}{L} \end{bmatrix} \qquad \text{EQ.(62)}$$

The vector of indices $(ib_e)_{Column} =$ pointing to elements of $(b_e)_{Column}$ is defined in EQ.(63), and the vector of coefficients $(c_e)_{Column}$ is defined in EQ.(64).

$$(ib_e)_{Column} = \begin{bmatrix} 3 \\ 2 \\ 3 \\ 2 \\ 4 \\ 4 \\ 2 \\ 1 \\ 2 \\ 1 \\ 3 \\ 2 \\ 3 \\ 2 \\ 4 \\ 4 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} \qquad \text{EQ.(63)}$$

$$(c_e)_{Column} = \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ \frac{1}{2} \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \\ \frac{1}{2} \\ 1 \\ 1 \end{bmatrix} \qquad \text{EQ.(64)}$$

Figure 12 shows a MATLAB code that constructs the weight matrix of a planar frame structure made of Bernoulli beam and column elements. $nNodes$ is the number of nodes, and $CBeams$ and $CColumns$ are the connectivity matrices of the beams and columns respectively. The connectivity matrix of the beams has a dimensions of $nBeams \times 2$, and that of the columns has dimensions of $nColumns \times 2$, where $nBeams$ and $nColumns$ are the number of beam and column elements respectively. The first column of the connectivity matrix stores the node at the beginning of the element, and the second column stores the node at the end of the element. This code assumes that a beam's first node is at its left extremity, and a column's first node is at its bottom extremity.



```matlab
function [W,I,J] = BuildWeightMatrixBernoulli (CBeams,CColumns,nNodes)
%Determine number of Beams and Columns and Degrees of Freedom
nBeams = size(CBeams,1);
nColumns = size(CColumns,1);
nDOF = nNodes*3;

%Use the Connectivity Matrices CBeams and CColumns to determine the degrees
%of freedom associated with each Beam and Column element.
DOFsBeam = [CBeams(:,1)*3-2 CBeams(:,1)*3-1 CBeams(:,1)*3 ...
    CBeams(:,2)*3-2 CBeams(:,2)*3-1 CBeams(:,2)*3];
DOFsColumns = [CColumns(:,1)*3-2 CColumns(:,1)*3-1 CColumns(:,1)*3 ...
    CColumns(:,2)*3-2 CColumns(:,2)*3-1 CColumns(:,2)*3];

%Determine ie and je for the beams and columns. Then determine the Icon and
%Jcon for the beams and columns
ieBeams = [1 4 2 3 5 6 2 3 5 6 1 4 2 3 5 6 2 3 5 6];
jeBeams = [1 1 2 2 2 2 3 3 3 3 4 4 5 5 5 5 6 6 6 6];

IConBeam = DOFsBeam(:,ieBeams).';
JConBeam = DOFsBeam(:,jeBeams).';

ieColumns = [1 3 4 6 2 5 1 3 4 6 1 3 4 6 2 5 1 3 4 6];
jeColumns = [1 1 1 1 2 2 3 3 3 3 4 4 4 4 5 5 6 6 6 6];

IConColumn = DOFsColumns(:,ieColumns).';
JConColumn = DOFsColumns(:,jeColumns).';

%Determine ib for the beams and columns
% Local to each element, 4EI/L is 1, 6EI/(L^2)  is 2, 12EI/(L^3) is
% 3, and AE/L is 4
ibBeams = repmat([4 4 3 2 3 2 2 1 2 1 4 4 3 2 3 2 2 1 2 1] ...
    ,[nBeams,1]).';
ibColumns = repmat([3 2 3 2 4 4 2 1 2 1 3 2 3 2 4 4 2 1 2 1]...
    ,[nColumns,1]).';

%Use ib to calculate iBcon
iBcon = [ibBeams,ibColumns]+4*((1:(nBeams+nColumns)) - 1);

%Determine ce the vector of coefficients
ceBeams = repmat([1 -1 1 1 -1 1 1 1 -1 1/2 -1 1 -1 -1 1 -1 1 1/2 -1 1] ...
    ,[nBeams,1]).';
ceColumns = repmat([1 -1 -1 -1 1 -1 -1 1 1 1/2 -1 1 1 1 -1 1 -1 1/2 1 1] ...
    ,[nColumns,1]).';

ce = [ceBeams(:);ceColumns(:)];

%Give each Icon-Jcon pair a unique linear index
indBeam = sub2ind([nDOF,nDOF],IConBeam(:),JConBeam(:));
indColumn = sub2ind([nDOF,nDOF],IConColumn(:),JConColumn(:));
ind = [indBeam;indColumn];

%Determine I and J and the vector v
[C,~,v] = unique(ind);
[I,J] = ind2sub([nDOF,nDOF],C);

%Use the v, iBcon, and ce to construct the Weight Matrix W
W = sparse(v,iBcon(:),ce,size(C,1),4*(nBeams+nColumns));

%Sort the WeightMatrix by I then J
B = W;
B1 = sortrows([sparse(I),sparse(J),B],[2,1]);
I = full(B1(:,1)');
J = full(B1(:,2)');
W = B1(:,3:end);
end
```

Figure 12 A MATLAB code for building the weight matrix of a plane frame made of Bernoulli elements

3.4. Design Dead, Live, and Seismic Loads and Load Combinations

The frames are loaded with a dead load of 5 Ton/m per beam span, live load of 1 Ton/m, and a lateral earthquake load. The earthquake load was modeled as an equivalent static load in accordance with ASCE 7-10. The values of the spectral accelerations for the design level earthquakes used in this study are in accordance with the suggestions of the Lebanese Order of Engineers. The spectral acceleration at a period of 0.2 seconds was taken to be 1.2g, and the spectral acceleration at a period of 1 second was taken to be 0.4g. The load combinations considered in this study are as shown in Eq. (65), where D is the dead load, L is the live load, and E is the earthquake load.

$$\begin{gathered} 1.4D \\ 1.2\,D + 1.6L \\ 1.2D + L \pm E \\ 0.9D \pm E \end{gathered} \qquad \text{Eq. (65)}$$

3.5. Integrating the Genetic Algorithm into Structural Design

3.5.1. Design Variables

The design variables are the values that will be changed by the optimization algorithm. In this section, the design variables associated with the beams and columns of the frame structure will be introduced.

3.5.1.1. Beam Design Variables

This section elaborates on the design variables associated with the beams of the frame structure. The design variables associated with the beams of the frame structure are presented in Figure 13 and Table 1. In Table 1, N_b is the number of beams, N_s is the number of stories of the frame, and N_j is the number of joints of the frame. The values in Table 1 are used for demonstration purposes, but the designer is free to choose the ranges they see fit for their design.

The following assumptions were made about the geometry and dimensions of the beams. All beams used in this research are rectangular beams, beams in the same story will share a common width. The width of the beams in a single story is calculated as follows. First, the minimum width of the columns supporting the beam is determined. Next, this width is multiplied by the beam width to minimum column width ratio (R_w^{beam}) presented in Table 1. This ensures that the beams are always at most as wide as the columns that support them. Beams that are wider than their supports were not considered in this research. Then, the beam width is multiplied by the beam height to width ratio (R_h^{beam}) to determine the beam's height. Both the beam width and column height are rounded to the nearest 5 cm.

Moreover, the following assumptions were made about the reinforcement of the beams. Spliced reinforcement bars will be provided to satisfy the structural integrity requirements of the ACI 318 and to support the standard hooks, and additional cutoff reinforcement bars will be inserted to increase the moment capacity at critical sections. Due to the stiffness of the joint and the effect of lateral loads, the moments will be greater at the ends of the beam than at the mid-spans. Therefore, the primary reinforcement is spliced near the mid-spans of the beam. The structural integrity requirements of the ACI-318 code require that the greater of two reinforcement bars or one quarter of the maximum reinforcement area be continuous or spliced unless terminated at the joint with a standard hook.. As for the cutoff reinforcement, it consists of rebars that are allowed to terminate after a certain cutoff length from their starting location. The purpose a cutoff bar is to provide additional flexural strength for a certain location on the beam. Using cutoff reinforcement bars reduces the cost since they are allowed to be terminated when they are no longer needed. In this research, the cutoff lengths of the bars at the extremities of the beams are considered design variables. The cutoff length of the mid-span reinforcement, if needed, is determined as the difference between the unsupported length of a beam and the cutoff lengths of the reinforcement at the extremities of a beam.

As for the seismic hooks, their number will be determined as follows. The ACI 318 code requires that the continuous reinforcement be enclosed with the seismic hoops and hook ties, so the number of continuous bars will be used to determine the number of hook ties. In addition, ACI 318 dictates that the seismic hoops are not allowed to be laterally spaced more than 36 cm. Therefore, the minimum number of hook ties and continuous bars will be affected by the width of the beam.

Table 1 Beam design variables

Name of Variable	Number of Variables	Range of Variable	Range Justification
Beam Height to Width Ratio (R_h^{beam})	N_b	1.5 to 2	Typical height to width ratio ranges
Beam Width to Minimum Column Width Ratio (R_w^{beam})	N_s	0.5 to 1	Only beams narrower than their supports are considered in this research

Name of Variable	Number of Variables	Range of Variable	Range Justification
Number of Primary Reinforcement (Top and Bottom)	N_s	2 to 9 bars	ACI requires minimum of 2 bars. 9 bars maximum to avoid congestion.
Diameter of Top Primary Reinforcement	N_j	10mm-12mm-14mm-16mm-20mm-25mm-32mm	Available in market.
Diameter of Bottom Primary Reinforcement	N_j	10mm-12mm-14mm-16mm-20mm-25mm-32mm	Available in market.
Number of Top Cutoff Reinforcement at Mid-Span	N_b	0 to 9 bars	Max 9 bars to avoid congestion.
Diameter of Top Cutoff Reinforcement at Mid-Span	N_b	10mm-12mm-14mm-16mm-20mm-25mm-32mm	Available in market.
Number of Bottom Cutoff Reinforcement at Mid-Span	N_b	0 to 9 bars	Max 9 bars to avoid congestion.
Diameter of Bottom Cutoff Reinforcement at Mid-Span	N_b	10mm-12mm-14mm-16mm-20mm-25mm-32mm	Available in market.
Number of Top Cutoff Reinforcement at Joints	N_j	0 to 9 bars	Max 9 bars to avoid congestion.
Diameter of Top Cutoff Reinforcement at Joints	N_j	10mm-12mm-14mm-16mm-20mm-25mm-32mm	Available in market.
Number of Left Bottom Cutoff Bars	N_b	0 to 9 bars	Max 9 bars to avoid congestion.
Diameter of Left Bottom Cutoff Bars	N_b	10mm-12mm-14mm-16mm-20mm-25mm-32mm	Available in market.
Number of Right Bottom Cutoff Bars	N_b	0 to 9 bars	Max 9 bars to avoid congestion.
Diameter of Right Bottom Cutoff Bars	N_b	10mm-12mm-14mm-16mm-20mm-25mm-32mm	Available in market.
Top Left Bars Cutoff Length	N_b	0 to 0.35 of the Beam’s Length	ACI recommendation
Top Right Bars Cutoff Length	N_b	0 to 0.35 of the	ACI

Name of Variable	Number of Variables	Range of Variable	Range Justification
		Beam’s Length	recommendation
Bottom Left Bars Cutoff Length	N_b	0 to 0.35 of the Beam’s Length	ACI recommendation
Bottom Right Bars Cutoff Length	N_b	0 to 0.35 of the Beam’s Length	ACI recommendation

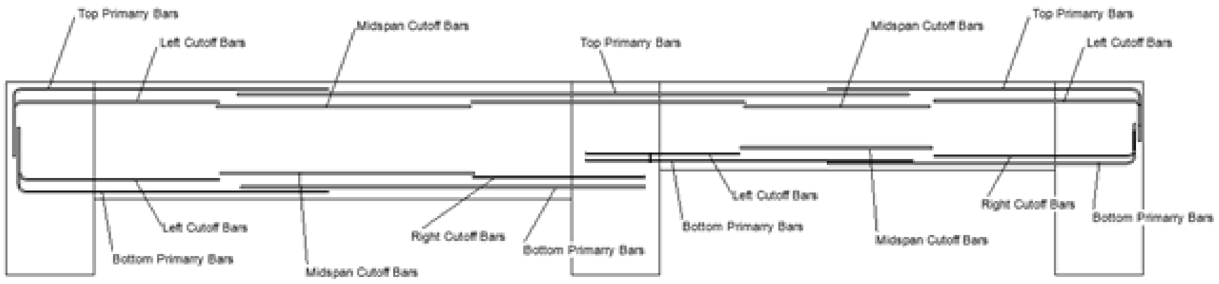


Figure 13 Reinforcement configuration in Beams

3.5.1.2. Column Design Variables

This section elaborates on the columns’ design variables. Table 2 and Figure 14 present the design variables associated with the columns. N_c denotes the number of columns. The columns are assumed to be rectangular, symmetric on both axes, and all longitudinal bars are constrained with either a seismic hoop or hook tie. A column’s width is restricted between 40cm and 80cm, and the column’s height is obtained by multiplying the column width with a variable that stores the column height to width ratio. Then, the column width and height are rounded to the nearest 5 cm. All longitudinal reinforcement bars are assumed to have the same diameter. The values in Table 2 are used for demonstration purposes, but the designer is free to choose the ranges they see fit for their design.

Table 2 Column design variables

Name of Variable	Number of Variables	Range of Variable	Range Justification
Column Width	N_c	40cm to 80cm	40cm minimum is chosen to avoid congestion of reinforcement at joints
Column Height to Width Ratio	N_c	1 to 2	Typical Height to width Ratios
Number of Primary Reinforcement	N_c	2 to 9 bars Top and Bottom	Max 9 bars to avoid congestion
Number of Secondary Reinforcement	N_c	0 to 20 Rows of Reinforcement	Max 20 bars to avoid congestion
Diameter of Reinforcement	N_c	10mm-12mm-14mm-16mm-20mm-25mm-32mm	Available on market.

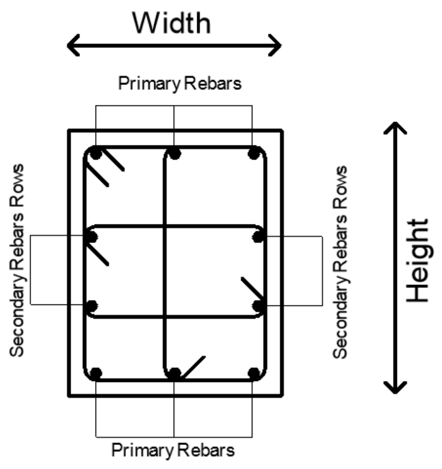


Figure 14 Configuration of reinforcement in a typical column section

3.5.2. Requirements and Constraints

3.5.2.1. Beam Moment Constraints

First, the ACI 318 code requires that the applied factored ultimate moment M_u be less than the beam's moment capacity φM_n along the entire span of beam. Since the beams are part of a plane moment resisting frame that is required to resist lateral forces in both positive and negative x-directions, each section is expected to have a positive moment capacity φM_n^+ and a negative moment capacity φM_n^-. Therefore, the ultimate moment M_u should have an algebraic value greater than φM_n^- and less than φM_n^+. The constraints in EQ. (66) and EQ. (67) are used to enforce this requirement. In this research, this constraint is checked at ten equally spaced points along the span of each beam.

$$g = \frac{(M_u - \varphi M_n^+)}{|\varphi M_n^+|} \qquad \text{EQ. (66)}$$

$$g = \frac{(\varphi M_n^- - M_u)}{|\varphi M_n^-|} \qquad \text{EQ. (67)}$$

Furthermore, beams that are part of a special moment resisting frame are required by the ACI 318-14 code to have the positive moment capacity near the joint $(\varphi M_n^+)^{joint}$ at least equal to half the absolute value of the negative moment capacity near the joint $|\varphi M_n^-|^{joint}$. Also, the minimum absolute value of the moment capacity $|\varphi M_n|^{min}$ along the span of the beam must be at least one quarter of the maximum absolute value of the moment capacity $|\varphi M_n|^{max}$. Those requirements can be satisfied using the constraints shown in EQ. (68) and EQ. (69).

$$g = \frac{\left(\frac{1}{2} - \frac{(\varphi M_n^+)^{joint}}{|\varphi M_n^-|^{joint}}\right)}{\frac{1}{2}} \qquad \text{EQ. (68)}$$

$$g = \frac{\left(\frac{1}{4} - \frac{|\varphi M_n|^{min}}{|\varphi M_n|^{max}}\right)}{\frac{1}{4}} \qquad \text{EQ. (69)}$$

3.5.2.2. Beam Reinforcement Ratio Requirements

The ACI code permits beams that are part of a special moment resisting frame to have a reinforcement ratio ρ as high as 2.5%, and the minimum flexural requirements entail having a reinforcement ratio of at least 0.33%. The constraints in EQ. (70) and EQ. (71) are used to enforce these requirements.

$$g = \frac{(0.33\% - \rho)}{0.33\%} \qquad \text{EQ. (70)}$$

$$g = \frac{(\rho - 2.5\%)}{2.5\%} \qquad \text{EQ. (71)}$$

3.5.2.3. Beam Hoop Spacing Constraints

The hoop spacing along the span of the beam is controlled by both confinement requirements and shear demand. For confinement, the ACI 318 code requires satisfying the condition in EQ. (72).

$$s_c = \min\begin{cases} \dfrac{b_b}{4} \\ 6d_b \\ 6\ in \end{cases} \qquad \text{EQ. (72)}$$

Where b_b is the beam's width and d_b is the smallest diameter of the longitudinal reinforcement.

As for shear demand, the following capacity design procedure is followed to determine the spacing. The shear due to developing the probable moments at the faces of the supports is calculated using the free diagram in Figure 15. V_{e1} and V_{e2} are the shears developed at the ends of the beam, M_{pr1} and M_{pr_2} are the probable moment capacities at the ends of the beam, w_u is the distributed load, and l_n is the length of the beam. According to ACI 318, the probable moment strength is determined using section analysis while assuming that the concrete develops its ultimate strength f_c' and the tension reinforcement develops a tensile stress $f_s = 1.25\ f_y$. Using the free diagram and the equilibrium equations, the shears at the ends of the beam can be expressed as shown in EQ. (73) and EQ. (74).

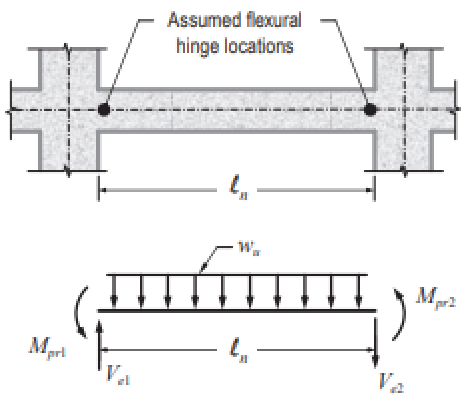


Figure 15 Free body diagram used to calculate the shear induced when the probable moment capacities are reached (Moehle, 2014)

$$V_{e1} = \frac{M_{pr1} + M_{pr_2}}{l_n} + \frac{w_u l_n}{2} \qquad \text{EQ. (73)}$$

$$V_{e2} = \frac{M_{pr1} + M_{pr_2}}{l_n} - \frac{w_u l_n}{2} \qquad \text{EQ. (74)}$$

V_{e1} and V_{e2} are determined for both clockwise and counterclockwise probable moment capacities, and then the beam is designed for the absolute maximum probable shear $(V_e)^{max}$. EQ. (75) shows the spacing requirement due to shear. A_v is the area of transverse reinforcement. The number of legs of transverse reinforcement is equal to the number of primary reinforcement bars, and the diameter is provided by the user. V_c is the shear strength of concrete and it is calculated per the ACI 318-14 requirements.

$$s_v = \frac{A_v f_y d}{\frac{(V_e)^{max}}{\varphi} - V_c} \qquad \text{EQ. (75)}$$

3.5.2.4. <u>Cutoff and Development Length</u>

ACI 318 sets the following requirements for the development length of the longitudinal reinforcement use in special moment resisting frames. When the bar needs to be terminated by a standard hook, EQ. (76) is used to determine the development length l_{dh}.

$$l_{dh} = \frac{f_y}{65\sqrt{f_c'}}\ d_b\ (psi) \qquad \text{EQ. (76)}$$

The development length in tension for straight bars is dependent on the location on the bar. In general, bars near the top face of the beam require a development length of $l_{ds} = 3.25l_{dh}$, and bars near the bottom face require a development length of $l_{ds} = 2.5l_{dh}$.

Near the face of the supports, the minimum cutoff lengths $(l_{cut})^{min}$ are determined as shown in EQ. (77)

$$(l_{cut})^{min} = l_d + max\begin{cases} d \\ 12d_b \end{cases}$$
EQ. (77)

Where l_d is the appropriate development length needed to develop the bars, and d is the depth of the tensile reinforcement being developed.

Then, the cutoff length of the reinforcement at the face of the left support $(l_{cut})^{left}$ and the cutoff length of the reinforcement at the face of the right support $(l_{cut})^{right}$ are determined as the maximum of the values of $(l_{cut})^{min}$ and the cutoff length variables shown in Table 1 Beam design variables.

Finally, the length of the cutoff reinforcement at mid-span is determined as shown in EQ. (78).

$$(l_{cut})^{middle} = l_n - (l_{cut})^{left} - (l_{cut})^{right}$$
EQ. (78)

$(l_{cut})^{middle}$ must be greater than twice the development length. This can be satisfied by the constraint in EQ. (79).

$$g = \frac{\left(2l_d - (l_{cut})^{middle}\right)}{2l_d}$$
EQ. (79)

3.5.2.5. Splice Length

The ACI 420 requires that the spliced reinforcement bars of beams subjected to cyclic loading have a minimum splice length l_{sp} shown in EQ. (80).

$$l_{sp} \geq \frac{1860}{\sqrt{f_c'}} \geq 20\ d_b\ (psi) \qquad \text{EQ. (80)}$$

In this research, the splices between the primary longitudinal reinforcements are assumed to be near the beams' mid-spans. Since this equation results in a maximum splice length of 1m, and all beams are longer than 1 m, the splice length is always feasible. The splice length is only used for estimating the additional quantity of reinforcement needed to form the lap splice.

3.5.2.6. Column Capacity Constraints

The column capacity requirement is demonstrated in interaction diagram shown in Figure 16. OB is the segment from the origin to the point that has the applied factored ultimate moment and axial load as coordinates. Segment OA is from the origin to the point A on the surface of the interaction diagram such that O, A, and B are collinear. For the point B to be inside the interaction diagram, OA must be greater than OB. The constraint in EQ. (81) satisfies this requirement.

$$g = \frac{(OB - OA\,)}{OA} \qquad \text{EQ. (81)}$$

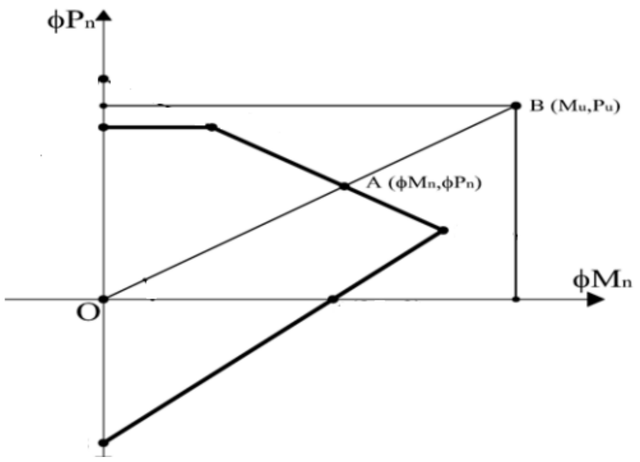


Figure 16 Interaction diagram with expected capacity at point A and applied load at point B (Kaveh & Sabzi, 2012)

3.5.2.7. The Strong Column Weak Beam Constraints

Failure of the columns during a seismic event can lead to more critical consequences than the failure of beams since columns support the weight of entire stories above them while beams support only loads on the on their spans (Moehle, 2014). The failure of all the columns in a single story can result in a story mechanism (Moehle, 2014). A story mechanism results in an instability that can lead to the collapse of the entire building (Moehle, 2014). A more preferred mechanism is the intermediate mechanism where the damage is concentrated in the beams and only some columns, so stability is maintained. To ensure this mechanism occurs, the ACI 318 requires adopting the strong column-weak beam design philosophy. The strong column-weak beams conditions require for each beam-column joint, the sum of the nominal moment capacities of the columns be 1.2 times greater than the sum of the nominal moment capacities of the beams as illustrated in Figure 17. This requirement can be imposed using the constraint in EQ. (82).

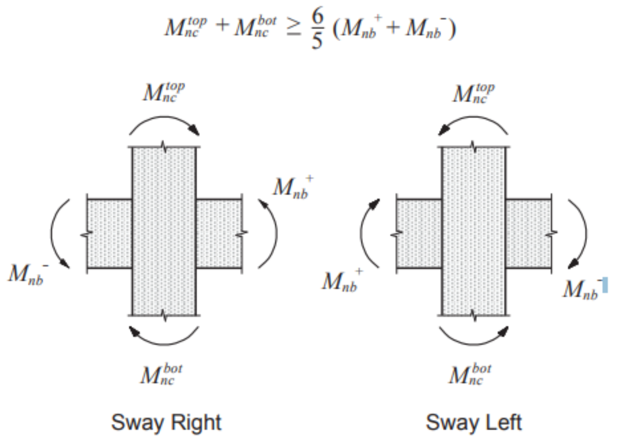


Figure 17 Free body diagrams of the joints subjected to the nominal moment capacities of the beams and columns (Moehle, 2014)

$$g = \frac{\left(1.2 - \frac{\sum M_{nc}}{\sum M_{nb}}\right)}{1.2} \qquad \text{EQ. (82)}$$

3.5.2.8. Column Hoop Spacing Constraints

The hoop spacing in columns should satisfy both confinement and shear demand requirements. The confinement requirements are from sections 18.7.5.3 and 18.7.5.4 of the ACI 318-14 code. EQ. (83) summarizes the confinement requirements. In EQ. (83), b_c and h_c are the width and height of the column, A_{sh_1} is the area of transverse reinforcement bars perpendicular to b_c , A_{sh_2} is the area of transverse reinforcement bars perpendicular to h_c, h_x is the maximum spacing between longitudinal bars, A_g is the gross cross sectional area of the column, A_{ch} is the area of the concrete core enclosed by the hoops, P_u is the maximum factored applied axial load, and k_f and k_n are determined from EQ. (84) and EQ. (85). In EQ. (85), n_l is the total number of longitudinal reinforcement bars supported by a seismic hoop or cross tie.

$$s_c = min\begin{cases} \frac{b_c}{4} \\ 6\ in \\ 4+\frac{14-h_x}{3} \\ \frac{A_{sh_1}}{b_c\left(0.3\left(\frac{A_g}{A_{ch}}-1\right)\left(\frac{f_c'}{fy}\right)\ \right)} \\ \frac{A_{sh_1}}{b_c\left(0.09\frac{f_c'}{f_y}\right)} \\ \frac{A_{sh_1}}{b_c\left(\frac{0.2k_f k_n P_u}{f_y A_{ch}}\right)} \\ \frac{A_{sh_2}}{h_c\left(0.3\left(\frac{A_g}{A_{ch}}-1\right)\left(\frac{f_c'}{fy}\right)\ \right)} \\ \frac{A_{sh_2}}{h_c\left(0.09\frac{f_c'}{f_y}\right)} \\ \frac{A_{sh_2}}{b_c\left(\frac{0.2k_f k_n P_u}{f_y A_{ch}}\right)} \end{cases} \qquad \text{EQ. (83)}$$

$$k_f = max\left(\frac{f_c'}{25000}+0.6, 1\right)\ \ (psi) \qquad \text{EQ. (84)}$$

$$k_n = \frac{n_l}{n_l - 2} \qquad \text{EQ. (85)}$$

As for the shear demand requirements, the spacing is determined similar to that of beams using EQ. (75).

3.5.2.9. Joint Shear Requirements

Checking the joint shear is critical because it can often govern the size of the columns of moment resisting frames (Moehle, 2014). According to the ACI 318-14 code, the capacity design approach is used to determine the joint shear. First, the free body diagram in Figure 18 is used to determine the shear V_{col} at the mid-height of the columns confining the beam-column joints. $M^{-}_{pr,A}$ and $M^{+}_{pr,B}$ are the probable moment capacities of the beams on the left and right side of the joint, $V_{e2,A}$ and $V_{e1,B}$ are the shears calculated in EQ. (86), and l_c is the distance between the mid-heights of the columns confining the joint. Using the free diagram in Figure 18 and the equilibrium equations allows for determining V_{col} as shown in EQ. (86).

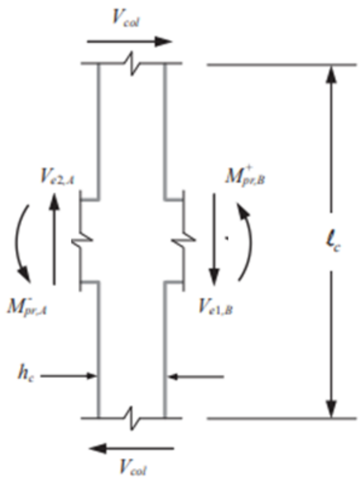


Figure 18 Free body diagram for calculating the columns' shear when the beams reach their probable moment capacities (Moehle, 2014)

$$V_{col} = \frac{M^{-}_{pr,A} + M^{+}_{pr,B}}{l_c} - \frac{V_{e2,A} + V_{e1,B}}{l_c} * h_c \qquad \text{EQ. (86)}$$

After calculating V_{col}, the free body diagram in Figure 19 is used to determine the shear inside the beam-column joint. From the section analysis, the compressive force C_{pr} developed in the concrete should be equal to the tensile force T_{pr} developed in the flexure reinforcement. Taking that into account and applying the equilibrium equations on Figure 19, allows for determining the joint shear as shown in Eq. (87) .

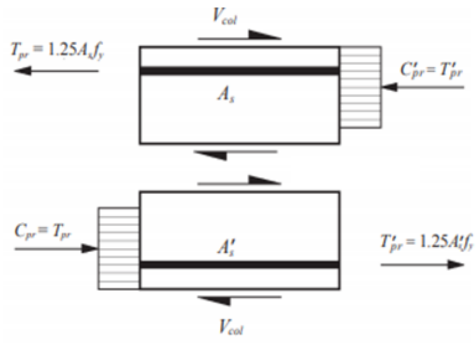


Figure 19 Free body diagram used for calculating the joint shear induced when the beams' reinforcements reach their probable capacities (Moehle, 2014)

$$V_u = V_{col} - T_{pr} - {T_{pr}}' \qquad \text{Eq. (87)}$$

The shear strength of the joint is determined as shown in EQ. (88).

$$V_j = \gamma\sqrt{f_c'}h_cb_c \ \ (psi) \qquad \text{EQ. (88)}$$

Where γ is dependent on the joint’s configuration.

To ensure that V_u is less than V_j, the constraint in EQ. (89) needs to be satisfied.

$$g = \frac{V_u - V_j}{V_j} \qquad \text{EQ. (89)}$$

3.5.2.10. Column Reinforcement Ratio Requirements

The ACI 318-14 code requires that the reinforcement ratio for column be between 1% and 6%. The constraints in EQ. (90) and EQ. (91) are used to enforce these requirements.

$$g = \frac{(1\% - \rho)}{1\%} \qquad \text{EQ. (90)}$$

$$g = \frac{(\rho - 6\%)}{6\%} \qquad \text{EQ. (91)}$$

3.5.2.11. Spacing of Longitudinal Bars

In the literature, the longitudinal bars spacing requirements are accounted for independently for beams and the columns that support them. In reality, the longitudinal reinforcement of the beams and columns pass by each other through the beam-column joint, so clashes between the beams and columns reinforcement should be eliminated to achieve a more practical design. In this research, the following procedure is followed to avoid clashes between the column and beam reinforcement. First, place the column's reinforcement and the beam's primary reinforcement bars such that they are equally spaced inside their respective sections. In the initial configuration, the beam's reinforcement will possibly pass through the column's reinforcement as shown in Figure 20. Therefore, if a clash is detected, move beam's primary reinforcement bars such that they are tangent to the column's reinforcement as shown in Figure 20.

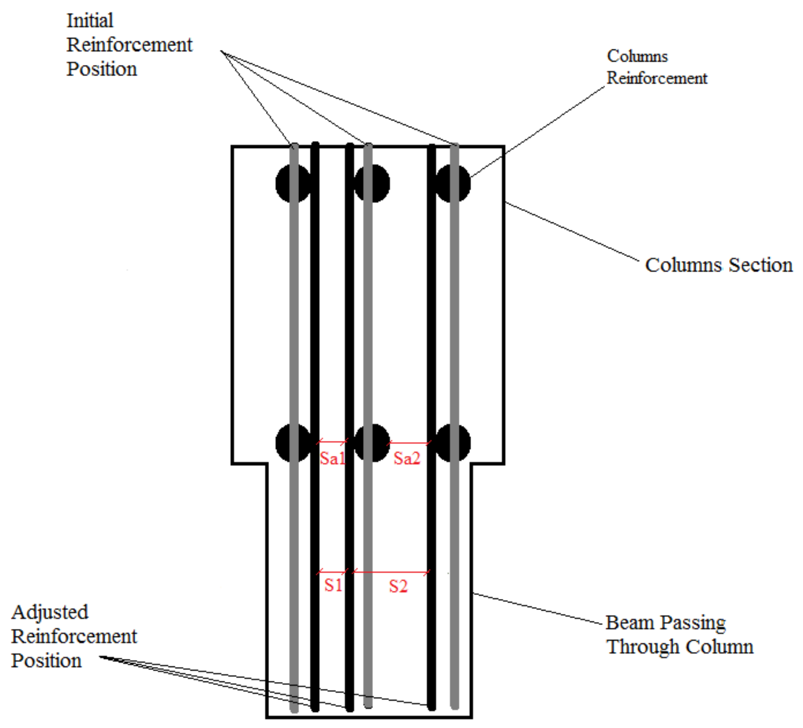


Figure 20 Adjustment of the beams longitudinal reinforcements' positions to avoid clashes with the columns reinforcement inside the beam-column joint

The reinforcements' adjusted positions can now be used to calculate the spacing Si of the beam's primary reinforcement. In the example in Figure 20, the spacing $S1$ and $S2$ need to be evaluated. The spacing needs to be compared to the ACI 318 minimum requirements shown in EQ. (92). This requirement can be satisfies using the constraint in EQ. (93).

$$s_{min} = max \begin{cases} d_b \\ 1\ in \end{cases} \qquad \text{EQ. (92)}$$

$$g = \frac{(s_{min} - si\ )}{s_{min}} \qquad \text{EQ. (93)}$$

The spacing requirements of the column's longitudinal reinforcement can be evaluated using a similar procedure as in EQ. (92) and EQ. (93).

What remains is dealing with placing the cutoff reinforcement bars. Figure 20 shows that after moving the longitudinal reinforcement, some spacing Sai may be available for placing the cutoff reinforcement. In this example, $Sa1$ and $Sa2$ could possibly allow additional reinforcement bars to pass through.

The primary reinforcement require a minimum clear spacing of s_{min} calculated as shown in EQ. (92), and the secondary reinforcement require a minimum clear spacing $s_{\min}^{cut}$ calculated as shown in EQ. (94), where d_b^{cut} is the diameter of the cutoff bars. Therefore, when Sai is bounded by primary reinforcement, a distance may need to be subtracted from Sai for each primary reinforcement bar to produce Sri to satisfy a minimum clear spacing $s_{\min}^{max} = \max(s_{min}, s_{\min}^{cut})$ as shown in Figure 21. In this example, $2s_{\min}^{max}$ is subtracted from $Sa1$, $s_{\min}^{max}$ is subtracted from the right of $Sa2$, and a distance is provided from the left of $Sa2$ such that it satisfies $s_{\min}^{max}$ and the column rebar is avoided. Then, the maximum allowable number of cutoff reinforcement bars that can pass through Sri can be calculated using EQ. (95).

$$s_{\min}^{cut} = max\begin{cases} d_b^{cut} \\ 1\ in \end{cases} \qquad \text{EQ. (94)}$$

$$n_{cut} = \frac{\left(Sri + s_{\min}^{cut}\right)}{s_{\min}^{cut} + d_b^{cut}} \qquad \text{EQ. (95)}$$

n_{cut} is compared to the variable controlling the number of cutoff bars shown in Table 1, and the minimum of the two is chosen.

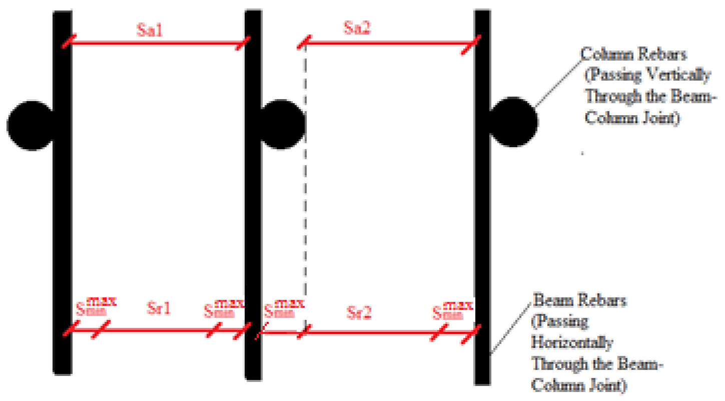

Sa1
Sa2
Column Rebars (Passing Vertically Through the Beam-Column Joint)
Smin max
Sr1
Smin max
Smin max
Sr2
Smin max
Beam Rebars (Passing Horizontally Through the Beam-Column Joint)

Figure 21 Reduction of the available spacing Sai to ensure the beam's primary reinforcements' spacing requirements are satisfied when the secondary reinforcement bars are placed

3.5.2.12. Drift Requirement

For concrete special moment resisting frames, the ASCE 7-10 requires the story drift δ_s to adhere to EQ. (96). C_d is the deflection amplification factor, which is taken as 5.5 for RC special moment resisting frames, I is the importance factor, which is dependent on the risk category of the structure, and δ_e is the elastic drift estimated from the linear static analysis. The drift requirement can be satisfied using the constraint in EQ. (97).

$$\delta_s = \frac{C_d}{I}\,\delta_e \leq 0.02 \qquad \text{EQ. (96)}$$

$$g = \frac{\delta_s - 0.02}{0.02} \qquad \text{EQ. (97)}$$

3.5.2.13. Column Connectivity Requirement

When columns belong to the same column grid line, columns on lower stories should have dimension greater or equal to the dimensions of the columns on the higher stories. This requirement is necessary so that the reinforcement bars of the columns of the higher story can be embedded inside the columns of the lower story. To enforce this requirement, the column dimensions of columns belonging to the same column grid line are sorted as shown in Figure 22.

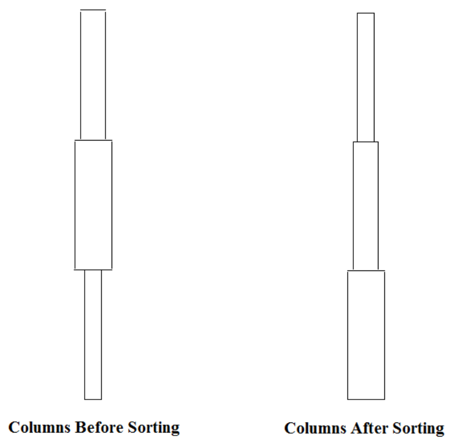


Figure 22 Sorting the columns' dimensions to satisfy connectivity requirements

3.5.2.14. Controlling the Locations Plastic Hinge Formation Along a Beam’s Span

In a typical moment resisting frame, the largest positive and negative moments occur at the ends of the beams and plastic hinges first form at those locations (Moehle, 2014). Figure 23 (a) demonstrates such a behavior. In cases as such, the beam plastic hinges undergo reversing cycles as the building sways back and forth (Moehle, 2014). This is the desirable behavior of moment resisting frames (Moehle, 2014). However, if the span of the beams or the uniform distributed load is relatively large, the maximum positive moments may occur away from the joints as shown in Figure 23 (b), leading to an undesirable behavior (Moehle, 2014). In this case, when plastic hinges form, they do not reverse but instead buildup resulting in progressively increasing rotations. (Moehle, 2014)

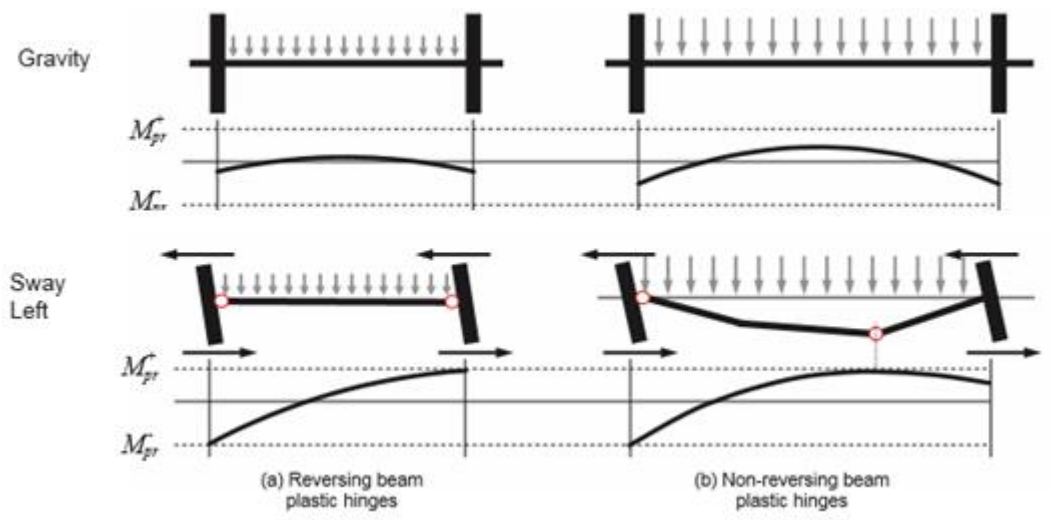

Gravity
M^{+}_{pr}
M^{-}_{pr}
Sway Left
M^{+}_{pr}
M^{-}_{pr}
(a) Reversing beam plastic hinges
(b) Non-reversing beam plastic hinges

Figure 23 Probable moment diagrams for (a) Reversing plastic hinges (b) non reversing plastic hinges due to larger spans or heavier gravity loads (Moehle, 2014)

This undesirable behavior can be avoided if the positive probable moment strength at the end of the beam M_{pr}^{end+} is selected according EQ. (98). EQ. (98) results in a moment diagram similar to that in Figure 24, where the shear maintains the same sign along the beam’s span (Moehle, 2014).

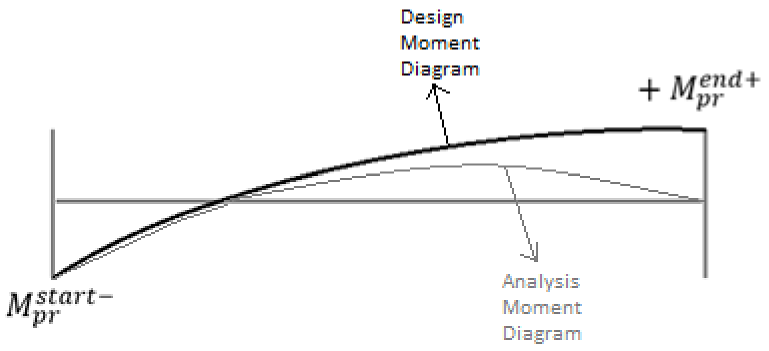


Figure 24 The moment diagram from structural analysis (grey), and the moment diagram resulting from satisfying EQ. (98) (black), the positive probable moment is chosen such that the shear is positive

$$|M_{pr}^{start-}| + |M_{pr}^{end+}| \geq \frac{w_u l_u^2}{2} \qquad \text{EQ. (98)}$$

M_{pr}^{start-} is the negative probable moment strength at the start of the beam, w_u is the factored gravity load, and l_u is the span of the beam. This equation can also be used for the case when the positive probable moment is at the start and the negative probable moment is at the end (Moehle, 2014).

When $-M_{pr}^{start-} + M_{pr}^{end+} = \frac{w_u l_u^2}{2}$, the shear at the start of the beam V_{start} is $w_u l_u$, and the shear at the end of the beam V_{end} is zero. M_{pr}^{start-} can be approximated from the factored applied moment M_u^{start-} as $M_{pr}^{start-} = 1.25 * \frac{M_u^{start-}}{0.9}$ (using ACI 318 Factors $M_u = 0.9M_n$ and approximating $M_{pr} \approx 1.25\, M_n$ since ACI 318 assumes steel probable strength f_{pr} is 1.25 its yield strength f_y). Knowing V_{start}, V_{end}, and M_{pr}^{start-}, M_{pr}^{end+} can be calculated.

3.5.3. Grouping Beams, Columns, and Joints

The number of design variables can grow significantly with the size of the frame being optimized. As a result, the time needed to optimize the cost of the frame can grow significantly as well (Govindaraj & Ramasamy, 2007). Therefore, the literature reduces the number of design variables by categorizing the beams and columns into groups such that the structural elements that belong to the same group share the same design variables (Govindaraj & Ramasamy, 2007). In the literature, the beams and columns are grouped using the engineer's intuition to identify structural elements that should share an identical design. In this research, an analytical approach is utilized, whereby a preliminary design is determined for each of the beams and columns of the frame, and elements that share the same preliminary design are assigned to the same group.

To conduct the preliminary design, the following steps are follows

1. Construct a beam cross section database and a column cross section database. The cross sections in both databases only include the design variables shown in Figure 25. Furthermore, the cross sections in the both databases satisfy the reinforcement ratio requirements and the longitudinal reinforcement spacing requirements. More information on constructing a database of cross sections can be found in the paper by Lee & Ahn (2003).
2. Run the structural analysis once with all beams and columns having average geometric properties.
3. For each beam, get the maximum absolute applied moment, and for each column, get the factored axial load and moment resulting from each of the load combinations.

4. For each beam, calculate the penalized cost considering only the moment capacity constraint, and choose the cross section with the minimum penalized cost.
5. The preliminary cross sections of the columns will be chosen using the following procedure. Start the preliminary design process with the columns on the lowest story. Calculate the penalized cost considering only the column capacity constraint, and choose the cross section with the minimum penalized cost. For the column on higher stories, only consider the cross sections in the database that have dimensions equal to or smaller than the dimensions of the column on the story below. Calculate the penalized cost considering only the column capacity constraint, and choose the cross section with the minimum penalized cost.
6. Assign the beams that have identical preliminary cross sections the same beam group identification number, and assign the columns that have identical preliminary cross section dimensions the same column group identification number.
7. To assign the joints to joint groups, perform the following procedure. For each joint, get and sort the beam group identification numbers of the beams to the left and right of the joint. The purpose of the sorting process is to neglect the order of the beam group identification numbers. Joints that share an identical sorted list of identification numbers are assigned to the same joint group.

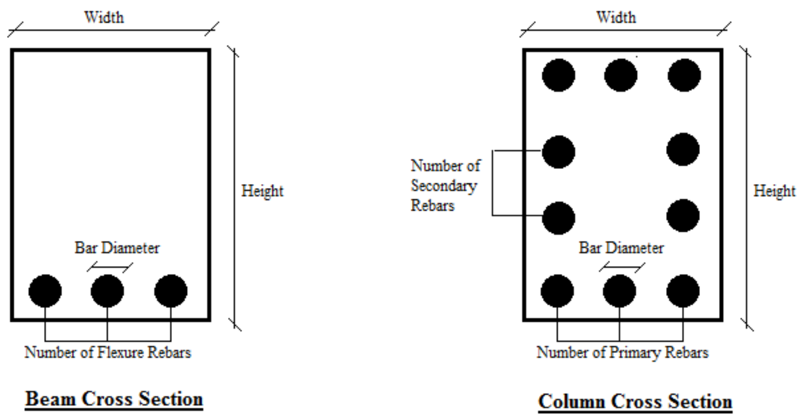


Figure 25 Typical beam and column cross sections used when creating the beam, column, and joint groupings

3.5.4. Summary

Figure 26 summarizes how the genetic algorithm will be integrated into the structural design process. MATLAB will be used to implement the genetic algorithm.

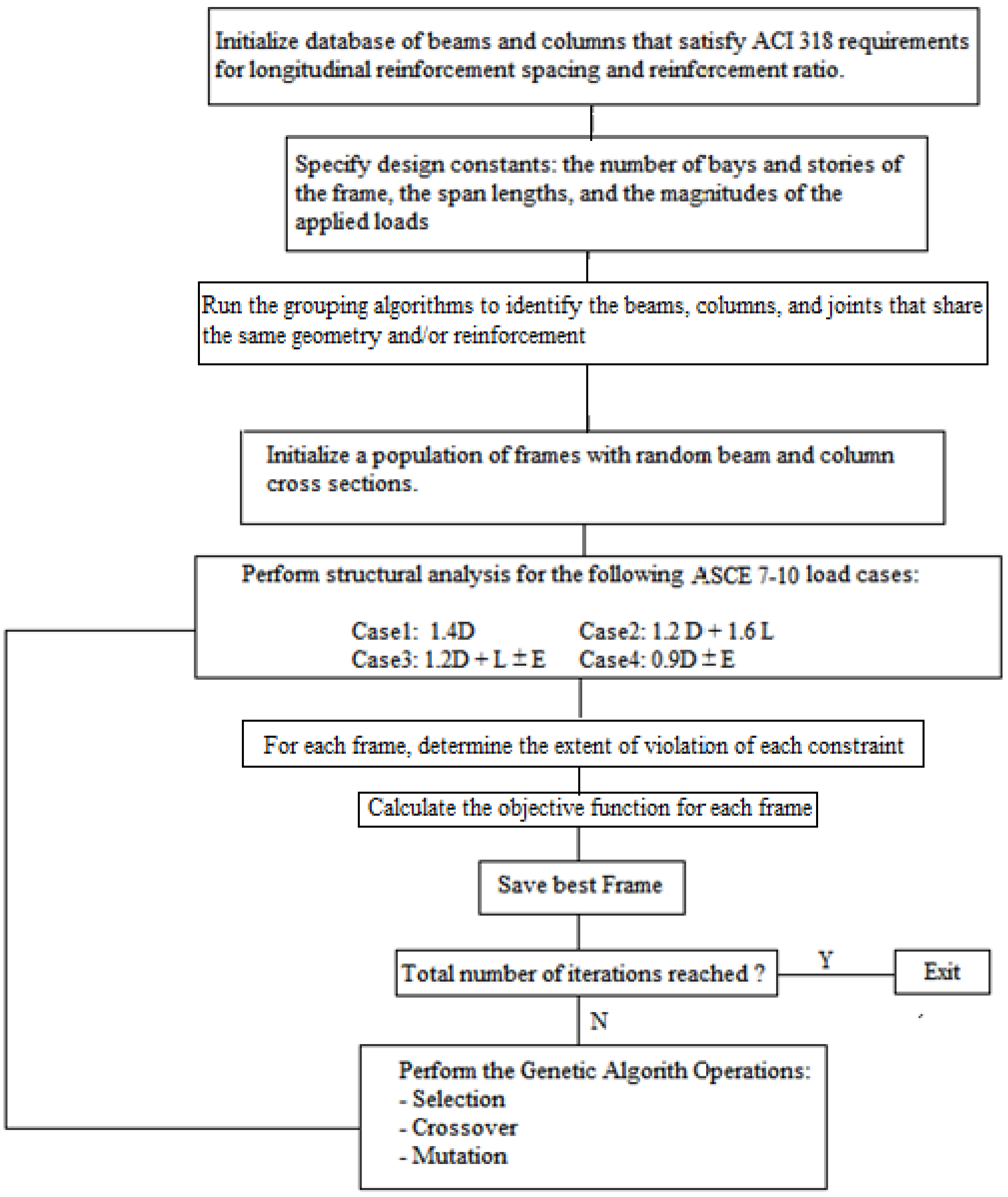

Initialize database of beams and columns that satisfy ACI 318 requirements for longitudinal reinforcement spacing and reinforcement ratio.
Specify design constants: the number of bays and stories of the frame, the span lengths, and the magnitudes of the applied loads
Run the grouping algorithms to identify the beams, columns, and joints that share the same geometry and/or reinforcement
Initialize a population of frames with random beam and column cross sections.
Perform structural analysis for the following ASCE 7-10 load cases:
Case1: 1.4D
Case2: 1.2 D + 1.6 L
Case3: 1.2D + L ± E
Case4: 0.9D ± E
For each frame, determine the extent of violation of each constraint
Calculate the objective function for each frame
Save best Frame
Total number of iterations reached ?
Y
Exit
N
Perform the Genetic Algorith Operations:
- Selection
- Crossover
- Mutation

Figure 26 Flow chart summary of integrating the genetic algorithm into structural design

3.6. Choosing the Diversity Maintaining Mechanism

The following methodology will be used to select the diversity maintaining algorithm. For each diversity maintaining mechanism, the GA will be run five times on each of the frames presented in Figure 27 and for each of the population sizes of 50, 100, and 150 (45 runs in total). Those population sizes will be initially tested because they are within the typical range of practical population sizes found in the literature (Govindaraj & Ramasamy, 2007; Lankhorst, 1996; Lee & Ahn, 2003). Then, the average penalized costs and the convergence times will be recorded and analyzed to choose a diversity maintaining mechanism. At this stage, the objective is to identify the most cost-effective algorithm (the algorithm that result in the lowest penalized cost) and eliminate algorithms that require larger convergence times to reach higher penalized costs.

The algorithms that pass this stage will then be run for increasing values of population size (200 – 250 increments) until they plateau. Then, the penalized costs and convergence times will be compared at the plateau. A plateau will be defined as the point at which an increment of the population size only results in a reduction of the penalized cost less than or equal to 1%.

At the Second stage, the following steps will be followed to eliminate algorithms.

- Choose the most cost-effective algorithm: the algorithm that result in the lowest penalized cost will be chosen.
- Eliminate algorithms that plateau at penalized costs larger than 10% of the cost achieved by the most cost-effective algorithm, and do not provide a significant improvement on the convergence times (at least 30% to 50%).
- From the remaining algorithms other than the most cost-effective algorithm, choose the one with the least convergence time.

Based on these criteria, the most cost-effective algorithm could be chosen alone or with an algorithm that is more computationally efficient while providing estimates of the penalized cost within a 10% range.

Before running the Genetic Algorithm, it is important to consider how the genetic algorithm will be terminated. One common method of terminating the GA is though specifying a ceiling for the number of iterations it can carry out. Common values found in the literature are 50, 100, 150, 200, and 400 iterations. In this research, the ceiling for the number of iterations will be set to 2000 to be conservative.

Although specifying a ceiling for the maximum number of iterations is a common and simple termination method, this approach may result in running the GA for an unnecessary number of iterations after it had converged to a local optimum. Therefore, a stopping criterion that stops the GA when it fails to improve is important to avoid unnecessary computational burden when possible. In this research, the GA will be halted if the criterion in EQ. (99) is satisfied. li is the current iteration index.

$$Stopping\ Criterion = \frac{Penalized\ Cost\left(\frac{li}{2}\right) - \ Penalized\ Cost(li)}{Penalized\ Cost(li)} \le 1\% \ \ if\ li \ge 500 \qquad \text{EQ. (99)}$$

3.7. Assessing Buildability and the Productivity of Workers

Many factors influence the productivity, but buildability is the most important (Jarkas, 2005). The Construction Industry Research and Information Association defines buildability as ‘the extent to which the design of a building facilitates ease of construction, subject for the overall requirements for the completed building” (Jarkas, 2005).

Jarkas (2005, 2010b, 2010a, 2011, 2012) developed regression models that relate labor productivity to various buildability factors. Such models can be used in quantifying the constructability of designs in a more objective manner. Designs with higher productivity values must have more favorable buildability factors meaning that they are easier to construct. Below are the regression models developed by Jarkas (2005, 2010b, 2010a, 2011, 2012).

3.7.1. Rebar Installation in Rectangular Columns

The following regression model defines the relationship between rebar installation in rectangular columns and buildability factors:

$$P_{rebar}^{column}(Kg/mh) = -5.76 + 4.98CBDia + 0.114Q \qquad \text{EQ. (100)}$$

Where P_{rebar}^{column} is the productivity of the workers during rebar installation, CBDia (mm) is the diameter of the longitudinal bars, and Q (Kg) is the total quantity of reinforcement.

3.7.2. Rebar Installation in Rectangular Beams

The following regression model defines the relationship between rebar installation in rectangular beams and buildability factors:

$$P_{rebar}^{beam}(Kg/mh) = 5.78 + 6.23CBDia - 2.65SDia + 0.0286Q - 0.0515W - 0.00832D \qquad \text{EQ. (101)}$$

Where P_{rebar}^{beam} is the productivity of the workers during rebar installation, CBDia (mm) is the diameter of the longitudinal bars, $SDia$ is a dummy variable that represents diameter of the stirrups ($SDia = 0\ for\ 8mm\ and\ SDia = 1\ for\ 10mm$), Q (Kg) is the total quantity of reinforcement, W is the width of the beam, and D is the depth of the beam.

3.7.3. Formwork Construction in Rectangular Columns

The following regression model defines the relationship between formwork construction for rectangular columns and buildability factors:

$$P_{formwork}^{column}(m^2/mh) = 2.41 + 0.379SA \quad \text{EQ. (102)}$$

Where $P_{formwork}^{column}$ is the productivity of workers during formwork construction, and SA is the total surface area of the completed formwork.

3.7.4. Formwork Construction in Rectangular Beams

The following regression model defines the relationship between formwork construction for rectangular beams and buildability factors:

$$P_{formwork}^{beam}(m^2/mh) = 5.46 + 0.0831SA + 1.43RF \quad \text{EQ. (103)}$$

Where $P_{formwork}^{beam}$ is the productivity of workers during formwork construction, SA is the total surface area of the completed formwork, and RF is a "repetition factor" that assumes two values: 0 if this is the first formed beam and 1 if this is a repeated beam.

3.7.5. Placement of Pumped Concrete

The following regression model can determine the relationship between pumped concrete productivity and buildability factors:

$$P_{concrete}\ (m^3/mh) = 2.31 + 0.00322V - 0.0238H - 0.00181SCR \quad \text{EQ. (104)}$$

Where $P_{concrete}$ is the productivity of concrete pumping labor, V (m^3) is the volume of the pumped concrete, H is the elevation above the ground level (m), and

SCR (Kg/m^3) is the steel congestion ratio defined as the ratio of quantity of steel (Kg) to the volume of pumped concrete.

3.8. Assessing the Effect of Buildability on the Cost Optimization Process

Translating buildability factors to monetary values facilitates integrating the concept of buildability into the penalized cost function in EQ. (9). For each structural member, the productivity of rebar installation P_{rebar}, formwork construction $P_{formwork}$, and concrete pouring $P_{concrete}$ are calculated using EQ. (100), EQ. (101), EQ. (102), EQ. (103), and EQ. (104).

Next, the man hours per structural member (m_i) can be obtained as follows:

$$m_i = \frac{Q_i}{P_{rebar}} + \frac{SA_i}{P_{Formwork}} + \frac{V_i}{P_{concrete}} \qquad \text{Eq. (105)}$$

The total man hours (MH) needed to construct the entire frame are approximated as

$$MH = \sum m_i \qquad \text{Eq. (106)}$$

Then, the average cost of labor (C_{labor}) is obtained by multiplying the total man hours by the average unit cost per man hour $(U_{manhour})$.

$$C_{labor} = MH \times U_{manhour} \qquad \text{Eq. (107)}$$

Finally, C_{labor} is integrated into the penalized cost in EQ. (9) to produce the following.

$$C_{penalized} = (C_{material} + C_{labor})(1 + G) \qquad \text{Eq. (108)}$$

To evaluate the effect of including buildability in the cost optimization process, frames having different numbers of bays and stories will be optimized using EQ. (9) and Eq. (108) independently, and the total cost, material costs, and labor costs are compared for the two independent optimizations.

Using the National Construction Estimator, it can be estimated that constructing 1 m^3 of concrete with 1% reinforcement ratio, requires around \$50 per Manhours. To account for increases in labor expenses and overhead costs, a cost of around \$100 per Manhours will be considered as well.

3.9. Unit Material Costs of the Frames

The average unit costs considered are $120\$/m^3$ for concrete, $6800\ \$/m^3$ for longitudinal steel, $10\ \$/m^2$ for formwork (Esfandiari, Urgessa, Sheikholarefin, & Dehghan Manshadi, 2018; Jahjouh et al., 2013; Pray, 2017).

3.10. Frames Considered In to Study The Diversity Maintaining Mechanisms and Buildability Factors

In this study, frames of different sizes are used to study the performance of the diversity maintaining mechanisms and the effect of buildability factors. Figure 27 shows the frames considered in this study.

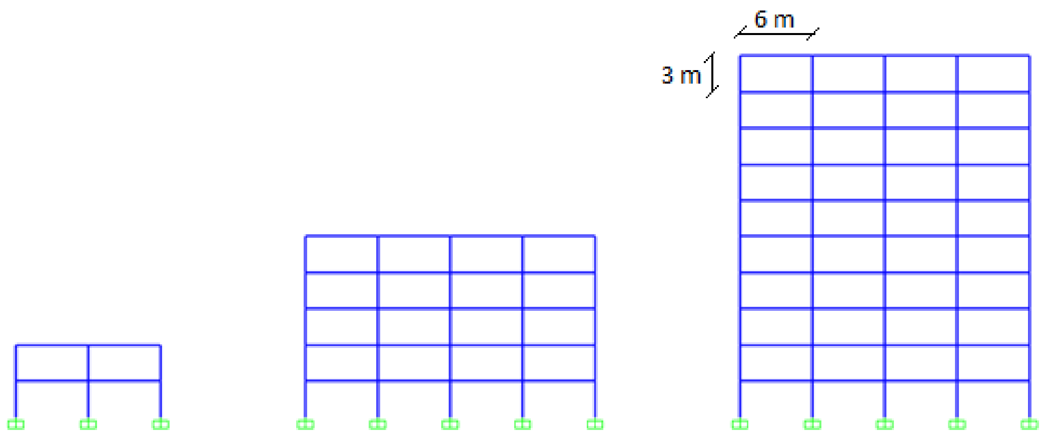


Figure 27 Frames considered in this study

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1. ANN Surrogate for Symmetric Two Bay Frame Subjected to Gravity Load

A neural network was trained to be a surrogate for the structural analysis of a symmetric two bay frame. The training was performed on an Intel Core i7 processor with 8 GB RAM. Figure 28 shows the eight input variables considered for this example problem. $Icolumn$ 1 and $Acolumn$ 1 are the moment of inertia and cross section area for the exterior columns, while $Icolumn$ 2 and $Acolumn$ 2 are the moment of inertia and cross section area for the interior column. Furthermore, Both beams are identical and have a moment of inertia $Ibeam$, cross section area $Abeam$, and length L. Moreover, the beams are subjected to a uniform gravity load Wu. Finally, the spans of the three columns is fixed at 3m, and the modulus of elasticity was fixed at 25 MPa, which is typical of normal strength concrete.

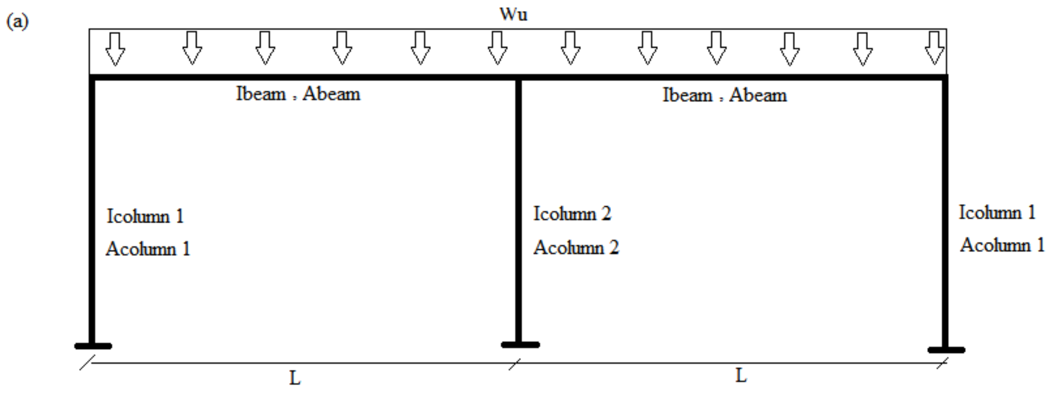


Figure 28 Inputs of two bay-one story frame neural network surrogate

The output variables considered are shown in Figure 29. Due to symmetry, numerous outputs can be neglected. $Mbeam$ and $Vbeam$ are the internal moment and shear in the beam at the end close to the exterior column, $Ncolumn1$, $Vcolumn$, and $Mcolumn$ are the axial, shear and moment reactions in the exterior column, and $Ncolumn2$ is the axial reaction in the interior column. Other internal force can be obtained using the six considered output variables and static equilibrium equations.

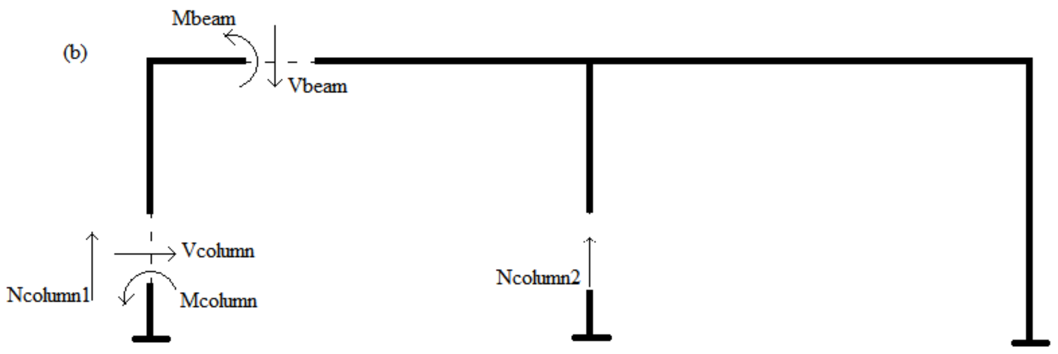


Figure 29 Outputs of two bay-one story frame neural network surrogate

The first neural network trained for this problem has eight inputs, six outputs, and two hidden layers with twenty hidden neurons each as shown in Figure 30. The network was trained using 300 training examples, 30 validation examples, and 3000 testing examples, and the ranges of the inputs are as shown in Table 3.

Table 3 Input ranges for frame structural analysis surrogate neural networks

	Span Length (mm)	**Moment of Inertia Beams/Columns (mm^4)**	**Cross Sectional Area Beams/Columns (mm^2)**	**Uniform Gravity Loads (KN/mm)**
Minimum	3000	10^9	10^4	0
Maximum	12000	10^{11}	10^6	0.2

Figure 31 shows that the network performed better on the training set than on the validation set implying that overfitting is an issue. To overcome overfitting, the number of training samples was increased to 1000. Figure 32 shows an improvement in the discrepancy between the training and validation errors implying better generalization. Then, a few more training regimens were attempted on neural networks of different sizes and using different amounts of training, and Table 4 shows the percentage errors of the predictions of all trained neural networks on the testing set. Table 4 shows that the third training attempt resulted in a maximum error less than 5%, and later attempts do not lead to a significant drop in the error. Therefore, the third neural network is selected as the final model. The time needed to train the third neural network is about two minutes.

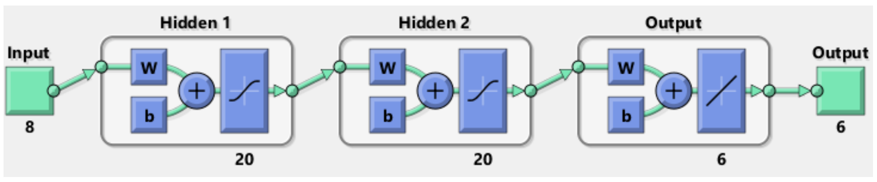


Figure 30 Two bay frame surrogate neural network architecture

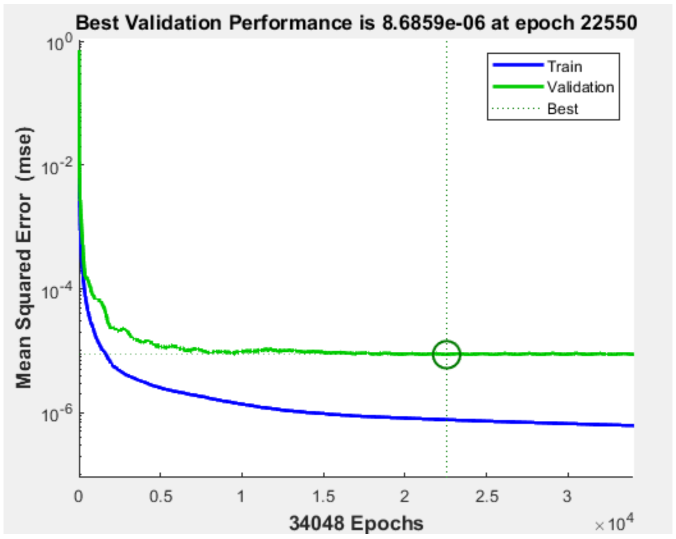


Figure 31 Mean squared error for training and validation data for a neural network with 20 neurons per hidden layer, and trained on 300 training example of a two bay one story frame.

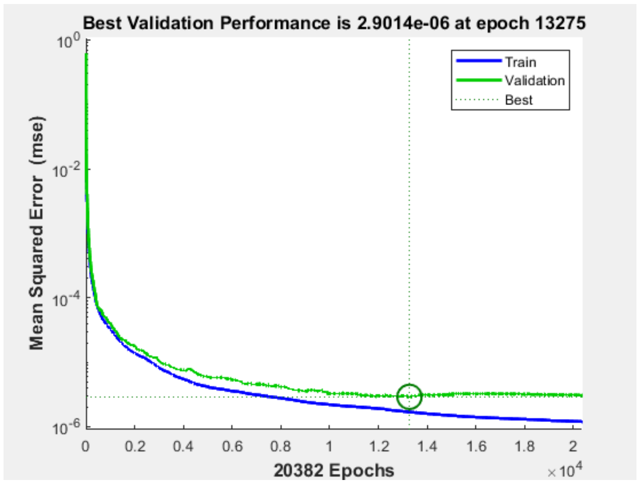


Figure 32 Mean squared error for training and validation data for a neural network with 20 neurons per hidden layer, and trained on 1000 training example of a two bay one story frame

Table 4 Comparison of the performance of different ANNs trained on a different amount of training data

	Number of Hidden Neurons	20	20	25	30	40
	Number of Training Samples	300	1000	1000	1000	3000

Absolute Percentage Error (%)	V_{Column}	5	3.4	1.6	1.79	1.82
	V_{Beam}	1.5	1	0.7	0.76	0.54
	N_{Column1}	1.2	0.8	0.5	0.45	0.36
	N_{Column2}	1.4	1	0.5	0.45	0.48
	M_{Beam}	6.5	5.5	4.2	3.88	2.84
	M_{Column}	5	4	1.7	1.56	1.98

Table 5 compares the time needed by the direct stiffness method and the neural network to analyze the frame. The neural network is about twenty times faster than the direct stiffness method.

Table 5 Time needed by the direct stiffness method versus the neural network to predict the internal forces in a two bay frame

Method	**Time (seconds)**
Direct Stiffness	1.8e-04
Neural Network Surrogate	8.9e-06

The next step is to build neural networks for larger frames, and to monitor the training time and processing time relative to the Direct Stiffness method.

4.2. Results for Two Story, Three Story, and Four Story Two Bay Frames

The same methodology was adopted to train neural networks to predict the internal forces of Two Story, Three Story, and Four Story Two Bay Frames. Figure 33 shows the performance of the neural networks relative to the direct stiffness method, and Figure 34 shows the time needed to train the neural networks to reach the desired 5% percentage error. The results show that although the neural networks are more computationally efficient than the direct stiffness method, the training time increases in an exponential manner to become fifteen hours for the four story two bay frame. The Direct Stiffness method could have been run about 135 million times in the time needed to train a neural network to simulate the analysis of a two bay four story frame subjected to gravity loads. Designing such a small frame for gravity would require an engineer significantly less than 135 million analysis runs.

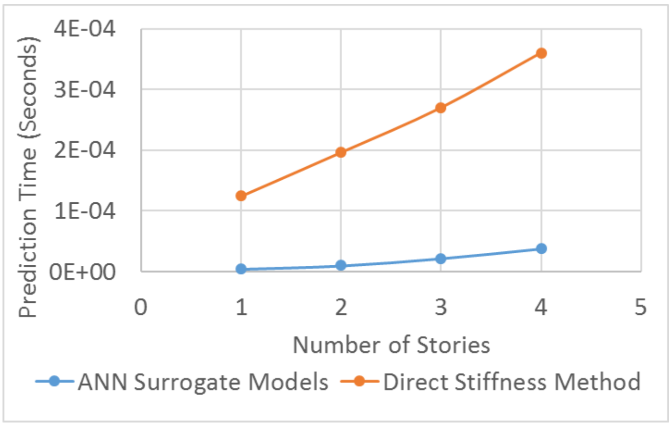


Figure 33 Comparison of the performance of ANNs to the Direct Stiffness Method for frames of different sizes

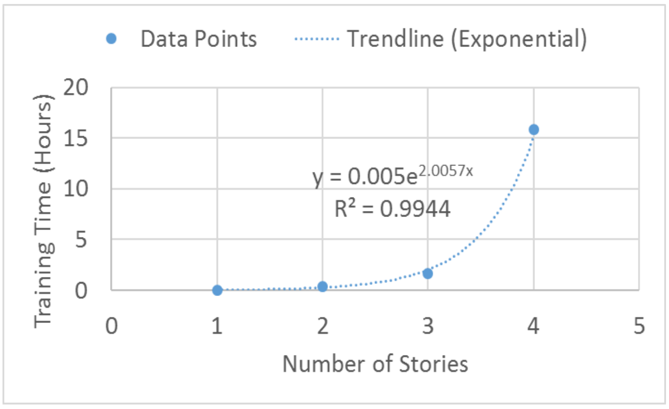


Figure 34 The time needed to training neural networks to simulate the analysis of frames of different sizes

4.3. Constructing the Stiffness Matrix of a Frame Using a Neural Network

The following steps were taken in order to determine the computational cost of constructing the global stiffness matrices of frames using neural networks. First, neural networks where trained to evaluate the stiffness matrices of frames having k bays and k stories (kxk frames). The values of k considered are k = 2, 3, 4, 5, 6, 7, 8. The training time and the prediction time of the neural networks were recorded. The experiment was performed on an Intel Core i7 processor with 8 GB RAM. Figure 35 shows the training time of the neural network as a function of the number of degrees of freedom in the frame structure. Figure 36 and Figure 37 show the time needed by neural networks and traditional algorithms to assemble a frame’s global stiffness matrix.

Figure 35, Figure 36, and Figure 37 highlight two issues with using neural networks to evaluate the global stiffness matrix. Figure 35 demonstrates the rapid nonlinear growth of the time needed to train the neural networks as a function of the degrees of freedom. While a neural network for a frame with 50 DOFs requires about 4 minutes to train, increasing the number of DOFs by five times resulted in the training time growing by around 180 times. Figure 36 shows that the time required by the neural network to evaluate the stiffness matrix has a nonlinear growth in comparison to the linear growth of the conventional algorithm. Matrix – Vector multiplication is needed to evaluate the stiffness matrix using this neural network leading time complexity of O (n^2) (Golub & Van Loan, 2013). Therefore, although the neural network approach saves a significant amount of time relative to the traditional algorithm for small frames with few DOFs, extrapolation of the two methods as shown in Figure 37 shows that the neural network method will lose its utility as the number of DOFs increases to around 560. As a result, a new approach needs to be investigated.

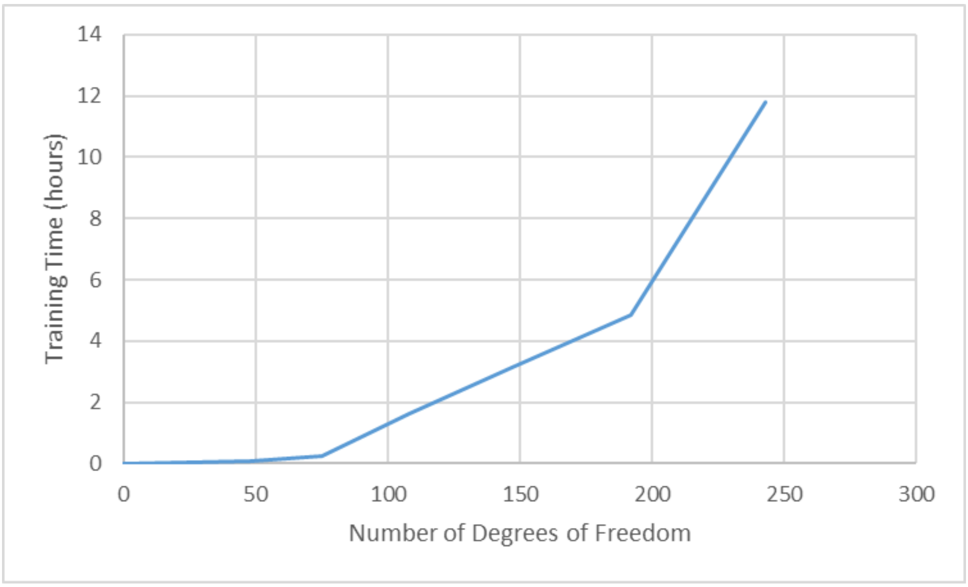


Figure 35 Neural network training time as a function of the number of DOFs

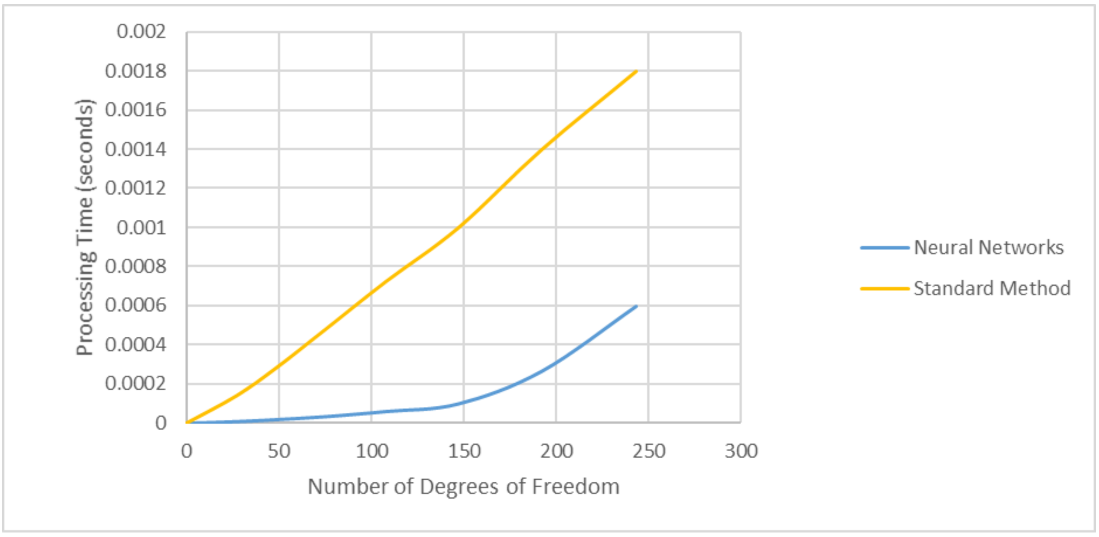


Figure 36 Neural network prediction time in comparison to the standard method of assembling the stiffness matrix

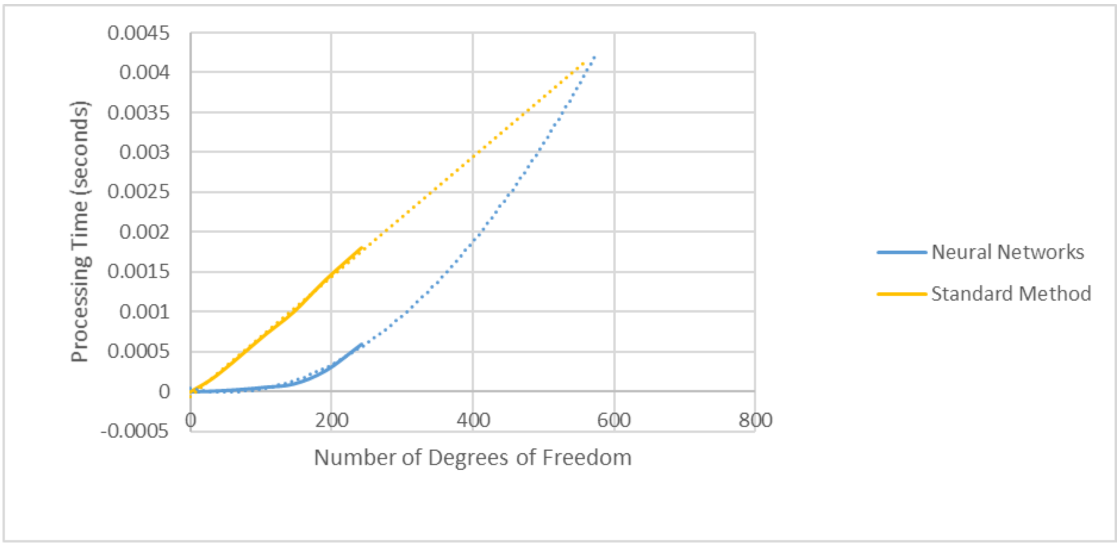


Figure 37 Extrapolation of the Standard and Neural network processing times shows a decrease in the utility of using neural networks as the number of DOFs increases

Since the time saved by neural networks of frame structures with less than 150 DOFs is significant and their training time is relatively low, the next suggested approach uses the neural networks for small frame structures to construct the global stiffness matrix of a larger frame structure. First, the large frame will be subdivided into smaller sub-frames. Second, neural networks will be used to evaluate the stiffness matrices of the sub-frames. Finally, the sub-frame stiffness matrices will be assembled to produce the global stiffness matrix of the complete frame. Figure 38, compares the time complexities of the subdivision approach to previous approaches. It can be noted that the prediction time saved by the subdivision approach is dependent on the size of the sub-frames. Using 2x2 sub-frames saved about 8% of the processing time, while using 4x4 sub-frames saves 40% of the processing time at best, while using a 4x8 sub-frames saves 56% compared to the Conventional method. Furthermore, for this method to be justified, a database of ANNs capable of predicting the stiffness matrices of frames of different sizes should be available.

The time needed to train the 2x2, 4x4, and 4x8 sub-frame ANNs are 2 minutes, 10 minutes, and 180 minutes respectively. The training time increases nonlinearly as shown in Figure 35. However, had ANNs been used without subdivision to predict the stiffness matrices of frames with more than 243 DOFs, the training time of those ANNs would have exceeded 15 hours. Therefore, this method saves on the training time of the ANNs.

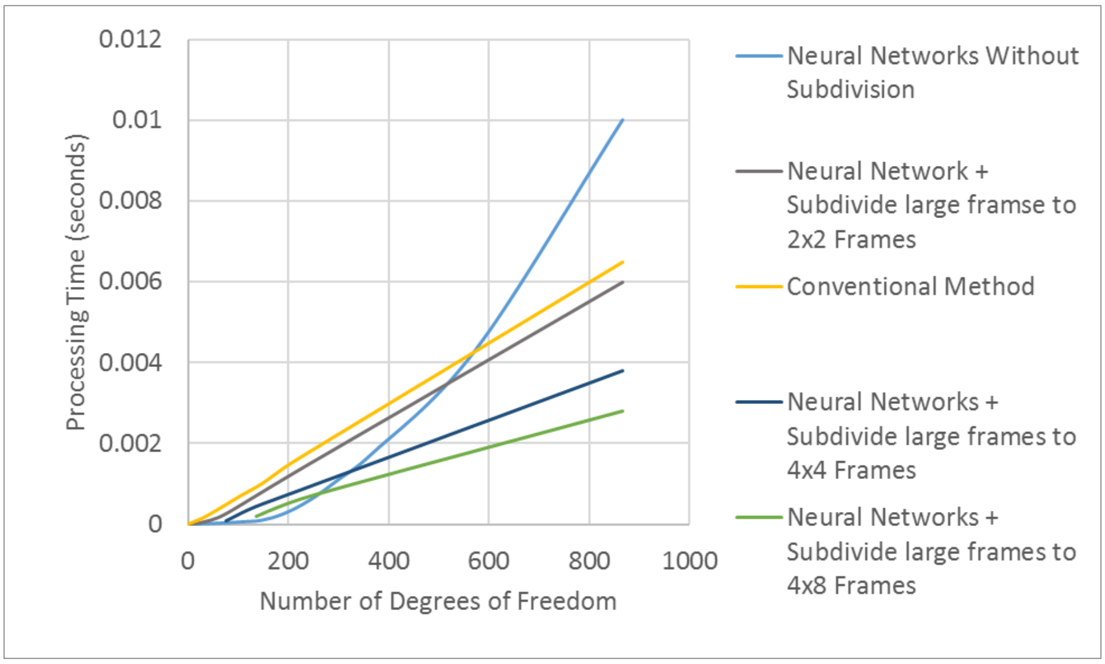


Figure 38 Comparison of processing time of neural networks with frame subdivisions to the previous methods

4.4. Comparing the Performance of Neural Network Approaches and the Vectorized Method for Constructing the Stiffness Matrix of a Frame

Figure 39 shows the processing time needed by all the Conventional method, Neural Network methods, and the Vectorized method to evaluate the global stiffness matrices of frames having a different number of DOFs. It can be noticed that the vectorized algorithm outperforms all of the ANN approaches that require subdividing the frame. Furthermore, when compared to the ANN approach without subdivision, Vectorized Stiffness requires similar processing time for frames with less than 175 DOFs, and Vectorized Stiffness requires less processing times for frames with more than 175 DOFs.

Moreover, Vectorized Stiffness is more practical than subdividing the frame. Subdivision requires beforehand training ANNs capable of processing the sub-frames of specific sizes. On the other hand, the Vectorized Method can process any frame of any size as a whole.

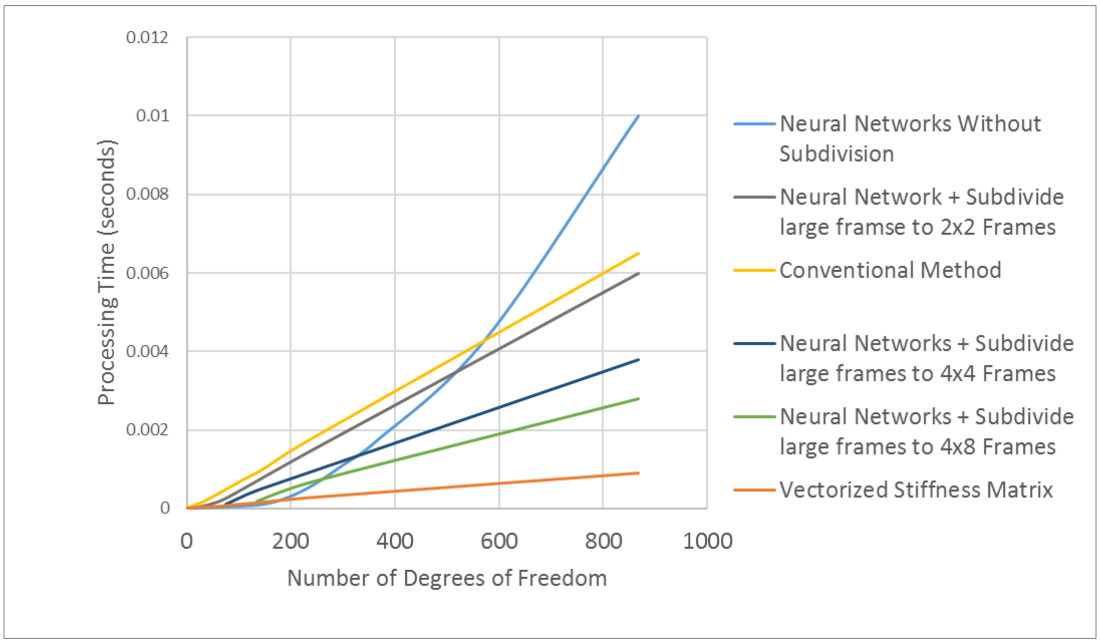


Figure 39 Comparison of the performance of the vectorized stiffness algorithm to other methods of evaluating the stiffness matrix

4.5. Comparison of the Conventional, Vectorized, and Weight Matrix Methods

4.5.1. Comparison of the Computational Time Required to Construct the Global Stiffness Matrices of Frames Using Conventional, Vectorized, and Weight Matrix Methods

The Conventional, Vectorized, and Weight Matrix methods were implemented in MATLAB for plane frames, and then the three algorithms were used to construct the stiffness matrices of plane n bay - n story frames where $n = \{1,2,3 \ldots 21\}$. The experiment was performed on an Intel Core i7 processor with 8 GB RAM. The time needed to assemble the global stiffness matrices was plotted as a function of the degrees of freedom of the frames as shown in Figure 40 , and Table 6 summarizes the results. The results show that both the Vectorized and Weight Matrix methods result in a significant speed up in the assembly time. On average, the Vectorized method is 7 times more efficient than the conventional method, and the Weight Matrix method is 31 times more efficient than the conventional method.

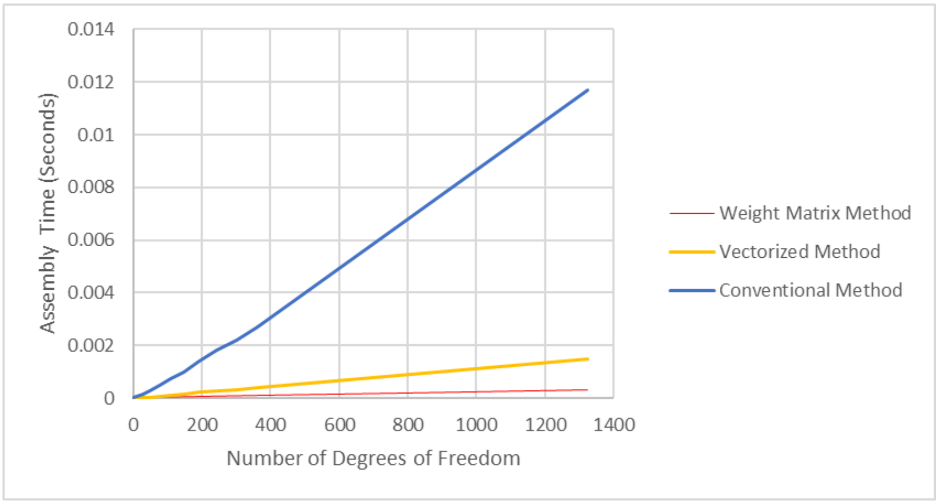


Figure 40 Performance of the Conventional, Vectorized, and Weight Matrix stiffness matrix assembly methods as a function of the number of degrees of freedom of the plane frame structures

Table 6 Summary of the speedups of the Vectorized and Weight Matrix methods relative to the Conventional method of assembling the global stiffness matrix

Method	Number of runs per one run of the Traditional Method
Conventional	1
Vectorized	7
Weight Matrix	31

4.5.2. Comparison of the Performance of the Linear Static Direct Stiffness Method using the Conventional, Neural Network, Vectorized Stiffness, Weight Matrix Methods

In this section, the performance of linear static structural analysis will be compared for the conventional, Neural Networks, Vectorized, and weight matrix methods. All methods will be implemented in MATLAB. The experiment was performed on an Intel Core i7 processor with 8 GB RAM.The time needed to complete the analysis using the mentioned methods will be monitored for frames having a different a number of degrees of freedom. In all cases, the Choleskey factorization algorithm is used to solve for the unknown degrees of freedom. The results in Figure 41 compare the times needed to conduct the linear static structural analysis by all of the methods. The direct stiffness method is about ten times more efficient when the conventional method is replaced by the weight matrix method in the MATLAB code. Furthermore, the linear static analysis using the weight matrix method is 100% less time consuming than the vectorized stiffness method. Finally, using the weight matrix method is about 260% more efficient than using neural networks.

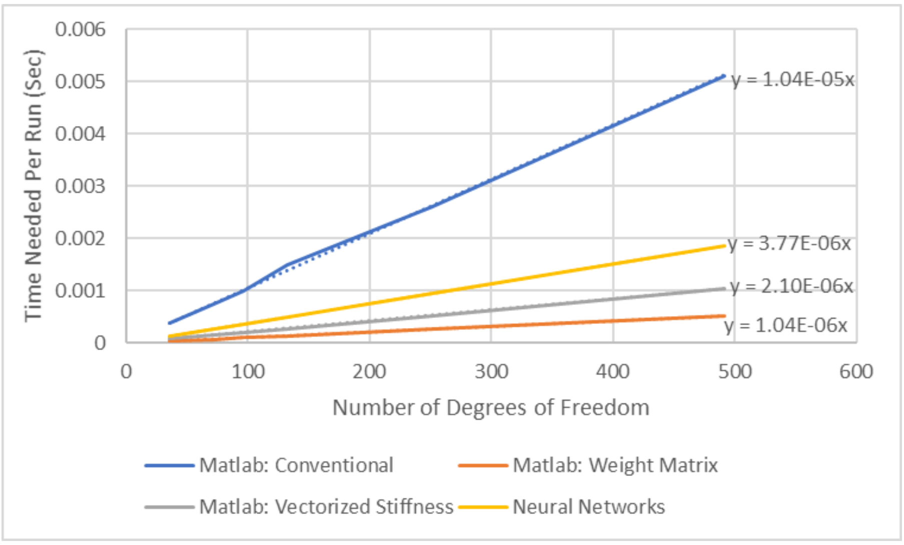


Figure 41 Comparison between the times needed to perform linear static structural analysis using the conventional method, neural networks, vectorized stiffness method, the weight matrix method

4.6. Comparing the Performance of the Diversity Maintaining Mechanisms on the Penalized Cost

Figure 42, Figure 43, and Figure 44 show the average penalized cost as a function of the population size for the different GA selection algorithms tested on 2D frames of Figure 27.

Figure 42 shows the results for the two story frame. Proportional selection resulted in average penalized costs out of the range presented in Figure 42, so the plot of the average penalized costs obtained using proportional selection is represented as a horizontal line at the top of the graphing area of the figure. As the population size increases from 50 to 150, the average penalized cost obtained when using Rank, Fuss, and Rank Space Selection monotonically decrease from $5860 to $5450 (7.5% decrease), from $5980 to $5520 (8.3% decrease), and from $5400 to $5170 (4.4% decrease) respectively. On the other hand, the average penalized cost obtained when using Tournament Selection remains about constant at $5400 as the population size increases from 50 to 150. Among the diversity maintaining selection algorithms, Rank Selection results on the highest average penalized costs, while Rank Space leads to the lowest penalized costs. Furthermore, it can be observed that using Tournament Selection and Rank Space Selection with a population size of 50 resulted in average penalized costs that are lower than those obtained when using Rank and Fuss with a population size of 150.

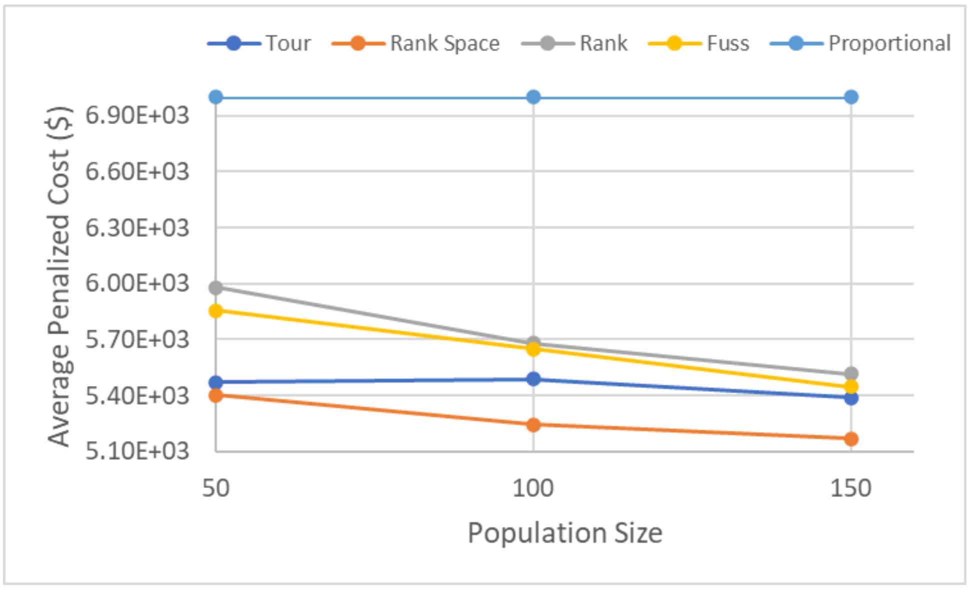


Figure 42 The variation of the average penalized cost of a two story frame as a function of the population size

Figure 43 shows the results for the five story frame. Similar to the results of the two story frame, proportional selection resulted in average penalized costs out of the range presented in Figure 43. Unlike the results of the two story frame, the behavior and relative performance of the diversity maintaining selection algorithms changed. As the population size increases from 50 to 150, the average penalized costs obtained when using Fuss Selection, Tournament Selection, and Rank Selection monotonically decrease from $41600 to $37900 (9.7% decrease), from $38400 to $33300 (15% decrease), and from $37100 to $29900(add %) respectively. Among these three algorithms, Rank Selection results in the lowest penalized costs, followed by Tournament Selection, and Fuss results in the highest costs. On the other hand, using Rank Space Selection results in a non-monotonic behavior. For population sizes of 50 and 100, the average penalized cost decreases from $33700 to $31100 when Rank Space is used. As the population size increases from 100 to 150, Rank Space results in an increase of the penalized cost from $31100 to $33100. For the population sizes of 50 and 100, Rank Space outperforms all other selection algorithms, while for a population size of 150 Rank Space is outperformed by Rank Selection. The difference between the lowest average penalized cost obtained by Rank Space and the lowest average penalized cost obtained by Rank Selection is only 4%.

The following steps were taken to confirm that the non-monotonic behavior of the Rank Space algorithm. First, the optimization was repeated using Rank Space for a population size of 150 leading to a similar result. Then, optimization using Rank Space was done for a population size of 200 showing that the penalized cost stabilizes around $31100. On the other hand, the average penalized cost continues to drop when optimization is carried out using the other algorithms for a population size of 200.

As the size of the frame increased, the behavior of the Rank Space algorithm became less predictable.

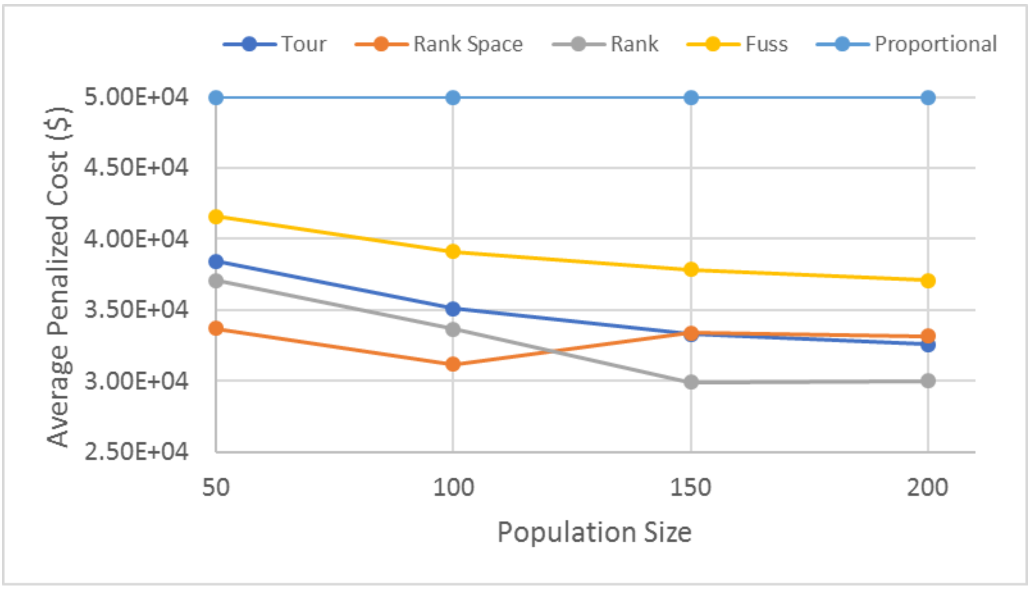


Figure 43 The variation of the average penalized cost of a five story frame as a function of the population size

Figure 44 shows the results for the ten story frame. As the population size increases from 50 to 150, the average penalized costs obtained when using Fuss, Tournament, and Rank Selection decrease from \$168000 to \$83000, from \$111000 to \$68300, and from \$80000 to \$61300 respectively. Among these three algorithms, Rank Selection results in the lowest penalized costs, followed by Tournament Selection, and Fuss results in the highest costs. On the other hand, when Rank Space is used, the average penalized cost increases from \$67400 to \$123000 as the population size increases from 50 to 150. Although Rank Space results in a non-desirable monotonically increasing behavior, Rank Space outperforms all other selection algorithms when the population size is set to 50, and results in an average penalized cost 10% higher than Rank Selection's best performance.

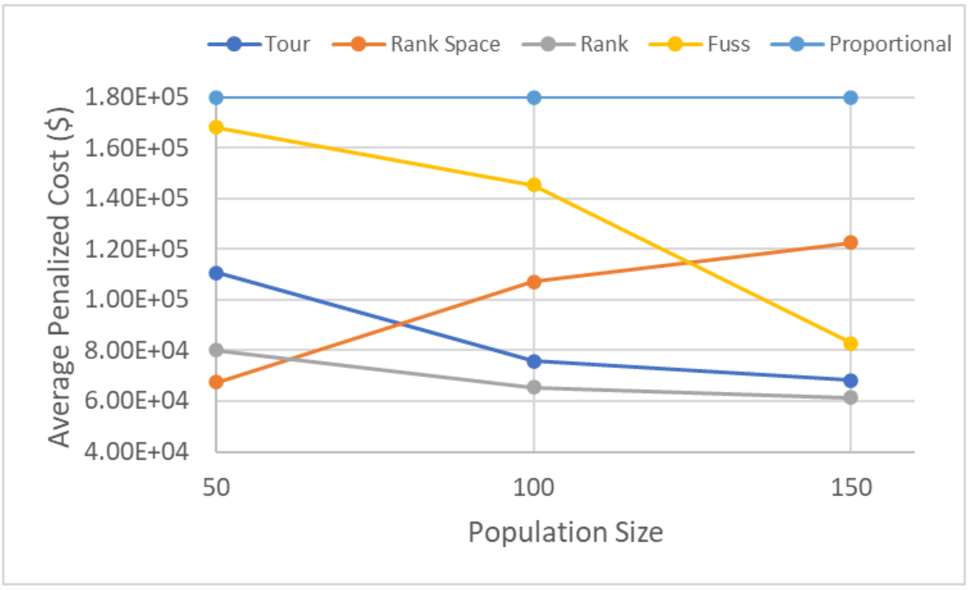


Figure 44 The variation of the average penalized cost of a ten story frame as a function of the population size

The following can be noted from the results in Figure 42, Figure 43, and Figure 44 collectively. First, as the size of the frame increased, the plots of the penalized cost as a function of population size remained monotonically decreasing when Rank, Fuss, and Tournament Selection algorithms were used. On the other hand, when Rank Space is used, the plots of the penalized cost as a function of population size change from monotonically decreasing, to non-monotonic, and then to monotonically increasing, when the frame size increased. This inconsistent behavior of the Rank Space algorithm discourages its use. However, the Rank Space algorithm always outperformed all other algorithms for a small population size of 50. (Rank Space can be used as indication)

The following procedure was performed to determine the mean performance of the algorithms on the average penalized cost independent of the population size and the size of the frames. First, in an attempt to normalize the costs of frames of different sizes, the average penalized costs were scaled as follows. A four bay ten story frame has twice the number of spans and five times the number of stories as a two bay two story frame.

Then, a four bay ten story frame is approximately 2x5 = 10 two bay two story frames. Therefore, the penalized costs of the two bay two story frames were scaled by a factor of 10. Similarly, a four bay ten story frame has the same number spans and twice the number of stories as a four bay five story frame. Then, a four bay ten story frame is made of about 1x2 = 2 four bay five story frames, so the penalized costs of the 2 four bay five story frame were scaled by a factor of 2. Scaling as such ensures that the prices of all frames have a similar influence or weight on the mean. Then, for each algorithm, the mean of the average penalized costs is obtained by summing all the average penalized costs associated with this algorithm in Figure 42, Figure 43, and Figure 44 (3 points for each algorithm in each figure), and then divide by the number summed results.

Figure 45 shows the mean of average penalized costs. Rank Selection results in the lowest mean average penalized cost of 64400. Tournament Selection, Rank Space, and Fuss result in mean average penalized costs larger than that of Rank Selection by 9%, 12%, and 39% respectively. Therefore, Rank Selection shows the best performance on the tested frames and population sizes followed by Tournament Selection, Rank Space, and Fuss is last.

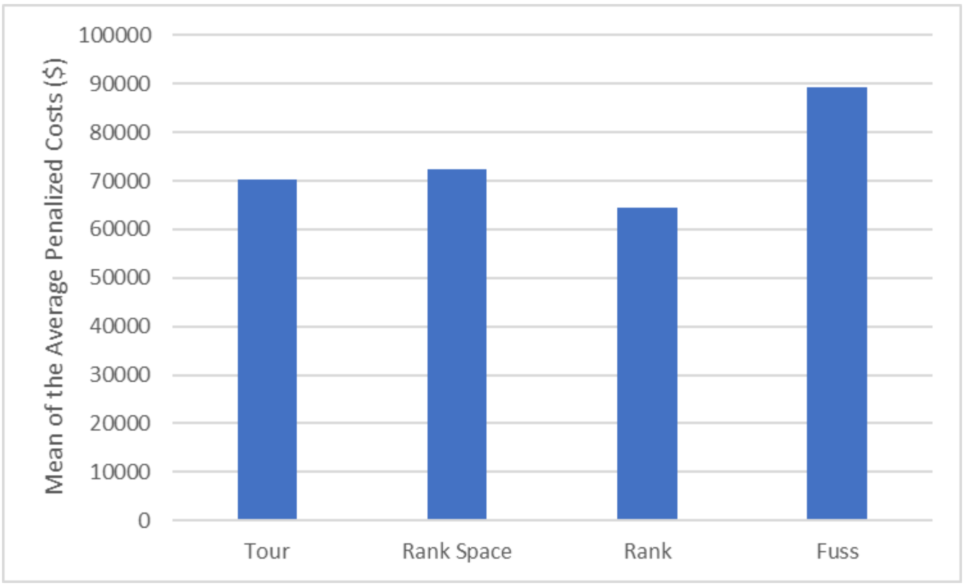


Figure 45 Mean of the average penalized cost of all of the tested diversity maintaining selection algorithms

4.7. Comparing the Performance of the Diversity Maintaining Mechanisms on the Convergence Time

Figure 46, Figure 47, and Figure 48 show the convergence time as a function of the population size for the different GA selection algorithms and for 2D Frames of different sizes. The costs were calculated assuming a mean man-hour unit cost of 50$/man-hour.

Figure 46 shows the convergence time for the two story frame. The algorithms were tested on an Intel 3.0 GHz Octa-Core machine with 16 GB RAM. Rank and tournament selection share similar convergence times. For both algorithms, the convergence time increases from 2 minutes to 6.3 (215% increase) minutes as the population size increases from 50 to 150. On the other hand, Fuss is consistently more time consuming than Rank and Tournament Selection, and Rank Space is the most time consuming algorithm. As the population size increases from 50 to 150, the convergence times of Fuss and Rank Space increase from 2.4 minutes to 7.9 minutes (229% increase), and from 2.5 minutes to 8.9 minutes (256% increase) respectively.

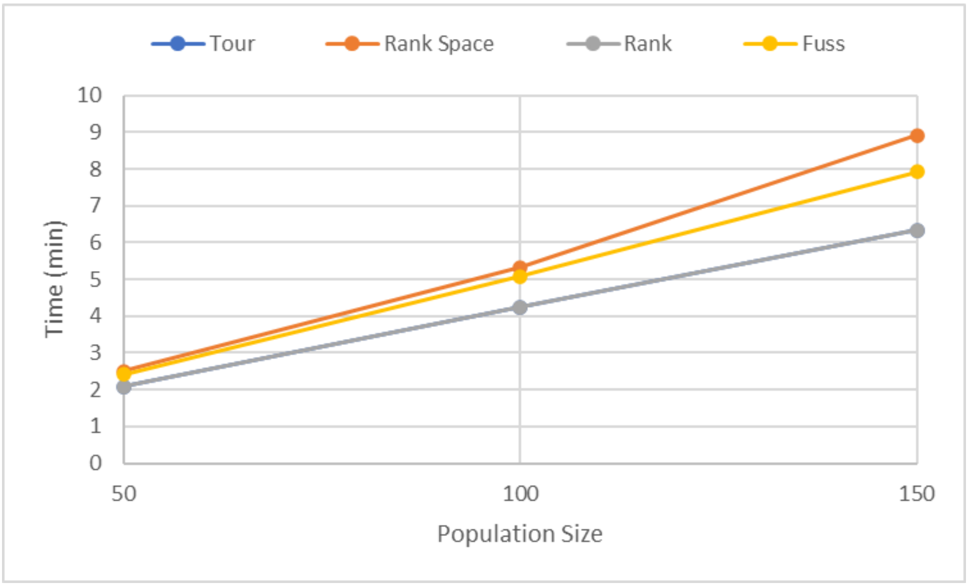


Figure 46 The variation of the convergence time of a two story frame as a function of the population size

Figure 47 shows the convergence time for the five story frame. As the population size increases from 50 to 150, the convergence time of Tournament, Fuss, Rank, and Rank Space increases from 7.5 minutes to 10 minutes (33% increase), from 16.6 minutes to 29.25 minutes (43% increase), from 5 minutes to 35.5 minutes (610% increase), and from 4 minutes to 40 minutes (900% increase) respectively.

It is important to note that Tournament Selection presents the most desirable behavior among the tested algorithms. Tournament Selection started with a relatively low convergence time for a population size of 50, and then this algorithm showed the lowest percentage increase in the convergence time (33% increase) as the population size increased to 150.

Furthermore, the relative performance of the selection algorithms on convergence time is dependent on the population size. It was observed that the when the Population size was 50, the Rank Space algorithm required the least time to converge followed by the Rank algorithm, then Tournament selection, and finally Fuss required the longest time to converge. However, when the population size was increased to 150, the Tournament Selection algorithm required the least time to converge followed by the Fuss, then Rank Selection, and finally Rank Space required the longest time to converge

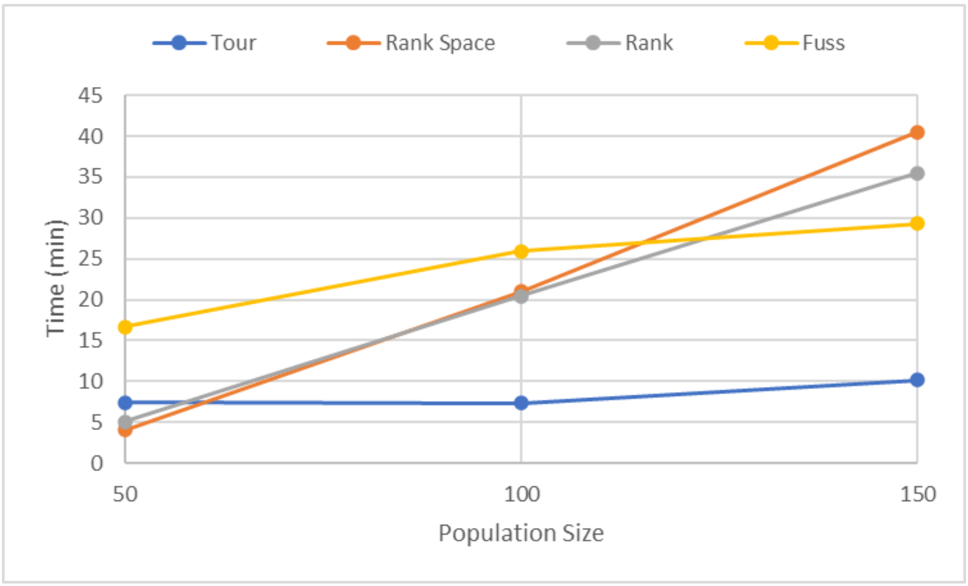


Figure 47 The variation of the convergence time of a five story frame as a function of the population size

Figure 48 shows the convergence time for the ten story frame. As the population size increases from 50 to 150, the convergence time of Tournament, Fuss, Rank, and Rank Space increases from 17 minutes to 21 minutes (23% increase), from 28 minutes to 107 minutes (280% increase), from 6 minutes to 57 (850% increase) minutes, and from 21 minutes to 141 minutes (570% increase) respectively.

Similar to what was observed with the results of the five story frame, tournament Selection presents the most desirable behavior among the tested algorithms. Tournament Selection started with a relatively low convergence time for a population size of 50, and then this algorithm showed the least percentage increase in the convergence time (23% increase) as the population size increased to 150.

Moreover, similar to the results observed for the Five Story frame the relative performance of the selection algorithms on the convergence time is dependent on the population size.

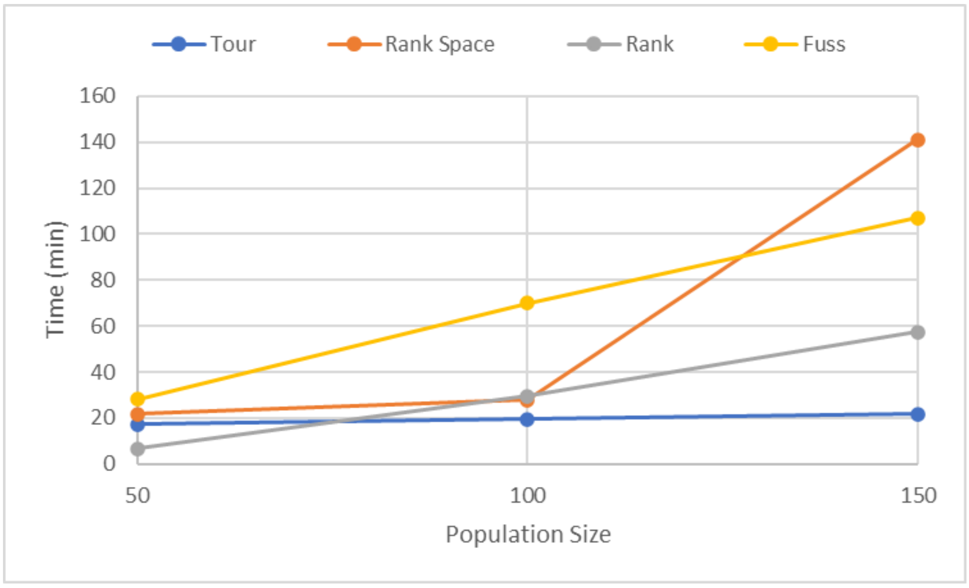


Figure 48 The variation of the convergence time of a ten story frame as a function of the population size

The following can be noted from the results in Figure 46, Figure 47, and Figure 48 collectively. First, Tournament selection was generally the algorithm that required the least convergence time, and it showed the most desirable behavior. For population sizes of 100 and 150, Tournament selection was the most efficient algorithm regardless of the size of the frame. For a population size of 50, Tournament selection required at most 11 minutes more than the most time efficient algorithm to converge. Moreover, Fuss was generally less time efficient than both Rank and Tournament Selection. Fuss did not outperform Tournament Selection in any case, and it only outperformed Rank Selection on the Five story frame optimized using a population size of 150. Furthermore, the Rank Space algorithm is the least time efficient algorithm for a population size of 150 regardless of the size of the frame.

Then, the following procedure was performed to determine the mean performance of the algorithms on the convergence time independent of the population size and the size of the frames. For each algorithm, the mean convergence time is obtained by summing all the convergence times associated with this algorithm in Figure 46, Figure 47, and Figure 48 (3 points for each algorithm in each figure), and then divide by the number summed results.

Figure 49 shows the mean convergence time of each algorithm. The figure shows that Tournament Selection is the most efficient algorithm requiring and average convergence time of 10 minutes. Rank Selection, Rank Space, Fuss require mean convergence times of 18 minutes, 30 minutes, and 34 minutes respectively.

From the results in Figure 45 and Figure 49 it can be deduced that for the tested frames and population sizes, Tournament selection is the most computationally efficient algorithm, and Rank Selection results in the lowest penalized cost. Tournament Selection can obtain penalized costs 10% more expensive than those obtained by Rank Selection, but within half the time needed by Rank Selection. Rank Space and Fuss demonstrate relatively high penalized costs and convergence times compared to Tournament Selection and Rank Selection. Furthermore, Rank Space demonstrates an undesirable increasing relation between the penalized cost and population size. Therefore, Rank Space and Fuss will be discarded.

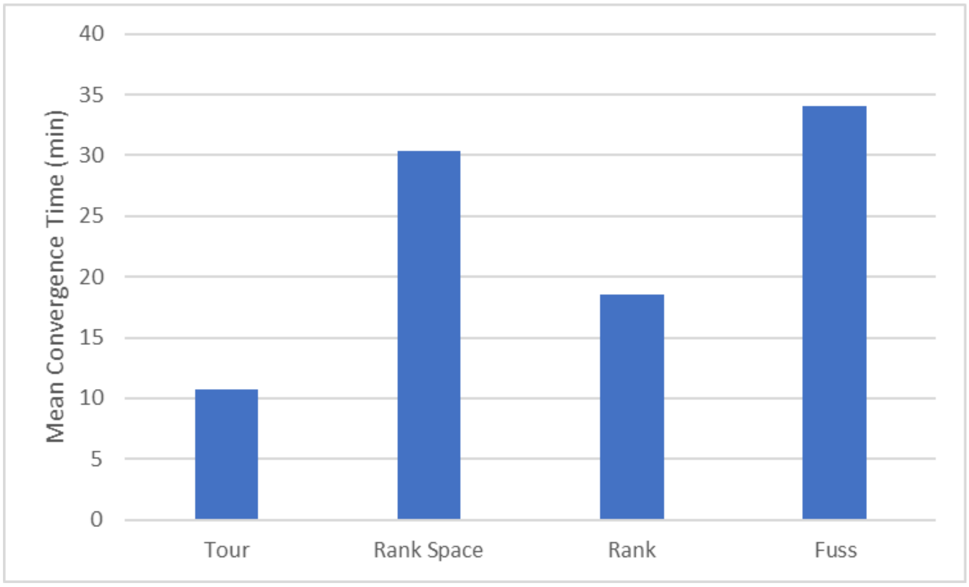


Figure 49 Mean of the convergence times of all of the tested diversity maintaining selection algorithms

4.8. Comparing Tournament Selection and Rank Selection

Figure 50, Figure 51, and Figure 52 show the variation of the Average penalized costs as a function of the population size for frames in Figure 27 optimized using Rank and Tournament Selection.

Figure 50 shows the results of the two story frame. The results show that for both Rank and Tournament Selection, the average penalized cost plateaus for a population size of 150. Increasing the population size to 600 only changed the average penalized costs by 0.4% and 0.1% for Tournament and Rank Selection respectively.

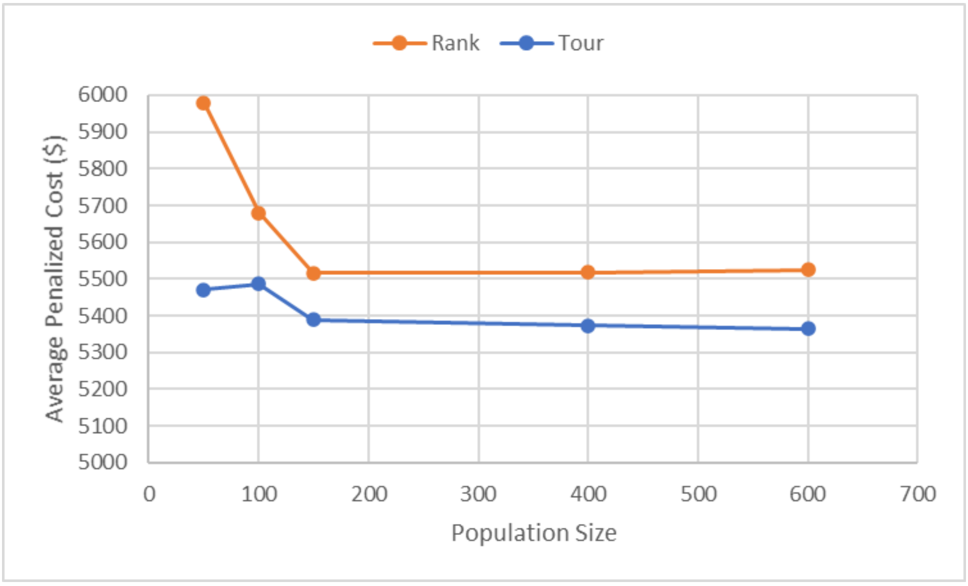


Figure 50 The average penalized cost as a function of the population size for a two story frame for rank and tournament selection

Figure 51 shows the results of the five story frame. For Rank Selection, increasing the population size from 150 to 400, decreases the penalized cost by only 0.2%. The average penalized cost obtained when using Rank Selection is plateaued at a population size of 150, since the increasing the population by 350 individuals resulted in a negligible change. However, for Tournament Selection, increasing the population size from 150 to 400 decreases the penalized cost from $33200 to $31900 (4% decrease), so Tournament Selection did not plateau at a population size of 150. Then, increasing the population size from 400 to 800 decreases the penalized cost from $31900 to $30165 (5.7% decrease). The algorithm plateaus at a population size of 800 since increasing the population size to 1000 only decreases the average penalized cost to $29860 (1% decrease).

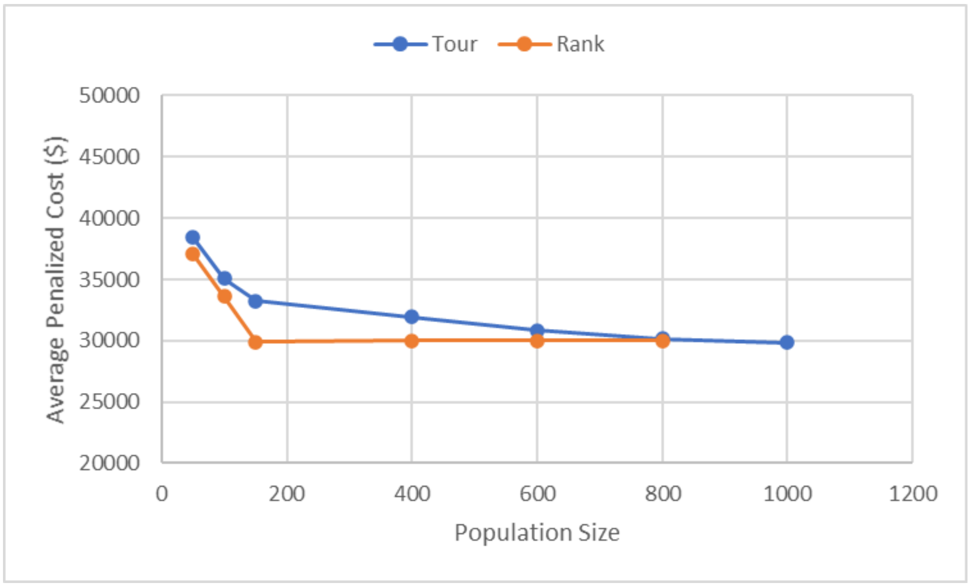


Figure 51 The average penalized cost as a function of the population size for a five story frame for rank and tournament selection

Figure 52 shows the results of the ten story frame. The results show that for both Rank and Tournament Selection, the average penalized cost plateaus for a population size of 600. Increasing the population size from 150 to 600 decreased the average penalized cost from $61282 to $54700 (12 % decrease), and from $68000 to $60100 (13.2% decrease) for Rank and Tournament Selection respectively. Increasing the population size from 600 to 800 only changed the average penalized costs by 1% and 0.73% for Tournament and Rank Selection respectively.

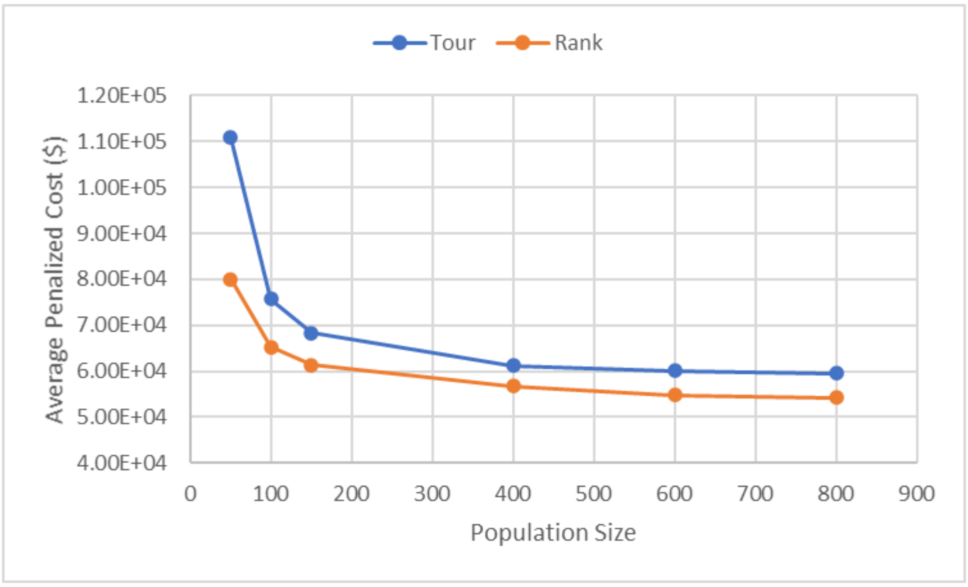


Figure 52 The average penalized cost as a function of the population size for a ten story frame for rank and tournament selection

To further test the results, both the Tournament Selection and the Rank Selection were used to determine the plateau of the penalized cost of the frame in Figure 53. Figure 54 shows the plateau for both algorithms. As the population size increases from 150 to 600 the average penalized cost decreases from $200000 to $171640 (16.5% decrease) and from $222000 to $169320 (30% decrease) for Rank Selection and Tournament Selection respectively. For a population size of 800, Rank Selection and Tournament Selection converged to penalized costs of $171640 and $169320 respectively. Then, when the population size increases to 1000, penalized costs of for Rank Selection and Tournament Selection change to $170400 (0.7% decrease) and $170650 (0.7% increase) respectively.

The maximum change in the penalized cost observed when increasing the population size from 150 until a plateau is reached is 17% for Rank Selection and 30% for Tournament Selection.

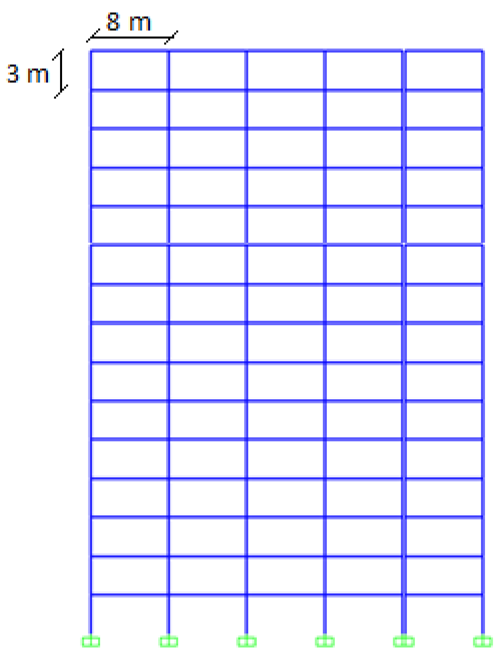


Figure 53 five bay-fifteen story frame

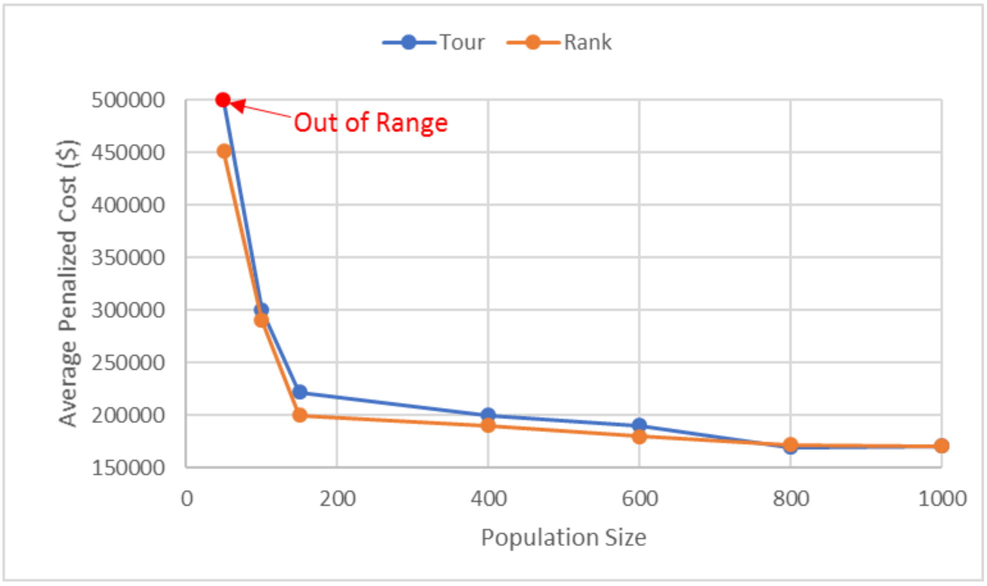


Figure 54 The average penalized cost as a function of the population size for a fifteen story frame for rank and tournament selection

Table 9Table 7, Table 8, and Table 9 Table 7 show the average penalized cost at the plateau, the optimization time of the last increment of the population size, and the cumulative time needed to run the multiple the GA while increasing the population size to reach the plateau. In the best case, Rank Selection outperformed Tournament Selection on the average penalized costs by 10%, but rank required twice to three times convergence time to reach that result in some cases. Therefore, Tournament Selection is the most practical algorithm to use.

Furthermore, Table 8 and Table 9 show that experience with choosing the population size can significantly reduce the time needed to reach the plateau of the average penalized cost. Had the ideal population sizes for the last increment been known from the start, the optimization time could have been reduced by more than a factor of two and a half.

Table 7 Average Penalized Cost at Plateau using Rank and Tournament Selection

	Average Penalized Cost at Plateau ($)			
Frame	Tournament Selection	Rank	Cost of Preliminary Design	% Change Rank Vs Tour
Fifteen Story	169320	171640	-	1.4
Ten Story	60100	54700	65300	9.9
Five Story	30000	30000	-	1.0
Two Story	5390	5515	-	2.3

Table 8 Optimization Time for Last Increment of Population Size using Rank and Tournament Selection

	Optimization Time for Last Increment of Population Size(min)	
Frame	Tournament Selection	Rank
Fifteen Story	125	387
Ten Story	70	156
Five Story	34	35

	Optimization Time for Last Increment of Population Size(min)	
Two Story	6	6

Table 9 Cumulative Optimization Time of different frames using Rank and Tournament Selection

	Cumulative Optimization Time (min)	
Frame	Tournament Selection	Rank
Fifteen Story	284	866
Ten Story	176	373
Five Story	83	60
Two Story	12	16

4.9. The Time Saved by the Weight Matrix Method on GA Optimizations

Figure 55 compares the time needed to complete one GA iteration using the Conventional and Weight Matrix Method to construct the stiffness matrix. The results show that using the weight matrix method results in a 35% improvement in computational efficiency.

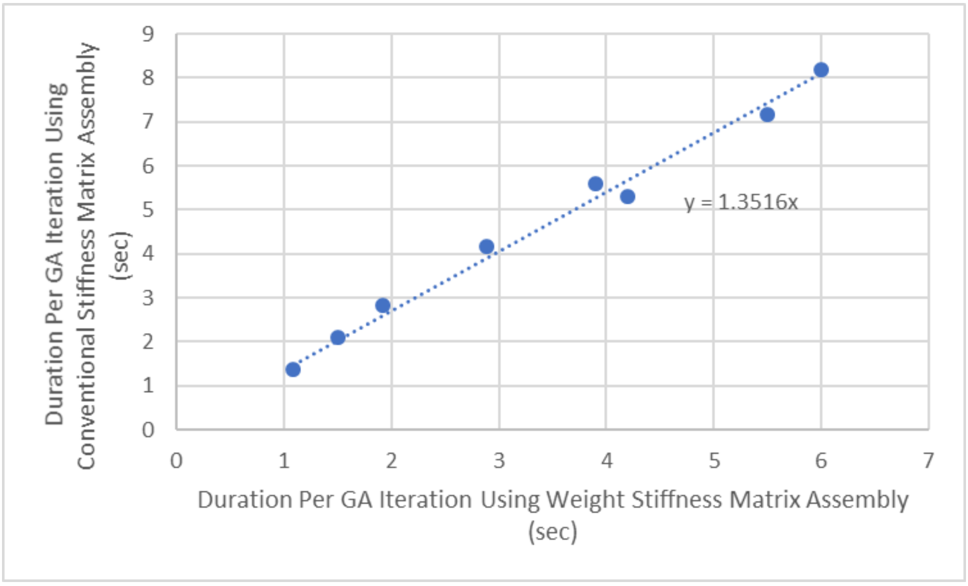


Figure 55 Comparison between the time needed to complete one GA iteration using the Conventional and Weight Matrix Method to construct the stiffness matrix

4.10. Study of the Influence of Buildability Factors on the Total Man Hours and Material Costs of the Frames

In this section, the effect of the buildability factors on the number of man-hours and material costs needed to construct the frame designs is studied. The frames in Figure 27 were analyzed for three cases. In the first cases, the buildability was neglected during the optimization process by setting the labor cost to zero. The second and third cases consider the effect of buildability during the optimization process, but each case uses a different labor cost. In the second case, the labor cost was set to $50 per man-hour. In the third case, the labor cost was set to $100 per man-hour. Each frame was optimized five times for each of the mentioned cases.

The results in Figure 56, Figure 57, Figure 58, and Figure 59 show that the effect of buildability on the material costs and man-hours is insignificant. The results in Figure 56 show that considering buildability with a $50 per man-hour labor cost only dropped the man hours needed to construct the frame by 1.4%. Similarly, the results in

Figure 57 show that considering buildability with a $100 per man-hour labor cost only dropped the man hours needed to construct the frame by 1.3%. Furthermore, the results in Figure 58 and Figure 59 show that considering buildability with a $50 and $100 per man-hour labor cost only change the material costs needed to construct the frame by about 1%.

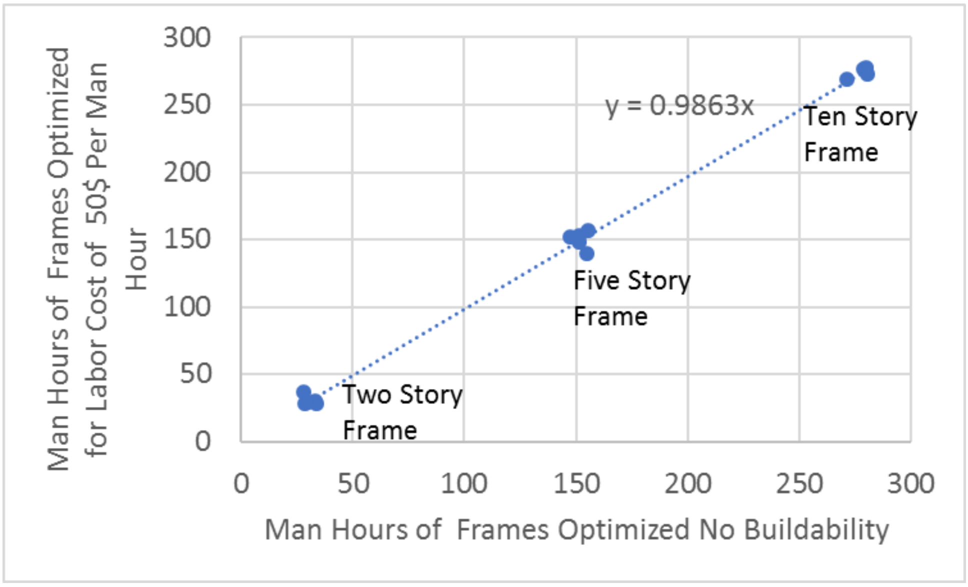


Figure 56 Comparison of man-hours for frames optimized without considering buildability and frames optimized considering buildability with a labor cost of $50 per man hour

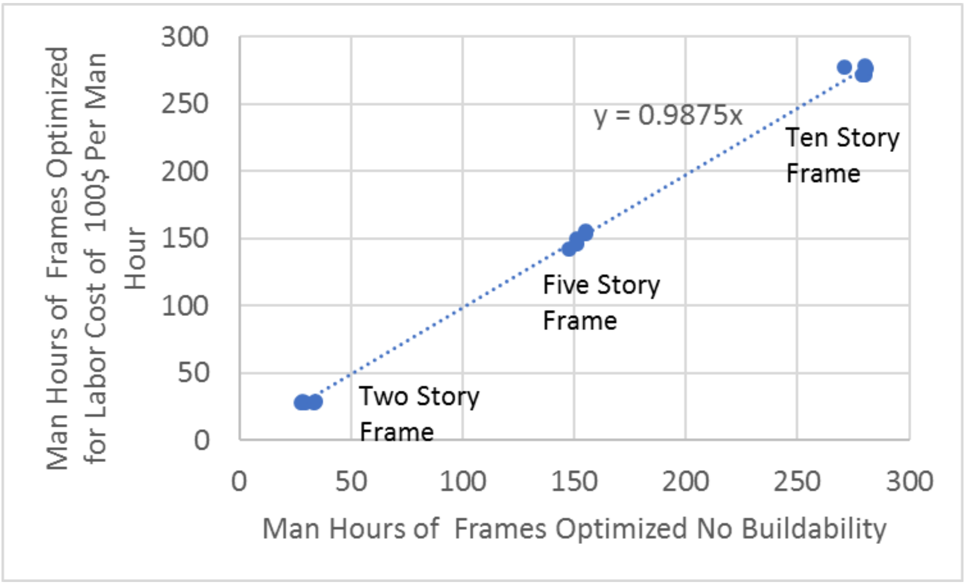


Figure 57 Comparison of man-hours for frames optimized without considering buildability and frames optimized considering buildability with a labor cost of $100 per man hour

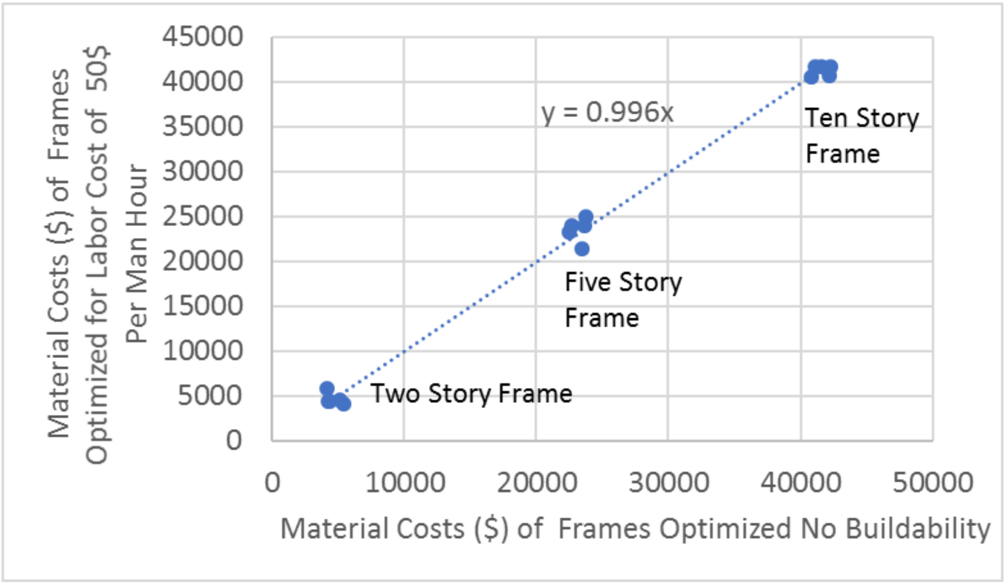


Figure 58 Comparison of material costs for frames optimized without considering buildability and frames optimized considering buildability with a labor cost of $50 per man hour

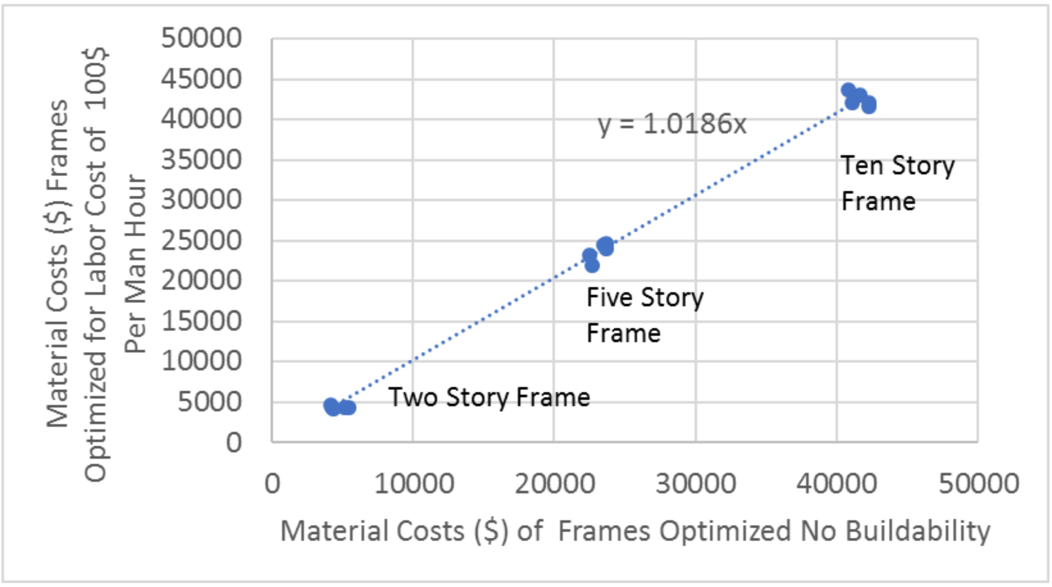


Figure 59 Comparison of material costs for frames optimized without considering buildability and frames optimized considering buildability with a labor cost of $100 per man hour

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

The objective of this research is to produce a computationally efficient framework that is capable of automating the structural design process of special moment resisting frames. To improve the computational efficiency, artificial neural networks were investigated as complete/partial replacements for structural analysis, and a novel algorithm called the Weight Matrix method was suggested to efficiently update the global stiffness matrix of a frame structure. To automate the design process, the Genetic Algorithm was used with constraints that satisfy the requirements of special moment resisting frames, different diversity maintaining algorithms were investigated to mitigate premature convergence, and the effect of buildability factors on the total cost of the frames was studied. The results showed the following:

- The training time of neural networks grows exponentially with the size of the plane frame making them impractical replacements for structural analysis.
- For small frames, using neural networks (without subdividing the frame) is at best 20 times more computationally efficient than the conventional method of constructing the global stiffness matrices of frames.
- The prediction time of neural networks used (without subdividing the frame) to construct the stiffness matrix of a plane frame structure has a

time complexity of O(n^2), so neural networks lose their utility as the size of the frame structure increases.

- When frames are subdivided to sub-frames, artificial neural networks can be used to predict the stiffness matrices of the sub-frames, and then the stiffness matrices of the sub-frames can be used to predict the global stiffness matrix of the complete plane frame. This method evades the need for training neural networks for large frames, but the savings on the prediction time of this method compared to the conventional method of constructing the stiffness matrix are dependent on the size of the sub-frames. Larger subframes require training larger neural networks, but can result in faster predictions.
- Algorithmic methods that rely on vector operations such as the Vectorized Stiffness algorithm are more practical and computationally efficient than ANNs in predicting the global stiffness matrices of plane frames. Unlike the ANN methods, this algorithmic method does not require balancing between training time and prediction time.
- A novel Weight Matrix method was suggested and shown to be 4.3 times more computationally efficient than the Vectorized Stiffness Matrix method and 30 times more efficient than the conventional method for constructing the global stiffness matrices of plane frames.
- Different diversity maintaining algorithms were studied on frames of different sizes. The Rank Space and Fuss algorithms resulted in high-penalized costs and convergence times compared to Rank selection and Tournament Selection. Rank Selection can outperform Tournament

Selection on the penalized cost by 10% in some cases, but Tournament Selection requires half to one-third the convergence time for large frames.

- The maximum change in the penalized cost observed when increasing the population size from 150 until a plateau is reached is 17% for Rank Selection and 30% for Tournament Selection.
- The genetic algorithm resulted in a 16% reduction in the cost of the preliminary design of the ten story frame
- Replacing the Conventional Method of constructing the stiffness matrices of frame structures with the Weight Matrix method resulted in a 35% decrease in the convergence time of the Genetic Algorithm.
- Studying the effect of buildability on the material cost and man-hours needed to construct plane frames showed that considering buildability factors has a negligible effect of about 1% on the man hours and material costs.

5.2. Future Work

Although this research answers many questions regarding increasing the computational efficiency of structural analysis of plane frames and optimizing of their costs, some subjects can be investigated in future works.

- This research is confined to plane frame structures, so a similar investigation can be conducted for 3D frames.

- This research was concerned with reinforced concrete moment resisting frames. Investigations need to be done for other lateral load resisting systems (Steel moment resisting frame systems, Shear wall systems, Combined Systems…)
- The computational efficiency of the Weight Matrix Method should be investigated for 3D frame elements, shell elements, and solid elements.
- This work can be repeated if promising novel diversity maintaining mechanisms emerge in the literature.

REFERENCES

ACI Committee 318. (2014). Building Code Requirements for Structural Concrete. In *American Concrete Institute*.

Afaq Ahmad. (2017). *Reinforced Concrete (RC) Structures Analysis and Assessment with Artificial Neural Networks (ANNs)*. Heriot-Watt University.

American Society of Civil Elilgineers. (2013). ASCE 7-10. In *American Society of Civil Engineers, Minimum Design Loads for Buildings and Other Structures (ASCE 7-10).* https://doi.org/http://dx.doi.org/10.1061/9780784412916

Babaei, M., & Mollayi, M. (2016). MULTI-OBJECTIVE OPTIMIZATION OF REINFORCED CONCRETE FRAMES USING NSGA-II ALGORITHM. *Engineering Structures and Technologies*. https://doi.org/10.3846/2029882x.2016.1250230

Baldi, P. (2018). The inner and outer approaches to the design of recursive neural architectures. *Data Mining and Knowledge Discovery*. https://doi.org/10.1007/s10618-017-0531-0

Balling, R. J. (1991). Optimal steel frame design by simulated annealing. *Journal of Structural Engineering (United States)*. https://doi.org/10.1061/(ASCE)0733-9445(1991)117:6(1780)

Basheer, I. A., & Hajmeer, M. (2000). Artificial neural networks: Fundamentals, computing, design, and application. *Journal of Microbiological Methods*. https://doi.org/10.1016/S0167-7012(00)00201-3

Camp, C. V. (2007). Design of space trusses using big bang-big crunch optimization. *Journal of Structural Engineering*. https://doi.org/10.1061/(ASCE)0733-9445(2007)133:7(999)

Chen, L. (2011). *Programming of Finite Element Methods in Matlab*.

Chipperfield, K., Vance, J. M., & Fischer, A. (2006). Fast meshless reanalysis using combined approximations, preconditioned conjugate gradient, and Taylor series. *AIAA Journal*. https://doi.org/10.2514/1.14170

Cuvelier, F., Japhet, C., & Scarella, G. (2013). An efficient way to perform the assembly of finite element matrices in Matlab and Octave. *CoRR*, *abs/1305.3*. Retrieved from http://arxiv.org/abs/1305.3122

Cuvelier, F., Japhet, C., & Scarella, G. (2016). An efficient way to assemble finite element matrices in vector languages. *BIT Numerical Mathematics*. https://doi.org/10.1007/s10543-015-0587-4

Davis, T. A. (2011). Direct Methods for Sparse Linear Systems. In *Direct Methods for Sparse Linear Systems*. https://doi.org/10.1137/1.9780898718881

Esfandiari, M. J., Urgessa, G. S., Sheikholarefin, S., & Dehghan Manshadi, S. H. (2018). Optimization of reinforced concrete frames subjected to historical time-history loadings using DMPSO algorithm. *Structural and Multidisciplinary Optimization*. https://doi.org/10.1007/s00158-018-2027-y

Esfandiari, M. J., Urgessa, G. S., Sheikholarefin, S., & Manshadi, S. H. D. (2018). Optimum design of 3D reinforced concrete frames using DMPSO algorithm. *Advances in Engineering Software*. https://doi.org/10.1016/j.advengsoft.2017.09.007

Ghosh, S., Das, N., & Nasipuri, M. (2019). Reshaping inputs for convolutional neural network: Some common and uncommon methods. *Pattern Recognition*. https://doi.org/10.1016/j.patcog.2019.04.009

Golub, G. H., & Van Loan, C. F. (2013). Matrix Computations (4th Ed.). In *Johns*

Hopkins University Press.

Govindaraj, V., & Ramasamy, J. V. (2005). Optimum detailed design of reinforced concrete continuous beams using Genetic Algorithms. *Computers and Structures*. https://doi.org/10.1016/j.compstruc.2005.09.001

Govindaraj, V., & Ramasamy, J. V. (2007). Optimum detailed design of reinforced concrete frames using genetic algorithms. *Engineering Optimization*. https://doi.org/10.1080/03052150601180767

Haykin, S. (2008). Neural Networks and Learning Machines. In *Pearson Prentice Hall New Jersey USA 936 pLinks*. https://doi.org/978-0131471399

Heap, R. (2013). *Real-time visualization of finite element models using surrogate modeling methods*. Brigham Young University.

Herrera, F., Lozano, M., & Verdegay, J. L. (1998). Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis. *Artificial Intelligence Review*. https://doi.org/10.1023/A:1006504901164

Hoos, H., & Stutzle, T. (2004). *Stocastic Local Search: Foundations and Applications*. Morgan Kaufmann.

Hutter, M. (2002). Fitness uniform selection to preserve genetic diversity. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002*. https://doi.org/10.1109/CEC.2002.1007025

Jahjouh, M. M., Arafa, M. H., & Alqedra, M. A. (2013). Artificial Bee Colony (ABC) algorithm in the design optimization of RC continuous beams. *Structural and Multidisciplinary Optimization*. https://doi.org/10.1007/s00158-013-0884-y

Jarkas, A. M. (2005). *An investigation into the influence of buildability factors on productivity of insitu reinforced concrete construction*. University of Dundee.

Jarkas, A. M. (2010a). THE IMPACTS OF BUILDABILITY FACTORS ON FORMWORK LABOUR PRODUCTIVITY OF COLUMNS. *JOURNAL OF CIVIL ENGINEERING AND MANAGEMENT*. https://doi.org/10.3846/jcem.2010.53

Jarkas, A. M. (2010b). The influence of buildability factors on rebar fixing labour productivity of beams. *Construction Management and Economics*. https://doi.org/10.1080/01446191003703482

Jarkas, A. M. (2011). Buildability factors that influence micro-level formwork labour productivity of beams in building floors. *Journal of Construction in Developing Countries*.

Jarkas, A. M. (2012). Influence of Buildability Factors on Rebar Installation Labor Productivity of Columns. *Journal of Construction Engineering and Management*. https://doi.org/10.1061/(asce)co.1943-7862.0000425

Kao, C. S., & Yeh, I. C. (2016). Using neural networks to integrate structural analysis package and optimization package. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-015-1878-z

Kassimali, A. (2011). *Matrix Analysis of Structures*. Cengage Learning.

Kaveh, A., & Sabzi, O. (2012). Optimal design of reinforced concrete frames Using big bang-big crunch algorithm. *International Journal of Civil Engineering*.

Kotsovou, G. M., Cotsovos, D. M., & Lagaros, N. D. (2017). Assessment of RC exterior beam-column Joints based on artificial neural networks and other methods. *Engineering Structures*. https://doi.org/10.1016/j.engstruct.2017.04.048

Kwak, H. G., & Kim, J. (2008). Optimum design of reinforced concrete plane frames based on predetermined section database. *CAD Computer Aided Design*.

https://doi.org/10.1016/j.cad.2007.11.009

Lankhorst, M. M. (1996). *Genetic algorithms in data analysis*. University of Groningen.

Lee, C., & Ahn, J. (2003). Flexural Design of Reinforced Concrete Frames by Genetic Algorithm. *Journal of Structural Engineering*. https://doi.org/10.1061/(asce)0733-9445(2003)129:6(762)

Leng, J. (2016). Optimization techniques for structural design of cold-formed steel structures. In *Recent Trends in Cold-Formed Steel Construction*. https://doi.org/10.1016/B978-0-08-100160-8.00006-2

Li, G., Jia, S., Yu, D. H., & Li, H. N. (2018). Woodbury approximation method for structural nonlinear analysis. *Journal of Engineering Mechanics*. https://doi.org/10.1061/(ASCE)EM.1943-7889.0001464

Loja, M. A. R., Barbosa, J. I., & Mota Soares, C. M. (2014). Analysis of sandwich beam structures using kriging based higher order models. *Composite Structures*. https://doi.org/10.1016/j.compstruct.2014.08.019

Lowen, R., & Verschoren, A. (2008). Foundations of Generic Optimization: Applications of Fuzzy Control, Genetic Algorithms and Neural Networks. In *Mathematical Modelling: Theory and Applications*.

McGuire, W., Gallagher, R. H., & Saunders, H. (1982). Matrix Structural Analysis. *Journal of Mechanical Design*. https://doi.org/10.1115/1.3256379

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

Moehle, J. P. (2014). *Seismic Design of Reinforced Concrete Buildings* (1st ed.). McGraw Hill Education.

Paya-Zaforteza, I., Yepes, V., Hospitaler, A., & González-Vidosa, F. (2009). CO2-

optimization of reinforced concrete frames by simulated annealing. *Engineering Structures*. https://doi.org/10.1016/j.engstruct.2009.02.034

Paya, I., Yepes, V., González-Vidosa, F., & Hospitaler, A. (2008). Multiobjective optimization of concrete frames by simulated annealing. *Computer-Aided Civil and Infrastructure Engineering*. https://doi.org/10.1111/j.1467-8667.2008.00561.x

Perez-Ramirez, C. A., Amezquita-Sanchez, J. P., Valtierra-Rodriguez, M., Adeli, H., Dominguez-Gonzalez, A., & Romero-Troncoso, R. J. (2019). Recurrent neural network model with Bayesian training and mutual information for response prediction of large buildings. *Engineering Structures*. https://doi.org/10.1016/j.engstruct.2018.10.065

Pilkington, J. L., Preston, C., & Gomes, R. L. (2014). Comparison of response surface methodology (RSM) and artificial neural networks (ANN) towards efficient extraction of artemisinin from Artemisia annua. *Industrial Crops and Products*. https://doi.org/10.1016/j.indcrop.2014.03.016

Pray, R. (2017). *2017 National Construction Estimator* (65th ed.). California: Craftsman Book Company.

Rafiq, M. Y., Bugmann, G., & Easterbrook, D. J. (2001). Neural network design for engineering applications. *Computers and Structures*. https://doi.org/10.1016/S0045-7949(01)00039-6

Reyes, E. N., & Steidley, C. (1998). Optimization using simulated annealing. *Northcon - Conference Record*.

Rozenberg, G., Back, T., & Kok, J. N. (2012). Handbook of Natural Computing. In *Handbook of Natural Computing*. https://doi.org/10.1007/978-3-540-92910-9

Sareni, B., & Krähenbühl, L. (1998). Fitness sharing and niching methods revisited.

IEEE Transactions on Evolutionary Computation. https://doi.org/10.1109/4235.735432

Slobbe, G. (2015). *OPTIMISATION OF REINFORCED CONCRETE STRUCTURES*. Delft University of Technology.

Sudholt, D. (2018). The Benefits of Population Diversity in Evolutionary Algorithms: {A} Survey of Rigorous Runtime Analyses. *CoRR*, *abs/1801.1*. Retrieved from http://arxiv.org/abs/1801.10087

Sun, R., Liu, D., Xu, T., Zhang, H., & Zuo, W. (2014). New adaptive technique of kirsch method for structural reanalysis. *AIAA Journal*. https://doi.org/10.2514/1.J051597

Vicario, G., Craparotta, G., & Pistone, G. (2016). Meta-models in Computer Experiments: Kriging versus Artificial Neural Networks. *Quality and Reliability Engineering International*. https://doi.org/10.1002/qre.2026

Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. *Advances in Neural Information Processing Systems*.

Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft Computing*. https://doi.org/10.1007/s00500-016-2474-6

Wang, F.-S., & Chen, L.-H. (2013). Heuristic Optimization. In *Encyclopedia of Systems Biology*. https://doi.org/10.1007/978-1-4419-9863-7_411

Waszczysznk, Z. (1999). *Neural Networks inthe Analysis and Design of Structures*. CISM International Centre for Mechanical Sciences.

Winston, P. H. (1992). *Artificial Intelligence (3rd Ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Wu, R. T., & Jahanshahi, M. R. (2019). Deep convolutional neural network for structural dynamic response estimation and system identification. *Journal of Engineering Mechanics*. https://doi.org/10.1061/(ASCE) EM.1943-7889.0001556

Xie, H., & Zhang, M. (2013). Tuning Selection Pressure in Tournament Selection. *IEEE Transactions on Evolutionary Computation*.

Zhou, Y. (2012). *Study on genetic algorithm improvement and application* (Vol. 2). WORCESTER POLYTECHNIC INSTITUTE.

Zuo, W., Bai, J., & Yu, J. (2016). Sensitivity reanalysis of static displacement using Taylor series expansion and combined approximate method. *Structural and Multidisciplinary Optimization*. https://doi.org/10.1007/s00158-015-1368-z