A Two-Phase Genetic Algorithm for Simultaneous Dimension, Topology, and Shape Optimization of Free-Form Steel Space-Frame Roof Structures

Thesis

Presented in Partial Fulfillment of the Requirements for the Master of Science Degree in the Graduate School of The Ohio State University

Margaret E. Kociecki
B.S.C.E.
Civil Engineering
The Ohio State University
2012

Thesis Committee:
Dr. Hojjat Adeli
Dr. Mark McCord
Dr. Ethan Kubatko
Copyright by
Margaret E. Kociecki
2012
Abstract

The objective of this work is to study the effects of geometry on structural performance of free-form steel space-frame roof structures and to optimize the structures without compromising overall architectural forms. Minimum weight optimization is performed to better study the effects of geometric alterations on overall structural performance. The intent is to achieve a strong optimum shape with superior load-carrying capacity allowing for the smallest and lightest structural members to be used. A two-phase genetic algorithm (GA) is developed to perform minimum weight design of the roof structures which consist of rectangular hollow structural sections (HSS). The new methodology is applied to two example roof structures subjected to the AISC LRFD code (AISC, 2005) and ASCE-10 snow, wind, and seismic loading (ASCE, 2010). Both are train station roofs for the Ottawa Light Rail Transit (OLRT) system to be built in Ottawa, Canada, in 2018. The structures are made up of a diamond-shaped grid pattern and their members are subjected to torsion in addition to bending and axial forces.

The GA was developed to perform simultaneous dimension, topology, and shape optimization and resulted in final designs which are 22% and 24% lighter than the initial designs created in a design office for the two roof structures. This global optimum solution was achieved in less than 19 hours on a standard workstation machine with a 2.83 GHZ dual core processor, a relatively short amount of time considering the complexity of both the structures and the optimization problem.
Acknowledgements

It is a pleasure to thank those who made this thesis possible:

• Dr. Hojjat Adeli, my adviser, for encouragement and guidance as well as my committee members, Dr. Mark McCord and Dr. Ethan Kubatko,

• The Ohio State University FE Program for providing me with a GTA position,

• John George and fellow structural engineers at URS Corporation for getting me acquainted with the OLRT project, helping with initial conception of the topic, and providing continued support,

• The City of Ottawa for allowing me to use their train stations,

• Busby, Perkins + Will for providing unique architectural designs,

• Computers and Structures, Inc. and MathWorks for providing the software necessary to implement my algorithm,

• My colleagues and staff in the OSU Civil Engineering Department,

• And my family, John and Bonny Kociecki, and Andrew Martin.
Vita

2005-2010.................................B.S. Civil Engineering, The Ohio State University

2011.................................Structural Engineering Intern, URS Corp.

2010-2012.................................Graduate Teaching Associate, First-Year Engineering Program, The Ohio State University

Fields of Study

Major Field: Civil Engineering

Specialization: Structural Engineering
# Table of Contents

Chapter 1: Introduction and Literature Review ................................................................. 1  
  1.1 Introduction and Problem Definition ...................................................................... 1  
  1.2 Literature Review ............................................................................................... 5  
    1.2.a Classical Optimization Methods ................................................................. 5  
    1.2.b Heuristic Optimization Methods ................................................................. 8  

Chapter 2: Genetic Algorithm Formulation and Dimension Optimization ...................... 17  
  2.1 Problem Definition ............................................................................................. 17  
  2.2 Optimization Problem Formulation using a Genetic Algorithm .......................... 23  
  2.3 Examples .......................................................................................................... 34  
    2.3.a Example 1 .................................................................................................. 34  
    2.3.b Example 2 .................................................................................................. 48  

Chapter 3: Topology Optimization ..................................................................................... 58  
  3.1 Problem Definition ............................................................................................. 58  
  3.2 Optimization Problem Formulation using a Genetic Algorithm .......................... 59  
  3.3 Examples .......................................................................................................... 64  
    3.3.a Example 1 .................................................................................................. 64  
    3.3.b Example 2 .................................................................................................. 68  

Chapter 4: Shape Optimization ......................................................................................... 74  
  4.1 Problem Definition ............................................................................................. 74  
  4.2 Optimization Problem Formulation using a Genetic Algorithm .......................... 75  
    4.2.a Method 1 Implicit Variation of Vertical Coordinates ................................. 77  
    4.2.b Method 2 Explicit Inclusion of Vertical Coordinates .................................... 79  
  4.3 Examples .......................................................................................................... 82  
    4.3.a Example 1 .................................................................................................. 82
4.3.b Example 2.................................................................................................................................92

Chapter 5: Conclusion and Future Work..........................................................................................109

References .........................................................................................................................................111

Appendix: MATLAB Code ..............................................................................................................119
List of Tables

Table 1: Final optimum design for Station 1 ................................................................. 48
Table 2: Final optimum design for Station 2 ................................................................. 57
Table 3: Final optimum design for Station 1 ................................................................. 67
Table 4: Final optimum design for Station 2 ................................................................. 73
Table 5: Final optimum design for Station 1 ................................................................. 91
Table 6: Final optimum design for Station 2 ............................................................... 107
List of Figures

Figure 1: Station 1 of the OLRT system (http://www.ottawalightrail.ca/en/) ................................ 3
Figure 2: Station 2 of the OLRT system (http://www.ottawalightrail.ca/en/) ............................... 4
Figure 3: 3D Perspective view of Station 1 .................................................................................. 17
Figure 4: Top view of Station 1 ................................................................................................. 18
Figure 5: Side view, along the length, of Station 1 ...................................................................... 18
Figure 6: Side view, along the width, of Station 1 and wind loading ........................................ 18
Figure 7: Typical cross-section of roof members ....................................................................... 19
Figure 8: Initial roof design regions of Station 1 ....................................................................... 19
Figure 9: Initial column design regions of Station 1 ................................................................... 20
Figure 10: Normal probability distribution for \( t = 3/8 \) inches ..................................................... 26
Figure 11: Example variable string representing a solution with six roof regions and one column region .................................................................................................................. 27
Figure 12: Example of uniform crossover operation .................................................................. 28
Figure 13: Flowchart for dimension optimization phase one ..................................................... 31
Figure 14: Flowchart for GA operations ..................................................................................... 32
Figure 15: Flowchart for phase two of the two-phase GA ......................................................... 33
Figure 16: Convergence curves for various ratios of objective and constraint functions .......... 36
Figure 17: The percentages of overstressed members in the final solution for various ratios of objective and constraint functions ................................................................. 37
Figure 18: Convergence curves comparing various overstressed penalty coefficients for phase one design ................................................................................................................ 39
Figure 19: Percentages of overstressed members in the final design of phase one for various overstressed penalty coefficients .................................................................................. 40
Figure 20: Convergence curves comparing various overstressed penalty coefficients for phase two design ................................................................................................................. 41
Figure 21: Convergence curves comparing various values for percent zeros in the mask string.. 42
Figure 22: Percentages of overstressed members in the final design for various percent zeros in the mask string................................................................. 43

Figure 23: Weight convergence curves using various probability of mutation values ................. 44

Figure 24: Fitness convergence curves using various probability of mutation values ............... 45

Figure 25: Roof design phase two (1-6 initial and added *) regions of Station 1....................... 46

Figure 26: Column design phase two (7-11 initial and added *) regions of Station 1................. 46

Figure 27: Convergence curve for two-phase dimension optimization for Station 1................. 47

Figure 28: 3D Perspective view of Station 2 ............................................................................. 49

Figure 29: Top view of Station 2 ......................................................................................... 50

Figure 30: Side view, along the length, of Station 2 ............................................................... 50

Figure 31: Side view, along the width, of Station 2 and wind loading..................................... 51

Figure 32: Initial roof regions of Station 2 ............................................................................. 52

Figure 33: Initial column regions of Station 2 ........................................................................ 52

Figure 34: Roof design phase two (1-6 initial and added *) regions of Station 2.................... 54

Figure 35: Column design phase two (7-12 initial and added *) regions of Station 2............ 55

Figure 36: Convergence curve for two-phase dimension optimization for Station 2............ 56

Figure 37: Diamond grid element with frame and cross members........................................ 58

Figure 38: Example string representing a solution with four roof regions, one column region, and two out of five possible cross members......................................................... 60

Figure 39: Flowchart Topology optimization phase one ....................................................... 63

Figure 40: Roof design final phase one (1-6) and phase two (*) regions of Station 1.......... 64

Figure 41: Column design final phase one (7-11) and phase two (*) regions of Station 1....... 65

Figure 42: Convergence curve for two-phase topology optimization for Station 1.............. 66

Figure 43: Fitness convergence for optimization sub-problem with region re-assignments every three iterations ............................................................................. 69

Figure 44: Roof design final phase one (1-5) and phase two (*) regions of Station 2......... 70

Figure 45: Column design final phase one (6-11) and phase two (*) regions of Station 2...... 71

Figure 46: Convergence curve for two-phase dimension and topology optimization for Station 2........................................................................................................ 72
Figure 47: Top view of Station 1 showing primary and secondary arches ................................. 74

Figure 48: Example string representing a solution with four roof regions, one column regions, three out of five possible cross members, and 2 4-bit shape optimization variables .......................... 76

Figure 49: Example of the one-point crossover operation for two 4-bit design variable .............. 77

Figure 50: Example arch for shape optimization methods 1 and 2 .................................................. 78

Figure 51: Flowchart for Shape optimization phase one .................................................................. 81

Figure 52: Primary arch showing third-order polynomial line and boundary conditions .............. 83

Figure 53: Upper and lower limits for the primary arch using minimum and maximum values for design variables A and ΔC ........................................................................................................... 85

Figure 54: Roof design final phase one (1-6) and phase two (*) regions of Station 1 ................... 86

Figure 55: Column design final phase one (7-11) and phase two (*) regions of Station 1 ............ 87

Figure 56: 3D perspective view of the final optimal shape for Station 1 ........................................ 87

Figure 57: Side view of the final optimal shape (shown in black) with the initial shape (shown in blue) for Station 1 ......................................................................................................................... 87

Figure 58: Shape design evolution for Station 1 ............................................................................... 88

Figure 59: Convergence curve for two-phase shape optimization for Station 1 ............................ 90

Figure 60: Top view of Station 2 including coordinate system, primary arches 1 through 15 noted in squares, secondary arches, design joints, and fixed joints ................................................................. 93

Figure 61: An example primary arch showing undesirable effects of c violating requirements, c≤1.0 and cH≥1.0 ..................................................................................................................................................... 93

Figure 62: An example primary arch, arch 7, showing a third-order polynomial curve passing through fixed joints and design joints .................................................................................................................. 94

Figure 63: An example primary arch, arch 13, showing a third-order polynomial curve passing through fixed joints and design joints ............................................................................................................. 96

Figure 64: Upper and lower limits for an example primary arch, arch 7, using minimum and maximum values for design variables H and c ........................................................................................................ 97

Figure 65: Upper and lower limits for an example primary arch, arch 13, using minimum and maximum values for design variables H and c ............................................................................................................ 97

Figure 66: 3D perspective and side views of Station 2 presenting the effects of design variable c .................................................................................................................................................... 98

Figure 67: Roof design final phase one (1-6) and phase two (*) regions of Station 2 ...................... 99

Figure 68: Column design final phase one (7-12) and phase two (*) regions of Station 2 .......... 100
Figure 69: 3D perspective view of the final optimal shape showing A) the steel frame only and B) the steel frame with wood panels for Station 2 ................................................................. 101

Figure 70: Side view along the length of Station 2 of the final optimal shape (shown in black) showing A) the steel frame with the initial shape (shown in blue) and B) the steel frame with wood panels ........................................................................................................... 102

Figure 71: Side view along the width of Station 2 of the final optimal shape (shown in black) showing A) the steel frame with the initial shape (shown in blue) and B) the steel frame with wood panels ............................................................................................................ 103

Figure 72: Shape Design Evolution for Station 2 .................................................................................. 104

Figure 73: Convergence curve for two-phase shape optimization for Station 2 .......................... 106
Chapter 1: Introduction and Literature Review

1.1 Introduction and Problem Definition

The objective of this work is to study the effects of geometry on structural performance of free-form steel space-frame roof structures and to optimize the structures without compromising overall architectural forms. Various characteristics of a structure including cross-sectional dimensions, topology, shape, and material can be optimized to improve its overall performance. The goal of the optimization can be a) minimizing the weight of the structure, b) minimizing the cost of the structure, c) minimizing the response such as drift in a high-rise building structure or average deflection of members, or a combination of these. Optimization can be done to create new designs or to improve existing designs.

Cross-sectional dimension optimization is used to minimize the weight or cost of the structure assuming a fixed geometry. In this case the design variables are member shapes and cross-sectional dimensions or areas. These are typically discrete variables, especially in the case of steel structures, due to limited manufactured sizes. The goal of topology optimization is to minimize the number of structural members or amount of material by optimizing the arrangement of members or distribution of material. In this case the design variables are the existence or non-existence of members or material distribution. The former is a discrete variable while the latter is continuous. In shape optimization the geometry of the structure is optimized using nodal coordinates as design variables which can be either discrete or continuous. Material optimization is performed to maximize strength or minimize weight of the structure by optimizing the material selection, a discrete variable chosen from a database. In all cases design constraints include displacement, stiffness, stress and buckling constraints.
Multiple characteristics of a structure can be optimized either individually or simultaneously. When performed simultaneously, the optimization is referred to as multi-objective optimization which is most often a mixed-variable problem, meaning that both discrete and continuous variables are involved. Dimension optimization is typically performed simultaneously with shape or topology optimization. As the number and type of variables or optimization schemes increases, computation time significantly increases in large structural engineering problems. This work includes simultaneous dimension, topology, and shape optimization of complex free-form structures. Minimum weight optimization is performed to better study the effects of geometric alterations on overall structural performance. The aim is to achieve a strong shape with superior load-carrying capacity allowing for the smallest and lightest structural members to be used.

Methods of structural optimization can be classified into two categories: classical search methods and heuristic search methods. Classical search methods use mathematical programming such as linear programming, nonlinear programming, general geometric programming, and optimality criteria methods. When used to solve large structural engineering problems, these methods do not usually produce a global optimal solution and require large amounts of gradient information which is either too cumbersome or impossible to calculate.

Most heuristic search methods used for structural design problems are inspired by nature and include genetic algorithms (GA) based on the Darwinian survival-of-the-fittest theory and particle swarm optimization (PSO) based on flocking of birds or movement of ants in a colony. These methods tend to be better at finding a global optimal solution and often produce a population of near optimal solutions. This allows the designer to consider objectives which cannot be or are not modeled. For example, if the objective is to minimize cost, then the near
minimum cost designs can be considered by the designer for additional unmodelled aesthetics and constructability objectives. The main disadvantage of heuristic search methods is computation time.

A two-phase genetic algorithm (GA) is developed for this research to perform minimum weight design of free-form steel space-frame roof structures consisting of rectangular hollow structural sections (HSS). The new methodology is applied to two example roof structures subjected to the AISC LRFD code (AISC, 2005) and ASCE-10 snow, wind, and seismic loading (ASCE, 2010). Both structures are train station roofs for the Ottawa Light Rail Transit (OLRT) system to be built in Ottawa, Canada, in 2018. The structures are made up of a diamond-shaped grid pattern, and their members are subjected to torsion in addition to bending and axial forces. The first example roof structure, referred to as Station 1 in this work, is presented in Figure 1 and the second, Station 2, is presented in Figure 2.
The algorithm is used to improve upon existing designs which were created in a design office using a fixed geometry developed by architects and include the design of member cross-sectional dimensions only. The structural designer used an iterative design process over a period of days and was limited due to architectural constraints. The algorithm presented allows for the integration of structural engineering into the early architectural design stages of complex free-form structures.

The GA developed for this work uses a two-phase approach to improve convergence and computation time. On a common workstation typical of most design offices, the algorithm is used to simultaneously perform dimension, topology, and shape optimization of complex structures. Advantages of the two-phase GA are three-fold: a) improvement of various aspects of the design by allowing both structural and architectural involvement in early stages of design, b) automation of the design process of complex one-of-a-kind structures, relieving the designer
of days of iterative design process, and c) achievement of considerably lighter and therefore more economical designs.

1.2 Literature Review

A number of important considerations must be taken into account in structural optimization of large structures, such as geometric uncertainties, smoothness and curvature of boundaries, convergence, stability, and computation time. These issues are addressed in the following section which presents a review of journal articles published on shape and topology optimization of structures. The review is divided into classical and heuristic optimization methods.

1.2.a Classical Optimization Methods

Yonekura and Kanno (2010) describe global topology optimization of two-dimensional (2D) trusses under load uncertainty. They define the problem as a 0–1 mixed integer semidefinite programming problem and solve it using a branch and bound method.

Walls and Elvin (2010) emphasize the high computational effort associated with structural optimization. The number of iterations used with sizing optimization increases exponentially with the number of member sizes in the database. The authors use an iterative optimization method based on the principle of virtual work to minimize the mass of a 60-story frame. Computational effort is reduced by varying member sizes for critical members only.

Niu et al. (2011) discuss the meaning of minimum compliance optimization. Compliance is defined by strain energy, the difference between the work done by reaction forces on initial displacements and the work done by external forces. However when no initial displacements exist, compliance is defined as the work done by external forces. In each case minimum
compliance implies maximum stiffness, and the authors argue that minimum compliance design will always produce optimum results.

In structural optimization most of the computational effort is due to structural analysis requiring repeated solutions of linear systems of equations. A solution to reduce the computational burden is performing approximate structural analysis. Amir and Sigmond (2011) provide an approximate method for the solution of the nested analysis equations to reduce computational efforts in topology optimization. They present 3D topology optimization of a cantilever beam using 90x15x30 mesh elements to minimize compliance.

Kumar and Parthasarathy (2011) report topology optimization of a 2D cantilever beam with 1600 finite elements using a sequential optimization algorithm with the goal of material distribution for minimal compliance by essentially maximizing stiffness. The authors use B-splines for both modeling and analysis to create smooth optimal geometries without using a superfine mesh. In this case B-splines are density functions which describe the boundaries with contours and naturally stay smooth throughout topology optimization.

Ma et al. (2011) perform mixed (discrete-continuous) dynamic reliability-based minimum weight design of 2D trusses. Structural shape and dimension optimization are considered simultaneously subjected to dynamic loads due to winds, earthquakes, or explosions.

Cai (2011) studies continuum tension-only or compression-only structures which require non-linear finite element analysis. He simplifies the structural analysis by replacing the actual material with a material which is similar in elastic parameters but isotropic and finds that the material distributes itself along paths of high stress much like it would using a more complicated analysis.
Andreassen et al. (2011) perform topology optimization using the standard optimality criteria method with a heuristic updating scheme on 2D beams with 7,500 finite elements. The optimization program was written in MATLAB which is not as powerful as formal computer programming languages. The authors propose two methods to use MATLAB more efficiently: 1) loop vectorization and 2) memory preallocation. Loop vectorization uses matrices so that slow while and for loops are not needed. Memory preallocation sets array sizes to maximum so that MATLAB does not reallocate memory and data into them with every iteration.

Arnout et al (2012) perform shape and thickness optimization of a variety of examples: an arch, a cantilever beam, a cantilever pipe, and an auditorium shell structure. The authors use a parameter-free approach which uses nodal coordinates and nodal and/or element thicknesses as design variables. The objective is to minimize volume (e.g. material use) and constraints are set to limit stress in the structure using the Kreisselmeier-Steinhauser function, a response function.

The difficulties associated with solving dynamic problems include solving equations of motion every iteration. Motamarri et al (2012) propose a solution to these time-consuming difficulties when performing topology optimization on nonlinear, dynamic problems. The authors use an equivalent linear system allowing a significant decrease in the number of nonlinear analyses required to obtain an approximate equivalent load. An example 0.4 m by 0.1 m by 0.001 m thick beam is used.

Optimality criteria methodology is performed by Rozvany and Sokol (2012) for topology optimization which considers support costs and unequal tension and compression stresses. They provide examples with different supports, a pinned arch and an arch with one pinned end and one roller.
1.2.b Heuristic Optimization Methods

Classical optimization methods are considered gradient-based methods while non-gradient-based optimization methods are heuristic. Heuristic methods, on the other hand, require many more finite element (FE) structural analyses because of a large population of solutions.

Hasancebi et al. (2010) consider 500-member space-frames and compare deterministic approaches such as mathematical programming schemes requiring initial designs with non-deterministic approaches such as heuristic schemes which start with random populations of designs. The authors compare seven non-deterministic algorithms concluding that simulated annealing and evolutionary searches give robust solutions with a fast and linear convergence while Harmony Searches and simple Genetic Algorithms converge slowly and are ineffective for large-scale problems.

Simultaneous topology, shape, and dimension optimization of 3D trusses is done by Noilublao (2011) using a multi-objective heuristic methods. The multiple objectives consider mass, compliance (stiffness), natural frequency, a frequency response function, and force transmissibility. When all three optimization types are performed simultaneously it is considered a single-stage approach. The authors argue that this is more efficient than a two-stage approach where topology optimization is done before shape and dimension optimization.

Most heuristic optimization methods are non-gradient because updating schemes depend on objective function values only, while gradient methods follow the steepest descent updating scheme. Sigmond (2011) compares non-gradient topology optimization (NGTO) methods with gradient topology optimization methods (GTO). The positive aspects of NGTO are that they perform global searches, produce discrete designs, and are easy to implement. The
negative aspects are that global solutions are not necessarily found via global searches and computational effort is high. The number of GTO evaluations increases linearly with design variables, but the number of NGTO evaluations increases exponentially with design variables.

Powerful computers have been a modern day contribution to heuristic structural design optimization problems. Ahsan et al (2012) took advantage of such hardware to solve minimum cost, mixed-variable optimization of a concrete I-girder bridge. Using an evolutionary optimization algorithm the authors were able to achieve a minimum cost design, considering materials, fabrication, and installation, in a matter of seconds.

Many structural design optimization problems are reliability based and involve maximizing reliability or minimizing the probability of failure. Greiner and Hajela (2012) perform multi-objective simultaneous dimension and topology optimization to minimize mass and maximize reliability. The evolutionary algorithm used results in a set of minimum weight designs for different reliability levels. Examples are provided for bar trusses consisting of up to 20 members.

Parvizian et al (2012) use a heuristic cell-based algorithm for topology optimization of cantilever beams with as many as 1,564 cells. The algorithm is extended to the finite cell method where the authors use the p-method to improve finite elements and use larger elements for shape functions. This provides the authors with a good approximation, results in smooth boundaries, and reduces computation time.

Patel and Choi (2012) use probabilistic neural networks with a classification approach for reliability-based heuristic topology optimization problems which minimize strain energy. The authors use a 100m x 100m truss example and a classifier associated with each design to indicate its level of safety.
Genetic Algorithms

Evolutionary optimization or Genetic Algorithms (GA’s) are based on the Darwinian survival of the fittest theory. Design variables make up the genetic information and are stored in binary bit strings known as chromosomes which represent an entire design solution. The design solutions are improved through iterations of operations including crossover, reproduction, and mutation. GA’s are improved by properly defining the search space and using a good initial design.

Winslow et al. (2010) study 3D free-form grid structures by performing simultaneous shape and topology optimization using a multi-objective GA (MOGA). The authors subdivide the surface into regions, each with one repeated unit cell which can be used to optimize any grid shell structure. The design variables are member orientation and height where orientation is defined by the angle between members in a unit cell and the angle between the unit cell and a local axis defining the region’s orientation.

Das et al. (2011) use an evolutionary algorithm for topology optimization of 3D cantilever structures and 3D structural components (e.g. an aircraft bulkhead and the sideframe of a railway freight wagon) with the goal of minimizing the weight of the structure subject to strength constraints. The finite element analysis required for topology optimization of complex structures makes computational effort extremely high. The authors propose a modified evolutionary structural optimization (ESO) method which improves the ESO criterion, the material elimination factor, and the convergence criteria and avoids obtaining meshes with checkerboard patterns and unrealistic results.

Kripakaran et al. (2011) perform trade-off studies on the costs and possible savings associated with different types of connections in a five bay, five story steel moment resisting
frames using a GA. The authors point out that a large number of design variables leads to a large search space which reduces the quality of the search.

Shape optimization problems are not always well posed and are usually non-convex, meaning that they have multiple local optimal solutions thus making the search for the global optimum solution more daunting. Fraternali et al. (2011) perform shape optimization of 3D dome-style roof structures using a GA with an arbitrary configuration of elements known as a ground structure and a fitness function based on strain energy. The authors employ form-finding techniques such as funicular curves and optimal thrust surfaces. They compare results from minimizing the meridian stress with minimizing displacement for spherical, baroque, and gothic dome roof structures, and find that minimizing displacement does not conserve the initial structure geometry. In other words, the optimal shapes for all three dome styles have a similar appearance and resemble a regular, spherical dome.

The computational cost of large structural shape or topology optimization problems is usually high, especially when combined with sizing optimization where two different types of design variables are used such as cross-sectional area and nodal coordinates leading to slow convergence or divergence. Wei et al. (2011) combine the simplex method with a GA and uses parallel computing to reduce the computation cost of optimization of 2D and 3D dome-type trusses with frequency constraints requiring dynamic sensitivity analyses.

Balamurugan et al. (2011) implements a two-phase GA optimization method where the first phase uses a binary representation of the structure to find a global optimum, and the second phase uses a geometric representation to further the search by finding a local optimum. The authors suggest that GA’s can often result in degenerate solutions and state that their first
phase optimization process finds unique solutions only. The authors perform minimum weight optimization on 2D cantilever plates.

Guo and Li (2011) use a Genetic Algorithm to compare dimension, shape, topology, and layer optimization results for a large 3D transmission tower with 1,052 members and 315 joints. The author’s four optimization methods consider section sizes only, section sizes and shape only, section sizes, shape, and topology, and section sizes, shape, and layers, and the best results (i.e. lighter structures) are obtained using the latter two combination optimization methods. The authors emphasize a variety of possible constraints one could consider including natural frequency, stress, displacement, and Euler buckling as well as symmetry, safety, and acceptable shapes and topologies.

Computation time is significantly improved when methods are employed to find good solutions faster. Ashtari and Barzegar (2012) develop such a method by combining a GA with fuzzy logic. They use a real-valued satisfaction parameter and a membership value. Each design has a membership value for its objective and constraint functions and if that value equals the satisfaction parameter, the solution is deemed a good one. They proposed methodology is used to perform minimum weight and cost optimization of a 4-story, 72-bar space truss, a 15-bar truss, a 10-bar truss, and a welded beam structure.

Sometimes multi-objective optimization can be difficult because the objectives are not related to one another and may even conflict. Gardenghi and Wiecek (2012) explore such problems referred to as multi-disciplinary optimization. The authors compare GAs with mathematical programming and look at quasiseperable and seperable problems which involve separating or splitting up the problem. They perform optimization of a speed reducer using
three objectives: 1) minimize total volume, 2) minimize stress in one of two shafts, and 3) minimize stress in the other shaft.

Liu et al (2012) perform minimum weight GA optimization on skeletal structures using an improved crossover operation and penalty function. Optimization is performed on a 2D 12-member truss and a 3D 72-member skeletal structure using frequency constraints.

**Particle Swarm Optimization**

Particle swarm optimization (PSO) also known as ant colony optimization (ACO) is a population-based method which involves using velocity vectors to steer the paths of individual particles toward improving their personal performance or the performance of the entire swarm. Dimou and Koumousis (2009) use a reliability-based particle swarm method to perform shape and dimension optimization of 2D trusses and arches with the goal of minimizing the probability of failure. Kaveh and Talatahari (2010) discuss minimum weight design of 2D steel frames using PSO and apply it to a 3-bay and 24-story steel frame.

A method similar to PSO is the Firefly Algorithm (FA) which is used by Gandomi et al. (2011) to minimize the cost of a welded beam, a pressure vessel, and a reinforced concrete beam, to minimize volume of a helical compression spring and a stepped cantilever beam, and to minimize weight of a car when subjected to side impact. The FA is inspired the flashing characteristics of fireflies and is proven through the authors’ examples to produce a more optimal solution than other heuristic algorithms such as PSO, GA’s, and differential evaluation (DE).

Luh and Lin (2011) use a two-stage approach to minimum weight PSO of the size, topology, and shape of a 2-tier, 39-member truss. A one-stage approach integrates all design
variables for each optimization into one and requires more computational effort than the proposed two-stage approach. First, binary topology optimization is performed, then size and shape optimization are performed using position and velocity vectors. The solution obtained is not necessarily globally optimal; however the authors report low calculation time and high diversity.

Mashayekhi et al (2012) perform reliability-based simultaneous dimension and topology optimization on space structures with double layer grids. The authors consider load uncertainties and look at Eigenvalues and stiffness to obtain failure modes and determine the probability of failure. The method of moving asymptotes and ant colony optimization is employed using a two-phase approach which considers the importance or structural significance of joints. Stiffness is maximized on a 20x20 double layer grid with 3,200 members.

Sharafi et al (2012) use ant colony optimization to minimize cost for 3D reinforced concrete frames. The authors consider systems and materials in their objective function and optimize characteristics including span lengths and column locations. Design standards are used as constraints and a 3-story, 37.5m x 37.5 m frame with a 6x6 column grid is optimized.

Harmony Search Method

The harmony search method is a population-based algorithm where design variables represent music pitches and solutions represent a musical chord. A set of random solutions is generated and pitch adjustments or small changes are made until the stopping criteria are reached.

There are two parameters used in harmony search methods; 1) harmony memory considering rate and 2) pitch adjusting rate. Initial selection of these typically determines the
success of the optimization process. However, Hasancebi et al. (2010) modify their harmony search method to adjust these parameters automatically throughout the weight minimization process for a 162-member planar steel frame thus rendering initial selection unnecessary.

Martini (2011) discusses multimodal size, shape, and topology optimization of 2D trusses and 3D arches, defined as generating a range of near-optimal solutions rather than a single best solution allowing the designer to consider aesthetics and constructability as a design factor. The author presents strategies for generating multiple solutions with harmony improvisation and local replacement in order to apply harmony search method to multimodal optimization.

Dome structures provide interesting examples for topology optimization. Carbas and Saka (2012) perform minimum weight optimization to determine the optimal number of rings, height, and member sizes for domes using all discrete variables. The authors consider 20 meter diameter Lamella, Network, and Geodesic domes. The use a Harmony Search method with AISC, LRFD constraints.

**Level Set-Based Methods**

Level set-based methods of optimization involve a so called structure “boundary” which moves to its optimal position. Boundary motion is described using either of two methods: 1) the Lagrangian method which observes trajectories of nodal points and 2) the Eulerian method which observes velocity at fixed positions where each point has a velocity vector field associated with it.

Discrete 0-1 integer programming can be ill-posed due to the lack of consideration for continuum mechanics. Lou et al. (2009) use an implicit level set-based method in which the
structure boundary is at the zero level set of a higher dimensional level set function. The authors perform shape and topology optimization of 2D beams with the objective of minimizing strain energy.
Chapter 2: Genetic Algorithm Formulation and Dimension Optimization

2.1 Problem Definition

The objective of this chapter is dimension optimization of free-form steel space-frame roof structures consisting of steel rectangular hollow structural sections (HSS) using a modified genetic algorithm. An example of such a structure, Station 1 (Figure 1), is shown in Figure 3 through 6. Figure 3 is a 3D perspective view of the structure while Figure 4 shows the top view, Figure 5 shows a side view along the length, and Figure 6 shows another side view along the width.

Figure 3: 3D Perspective view of Station 1
HSS are used because of the sections’ superior resistance to bending in both its major and minor axes. For practical and aesthetic reasons, it is assumed that every roof member has
the same width, \( b \), and the same depth, \( d \), (Figure 7). Columns, however, can either have the same width, \( b_c \), and depth, \( d_c \), or each can differ depending on the structure and/or designer preference. Both roof and column thicknesses, \( t \), can vary.

![Figure 7: Typical cross-section of roof members](image)

Regularity improves the constructability of a structure. Since the class of structures considered typically has hundreds of members, the designer often divides the structure into regions based on similar response characteristics (e.g. internal member forces and moments) to introduce regularity. In this case, each member in a region has the same cross-sectional dimensions. As an example, the roof structure of Figure 1 may be divided into six roof regions and five column regions shown in Figures 8 and 9.

![Figure 8: Initial roof design regions of Station 1](image)
Figure 9: Initial column design regions of Station 1

The objective function for the optimization problem is total weight of the structure and is expressed as:

\[
f(d, b, d_c, b_c, t_i, t_q) = \rho \left( \sum_{i=1}^{n} [bd - (b - 2t_i)(d - 2t_i)]L_i + \sum_{q=1}^{n_c} [b_c d_c - (b_c - 2t_q)(d_c - 2t_q)]L_q \right)
\]

(1)

where \( n \) is the number of groups of members in the roof structure with the same wall thickness, \( t_i \) is the wall thickness of members in roof group \( i \), \( L_i \) is the total length of members in roof group \( i \), \( n_c \) is the number of groups of columns with the same wall thickness, \( t_q \) is the wall thickness of members in column group \( q \), \( L_q \) is the total length of columns in column group \( q \), and \( \rho \) is the unit weight of steel. As such the total number of variables in this optimization problem is \( n + n_c + 4 \) (\( t_i \) for each \( n \), \( t_q \) for each \( n_c \), \( b \), \( d \), \( b_c \), and \( b_c \)).

Members of the roof structure are subjected to axial force, major-axis bending, minor-axis bending, shear force in the x and y-directions, and torsion. Columns are used in pairs in a V-shape form. They are pinned at the top and fixed at the bottom (Figures 3 through 6).

The basis of design is the AISC LRFD specifications. The following constraints are taken from chapters D through H of the LRFD code (AISC, 2005):
1. Axial Force, either tension (LRFD Chapter D) or compression (LRFD Chapter E)

\[
\left( \frac{P_{uj}}{\varphi P_{nj}} - 1 \right) \leq 0 \quad \text{for } j = 1,2, \ldots, N
\]  

(2)

2. Flexure, both major and minor axis (LRFD Chapter F)

\[
\left( \frac{M_{uxj}}{\varphi_0 M_{nxj}} - 1 \right) \leq 0 \quad \text{for } j = 1,2, \ldots, N
\]

(3)

\[
\left( \frac{M_{uyj}}{\varphi_0 M_{nyj}} - 1 \right) \leq 0 \quad \text{for } j = 1,2, \ldots, N
\]

(4)

3. Shear, both major and minor axis (LRFD Chapter G)

\[
\left( \frac{V_{uxj}}{\varphi V_{nxj}} - 1 \right) \leq 0 \quad \text{for } j = 1,2, \ldots, N
\]

(5)

\[
\left( \frac{V_{uyj}}{\varphi V_{nyj}} - 1 \right) \leq 0 \quad \text{for } j = 1,2, \ldots, N
\]

(6)

4. Combined flexure and axial force (LRFD Chapter H)

\[
\begin{align*}
\text{for } P_{uj} \geq 2, \\
\left[ \frac{P_{uj}}{\varphi P_{nj}} + \frac{8}{9} \left( \frac{M_{uxj}}{\varphi_0 M_{nxj}} + \frac{M_{uyj}}{\varphi_0 M_{nyj}} \right) - 1 \right] \leq 0 \\
\text{for } P_{uj} < 2,
\end{align*}
\]

\[
\left[ \left( \frac{P_{uj}}{\varphi^2 P_{nj}} + \frac{M_{uxj}}{\varphi_0 M_{nxj}} + \frac{M_{uyj}}{\varphi_0 M_{nyj}} \right) - 1 \right] \leq 0 \quad \text{for } j = 1,2, \ldots, N
\]

(7)

(8)

5. Combined torsion, shear, flexure, and compression/tension axial force (LRFD Section H)

\[
\begin{align*}
\left[ \left( \frac{P_{uj}}{\varphi P_{nj}} + \frac{M_{uxj}}{\varphi_0 M_{nxj}} \right) + \left( \frac{V_{uxj}}{\varphi V_{nxj}} + \frac{T_{uj}}{\varphi T_{nj}} \right)^2 - 1 \right] \leq 0 \\
\left[ \left( \frac{P_{uj}}{\varphi P_{nj}} + \frac{M_{uyj}}{\varphi_0 M_{nyj}} \right) + \left( \frac{V_{uyj}}{\varphi V_{nyj}} + \frac{T_{uj}}{\varphi T_{nj}} \right)^2 - 1 \right] \leq 0
\end{align*}
\]

\[
\text{for } j = 1,2, \ldots, N
\]

(9)

(10)

where \( N \) is the total number of members, \( M_{nxj} \) is the nominal flexural strength with respect to the major axis of member \( j \), \( M_{nyj} \) is the nominal flexural strength with respect to the minor axis of member \( j \), \( T_{nj} \) is the nominal torsional strength of member \( j \), \( P_{nj} \) is the nominal compression or
tension strength of member \( j \), \( V_{nxj} \) is the nominal shear strength of member \( j \) in the \( x \) direction, \( V_{nyj} \) is the nominal shear strength of member \( j \) in the \( y \) direction, \( M_{uxj} \) is the required flexural strength of member \( j \) with respect to the major axis, \( M_{uyj} \) is the required flexural strength of the member \( j \) with respect to the minor axis, \( T_{uj} \) is the required torsional strength of member \( j \), \( P_{uj} \) is the required compression or tension strength of member \( j \), \( V_{uxj} \) is the required shear strength of member \( j \) in the \( x \) direction, and \( V_{uyj} \) is the required shear strength of member \( j \) in the \( y \) direction.

The aforementioned constraints are highly nonlinear implicit and discontinuous functions of design variables for roof members and columns which are known to cause convergence and stability problems when using gradient-based optimization algorithms (Abuyounes and Adeli, 1986; Adeli and Kamal, 1986, 1991). For example, the flange local buckling constraint used to find \( M_{nj} \) in Eqs. (7) through (10) for noncompact sections is (AISC Eq. F7-2):

\[
M_n = M_p - (M_p - F_y S) \left( 3.57 \frac{b}{t} \sqrt{\frac{F_y}{E}} - 4.0 \right) \leq M_p
\]  

(11)

where \( M_n \) is the nominal flexural strength, \( F_y \) is the yield stress, and \( E \) is the modulus of elasticity of steel. \( M_p \) is the plastic moment and \( S \) is the elastic section modulus, both of which are functions of design variables \( d \), \( b \), and \( t \). The flange local buckling constraint for sections with slender flanges is (AISC Eq. F7-3):

\[
M_n = F_y S_{eff}
\]  

(12)

where \( S_{eff} \) is the section modulus determined using the following equation for effective width, \( b_e \) (AISC Eq. F7-4):

\[
---
\]
and the web local buckling equation also used to find $M_{nj}$ in Eqs. (7) through (10) for sections with noncompact webs is (AISC Eq. F7-5):

$$M_n = M_p - \left( M_p - F_y S_x \right) \left( 0.305 \frac{d}{t} \sqrt{\frac{F_y}{E}} - 0.738 \right) \leq M_p$$

where $S_x$ is the elastic section modulus with respect to the major $x$ axis, $h$ is the web depth, and $t_w$ is the web thickness. Lateral torsional buckling is not an issue because all HSS shapes provided in the AISC Steel Construction Manual (AISC, 2005) have adequate torsional resistance and their $d/b$ ratios are less than 6. A major advantage of GAs is that no function gradient computation is required (Sarma and Adeli, 2000a&b).

### 2.2 Optimization Problem Formulation using a Genetic Algorithm

The minimum weight optimization problem is solved using a modification of the Genetic Algorithm (GA) developed by Adeli and Cheng (1993, 1994a&b) for optimization of structures adapted into a two-phase algorithm to improve the optimization convergence and speed for large structural design problems. A GA finds optimal design solutions based on individual fitness measures of hundreds to thousands of possible design alternatives. Unlike classical optimization methods which use an objective function with multiple constraint equations, a standard GA uses one fitness function. As such, a GA can be used directly for unconstrained optimization problems only. In order to transform the constrained optimization problem into an unconstrained optimization problem, the following fitness function is developed by combining the objective function, Eq. (1), with the constraint equations, Eqs. (2) through (10):
\[
\min F(d, b, d_c, b_c, t_i, t_q) = \frac{\rho}{\beta} \left[ \sum_{i=1}^{n} (b d - (b - 2t_i)(d - 2t_i))L_i + \sum_{q=1}^{n_c} (b_c d_c - (b_c - 2t_q)(d_c - 2t_q))L_q \right] + \sum_{j=1}^{N} \left[ \gamma_j g_{c,j} \left( \text{d or } d_c, \text{b or } b_c, \text{t}_i \text{ or } t_q \right) - 1 \right]^2
\]  

(15)

where \( g_{c,j} \) represents the most critical constraint value (i.e. the constraint value which governs design) taken from Eqs. (2) through (10). The following is an example \( g_{c,j} \) when Eq. (7) governs:

\[
g_{c,j} = \left( \frac{P_{u,j}}{\rho_{P_n,j}} + \frac{8}{9} \left( \frac{M_{ux,j}}{\rho M_{nx,j}} + \frac{M_{uy,j}}{\rho M_{ny,j}} \right) \right)
\]  

(16)

In Eq. (15) \( \beta \) is a normalizing constant used to ensure that the objective function value does not dominate the fitness function unduly and establishes a weighting between the objective function’s and the constraint function’s contribution to the fitness function. The closer all \( g_{c,j} \) values are to 1.0, without going over, the more economically designed the structure. By weighting the fitness function more towards the governing constraint function, using a properly defined \( \beta \) value, the \( g_{c,j} \) values will converge towards 1.0 more quickly and efficiently.

In this research, \( \gamma_j \) is a penalty function which penalizes \( g_{c,j} \) values greater than 1.0. Without this penalty function, a \( g_{c,j} \) of say, 0.75, and an overstressed \( g_{c,j} \) of, say, 1.25 would contribute the same to the fitness function because they will have the same \((g_{c,j} - 1)^2\) value. The proper \( \gamma_j \) is determined by numerical experimentation in the examples section by testing various values in the range of 1.4 to 2.2.

The AISC Steel Construction Manual (AISC, 2005) provides a discrete number of manufactured HSS sizes with six possible \( t \) values and nine possible \( d \) and \( b \) values each. HSS sections in the AISC database have a maximum thickness of 5/8 inches. Such thicknesses are not large enough to carry the loads on most of the columns and potentially some of the roof members of the structures considered. As such, in this research the HSS database is expanded
for six additional thicknesses of 3/4 inches to 2 inches in increments of 1/4 inch, bringing the
total number of shapes in the HSS database to 9x9x(6+6)= 972. Station 1 (Figures 3 through 6)
has 288 members which means there are over 300,000 possible design solutions, a very large
design space.

To begin GA optimization, a population of \( N_p \)-1 design alternatives are generated around
the initial design for each design variable (d, b, d_c, b_c, t_i, and t_q) using a normal probability
distribution (\( N_p \) is the number of designs or solutions in a given population including the initial
design). An example probability distribution created for a \( t_i \) value of 3/8 inches is shown in
Figure 10. Thicknesses above 1.25 inches are impractical for roof members, so the normal
probability distribution is truncated beyond 1.25 inches to reduce the search space. Similarly,
normal probability distributions for columns include \( t \) values from 5/8 to 2 inches only. All initial
roof thicknesses (\( t_i \) values) are in the range of ¼ to ½ inches. Due to the nature of the offset
normal probability distribution, the majority of \( t_i \) values in the initial population will be greater
than the initial \( t_i \) thus increasing the average weight of the first iteration significantly as seen in
the examples section.
Populations of designs are represented by chromosomes or variable strings. Chromosomes can be made up of binary variables, actual variables (i.e. d, b, and t), or a combination of the two. If an 8-bit binary string is used to represent one design variable for 200 structural members, then the chromosome length is 1600 bits. If more design variables are introduced, their chromosome variables are simply added to the end of the string by concatenation.

Genetic information for dimension optimization consists of design variables $d$, $b$, $d_c$, $b_c$, $t_i$ and $t_q$. In this research, a variable string is used instead of a binary string because the design variables are discrete. Figure 11 presents an example variable string for a roof structure with six roof regions and one column region consisting of the following HSS shapes: 1) HSS12x8x3/8, 2) HSS12x8x1/2, 3) HSS12x8x5/8, 4) HSS12x8x3/8, 5) HSS12x8x1/4, 6) HSS12x8x5/8, and HSS12x18x1.5 columns.
Once the fitness for every design solution in the current population is calculated using the fitness function defined by Eq. (15), unfit solutions are eliminated using the following equation where any solution whose fitness is greater than the average, $F_{ave}$, of the population is reassigned a fitness of zero:

$$F_k(d, b, d_c, b_c, t_i, t_q) = \begin{cases} 
F_{ave} - F_k(d, b, d_c, b_c, t_i, t_q) & \text{for } F_k(d, b, d_c, b_c, t_i, t_q) < F_{ave} \\
0 & \text{for } F_k(d, b, d_c, b_c, t_i, t_q) \geq F_{ave}
\end{cases}$$

$$k = 1, 2, ..., N_p$$

(17)

where the fitness of an individual design solution is denoted $F_k$.

The three basic operations of a GA, reproduction, crossover, and mutation, are used to improve the fitness of each population from one generation (iteration) to the next. The reproduction operation selects the better fit designs, copies them, and places them into a mating pool allowing each to mate and reproduce. This selection process depends on the reproduction probability, $p_r$, of each design which is calculated using the following equation:

$$p_r = \frac{F_k}{F_{pop}}$$

(18)

where $F_{pop}$ is the summation of the fitnesses of the entire population. The number of copies of an individual design, $n_{cr}$, is determined using the following equation rounded to the nearest integer:
\[ n_{ck} = N_p \times p_r = \frac{F_k}{F_{ave}} \]  

After the reproduction operation is performed, the crossover operation mates the selected designs to create more fit offspring designs. The uniform crossover operation is used to combine genetic information between two parent designs. Uniform crossover selects two parent designs at a time from the mating pool and swaps variables corresponding to zeros in a binary string known as a mask. The mask is the same length as all variable strings and consists of a preselected percentage of zeros arranged randomly. This percentage is determined by numerical experimentation as discussed in the examples section. An example of uniform crossover for an example design variable string is shown in Figure 12. Each mask within a population is different, so the number of unique, randomly generated masks is equal to half of the number of solutions (parents) in the population multiplied by the total number of generations (populations).

![Figure 12: Example of uniform crossover operation](image)

The mutation operation is used to add diversity to the search space by randomly changing a variable in a design solution. During mutation the value of any chromosome variable may be changed to a randomly selected variable with each having a small probability of mutation of, say, 0.005. The proper value of the probability of mutation is problem-specific and is determined by numerical experimentation as discussed in the examples section.
Roof structures of the type considered in this research are complex and may not easily be divided into a few regions of members having the same cross-section. Ideally, to obtain the minimum weight design, each member should be considered individually as a separate design variable rather than using regions; however, this adds to the number of design variables. The computation time increases with the number of design variables significantly. To arrive at an economic design in a reasonable amount of time, a two-phase method is developed where the GA is performed on the structure divided into a few regions preselected by the designer until a near-global but slightly infeasible optimal solution is found. At this point the highly stressed members in each region are determined and additional variables representing each are added to the optimization problem. In other words, each individual highly stressed member is assigned its own region and the total number of regions is increased accordingly. A highly stressed member is defined as a member with a critical constraint value, $g_{c,j}$, of 0.95 or greater. While members with $g_{c,j}$ values ranging from 0.95 to 1.0 are not overstressed, they are sensitive to changes in the structure and can easily become overstressed. These members are removed from large regions strategically and are assigned their own regions so that they do not control the region and result in overdesigned members.

Next, in the second phase, the GA is performed on the expanded optimization problem again until all constraints are satisfied and an optimum solution is obtained. Otherwise, forcing all members to satisfy the design constraints in a one-step GA optimization process will result in a heavy and uneconomical structure. In addition, it will also require a large number of iterations on the order of hundreds, if not thousands, making the optimization process computationally impractical on standard workstations.
The standard GA design iteration is interrupted, that is phase one is completed, when both of the following two conditions are met:

1. The average weight of the structure in the current iteration has decreased less than 0.3% from the previous iteration, (this number was selected after trying three different values: 0.5%, 0.3%, and 0.1%) and
2. No more than 5% of the structural members are overstressed

During the second phase, the algorithm starts with a new initial design by increasing the thickness dimension of the overstressed members one increment at a time until all constraints are met. In this particular class of structures, columns with depth and width dimensions ranging from 12 to 20 inches, thicknesses of ¾ to 2 inches, and lengths of approximately 25 feet affect the total weight of the structure significantly. Increasing the column thicknesses by one increment of 1/4 inch increases the total weight of the structure by about 3%. Therefore during phase two of the design, outer dimensions of individual columns are also allowed to change in order to guide the design towards the global optimum. An initial population is created around the initial design using a normal probability distribution similar to that of phase one but with a smaller range of design variables to choose from (the initial design variable ±1 increment for d and b values and ±3 increments for t values).

Flowcharts describing the entire algorithm including phase one design, GA operations, and phase two design are presented in Figures 13, 14, and 15, respectively. In these figures, \( l = 1, 2, \ldots, N \) is the design iteration/generation counter where \( N \) is the maximum number of design iterations and \( p = 1, 2, \ldots, N \) is the design variable counter where \( N \) is the number of design variables.
Figure 13: Flowchart for dimension optimization phase one
Figure 14: Flowchart for GA operations
Figure 15: Flowchart for phase two of the two-phase GA
2.3 Examples

Two free-form steel space-frame roof structures are presented in this research. A500 Grade B steel is used for all roof frame members with a yield stress of $F_y = 46$ ksi and ultimate strength of $F_u = 58$ ksi. Structural analysis is performed using SAP2000 considering snow (S), dead (D), wind (W), and earthquake (E) loads assuming linearly elastic behavior but taking into account the p-delta affects. The stations are designed for a large snow load of 48.5 psf. Between steel frame members are triangular wood panels making up a dead load of 20 psf in addition to the self-weight of the steel. Wood panels are modeled as thin triangular plate elements and do not transfer in-plane forces. Wind loading is based on a 115 mph wind speed producing unique pressures for each example as described in the following sub-sections. Both stations are located on site class C soil conditions and are designed as ordinary moment resisting frames (OMRF) for a ground acceleration of 0.42g during seismic activity. The following load combinations are considered per ASCE 7-10 (ASCE, 2010):

1. $1.4D$
2. $1.2D + 1.6S + 0.5W$
3. $1.2D + 1.0W + 0.5S$
4. $1.2D + 1.0E + 0.2S$
5. $0.9D + 1.0W$
6. $0.9D + 1.0E$

2.3.a Example 1

The first roof structure, Station 1, is located along the Rideau River in Ottawa and is shown in Figures 3 through 6. The roof structure is 224 feet long, 75 feet wide, and 27 feet tall. It has 278 structural members in the roof plus 10 inclined columns. The structure is pin-connected to a concrete wall on the side without columns. Roof connections to either the concrete wall or the top of the columns are shown in Figures 4 and 6.
Based on a preliminary structural analysis of the initial design, it was determined that major-axis bending was the main controlling factor in design. Also, considering its shape regularity the roof structure is divided into six regions with similar major-axis bending distribution, each with the same cross-sectional member as shown in Figure 8. Columns are divided into five regions as shown in Figure 9 all with the same outer dimensions in phase one design. As such the initial (phase one) optimization problem has 15 design variables: thickness, \( t_i \), for each roof region, depth, \( d \), and width, \( b \), for all roof members, thickness, \( t_q \), for each column region, and depth, \( d_c \), and width, \( b_c \), for all columns.

Wind produces a windward pressure of 22.3 psf, a central pressure of 18.0 psf, and a leeward pressure of 11.5 psf all directed away from the roof per ASCE7-10 (ASCE, 2010) and is shown in Figure 6. The fundamental period of the structure is in the range of 0.27 to 0.36 seconds and the base shear is in the range of approximately 50 to 54 kips.

Values of the two parameters used in the constrained GA optimization algorithm, \( \beta \) and \( \gamma \), are problem-specific and determined from numerical experimentation. The first parameter, \( \beta \), involves the contributions of objective and constraint functions to the fitness function. The fitness function must be weighted properly such that the solution converges quickly to an optimum design meeting all constraints. In order to speed up convergence towards a feasible solution, a larger weight is assigned to the critical constraint function than the objective or weight function.

The objective and constraint function’s contributions to the fitness function are expressed as \( X\% \) and \( Y\% \) (or \( X/Y \)), respectively, where \( Y \geq X \). Percentages \( X \) and \( Y \) were determined after solving small test optimization sub-problems where an initial population of 30 design solutions underwent 20 design iterations. Convergence curves for four different ratios of
objective to constraint functions, 50/50, 45/55, 40/60, and 35/65, are presented in Figure 16.
The percentages of overstressed members in the final solution, that is after 20 design iterations,
for the aforementioned combinations are presented in Figure 17. A value of $X/Y=40/60$ is
selected considering the following: a) the final weight of the structure, b) the speed and
smoothness of the convergence curve, and c) the percentage of overstressed members.

Figure 16: Convergence curves for various ratios of objective and constraint functions (presented as $X/Y$
where $X =$ percentage contribution of objective function to fitness function and $Y =$ percentage
contribution of constraint function to fitness function)
Figure 17: The percentages of overstressed members in the final solution for various ratios of objective and constraint functions (X = percentage contribution of objective function to fitness function, and Y = percentage contribution of constraint function to fitness function)

The ratio 40/60 produced the lowest weight solution with the least amount of overstressed members. A significant drop in weight is observed from iteration 16 to 17 on the 40/60 convergence plot in Figure 16. This is the result of the column thicknesses decreasing by ¼ inch resulting in a significant change in weight since the 10 columns are relatively large: 16 inches deep by 20 inches wide by 25 feet long.

The Coefficient $\beta$ is determined using the objective and constraint functions values for the initial solution and remains constant throughout the optimization process. The following equations are used to determine $\beta$:

\[
F_o(d, b, d_c, b_c, t_i, t_q) = \frac{1}{\beta} f_o(d, b, d_c, b_c, t_i, t_q) + g_o(d, b, d_c, b_c, t_i, t_q)
\]  

(20)

\[
0.60F_o(d, b, d_c, b_c, t_i, t_q) = g_o(d, b, d_c, b_c, t_i, t_q)
\]  

(21)

\[
0.40F_o(d, b, d_c, b_c, t_i, t_q) = \frac{1}{\beta} f_o(d, b, d_c, b_c, t_i, t_q)
\]  

(22)
where $f(d,b,d_c,b_c,t,t)$ and $g(d,b,d_c,b_c,t,t)$ are the objective and constraint function values, respectively, and the subscript “o” is used for initial values of each function. The value of $\beta$, determined from the initial solution presented in Table 1, is 3.0879 for Station 1.

The goal of phase one design is to produce a low-weight solution which is close to the feasible optimal solution with a few constraints yet to be satisfied. Phase two of the algorithm takes the phase one final solution and improves it towards the optimum solution while ensuring all constraints are completely satisfied. As such the $\gamma$ value used during phase one should be relatively small in order to achieve a slightly under-design solution, and the $\gamma$ value used during phase two should be larger in order to force feasibility but not so high that it overdesigns the structure.

Various phase one $\gamma$ values of 1.4, 1.6, 1.8, 2.0, and 2.2 were tested using a similar optimization sub-problem with an initial population of 20 design solutions and 20 design iterations. Figure 18 displays convergence curves for different $\gamma$ values. Figure 19 is a plot of the percentage of overstressed members in the final design, after 20 iterations, for each $\gamma$. A $\gamma$ value of 2.2 results in the lowest percentage of overstressed members: 2.4% compared to 2.8% for $\gamma = 2.0$. However, it yields a high-weight final design of 205 kips compared to 187 kips for $\gamma = 2.0$. Therefore, $\gamma = 2.2$ seems to lead to an overdesign of the final solution while $\gamma = 2.0$ yields a relatively low-weight solution with few overstressed members and converges smoothly and relatively quickly. A value of $\gamma = 2.0$ is used in phase one.
Figure 18: Convergence curves comparing various overstressed penalty coefficients for phase one design.
Figure 19: Percentages of overstressed members in the final design of phase one for various overstressed penalty coefficients

Three phase two $\gamma$ values, 2.25, 2.5, and 2.75, were tested using an initial population of 10 design solutions over 10 design iterations. A smaller test problem is sufficient because the search in phase two is more refined and within a narrower range. All $\gamma$ values yielded feasible solutions with no overstressed members. Convergence curves are displayed in Figure 20. The $\gamma = 2.5$ is chosen based on its smooth convergence to a low-weight solution. The $\gamma = 2.75$ is less smooth than $\gamma = 2.5$ while $\gamma = 2.25$ yields the highest-weight solution.
Consequently, the following equation is used to determine the value of the penalty function $\gamma$ for each member:

$$
\gamma_j = \begin{cases} 
1.0 & g_{c_j} < 1.0 \\
2.0 & g_{c_j} \geq 1.0, \text{phase 1} \\
2.5 & g_{c_j} \geq 1.0, \text{phase 2}
\end{cases}$$

The uniform crossover operation uses a binary mask string with a certain percentage of zeros. The percentage of zeros determines the number of design variables which are swapped between parent designs and passed on to the offspring designs. Five different binary masks
with 40%, 50%, 60%, 70%, and 80% zeros were tested using a similar optimization sub-problem with an initial population of 20 design solutions and 20 design iterations. Convergence curves are displayed in Figure 21 and overstressed members in Figure 22. The binary mask with 70% zeros is chosen because it yields a fast and smooth convergence with a relatively low-weight design (177 kips) and low percentage of overstressed members (3.1%). The mask with 60% zeros yields a lower-weight design (169 kips) with a high percentage (6.3%) of overstressed members, and the mask with 50% zeros yields the lowest percentage of overstressed members (2.8%) but a high-weight design (198 kips).

![Figure 21: Convergence curves comparing various values for percent zeros in the mask string](image)

Figure 21: Convergence curves comparing various values for percent zeros in the mask string
Mutation is an essential operation in a GA, and its probability of occurrence though small, can greatly affect the efficiency of the optimization process. The probability of mutation was tested in a similar manner but with an initial population 50 design solutions to better represent the actual optimization problem. For small populations of 30 designs or fewer a high probability of mutation introduces variation into the search space. Small populations tend to have less variation in the initial population than larger populations of 50 or more designs, therefore the appropriate $p_m$ is dependent on population size.

Probability of mutation ($p_m$) values of 0.01, 0.005, and 0.001 were tested. The final weights are higher for this test than the previous ones but all values yielded less than 3% overstressed members. The larger population allowed for higher-weight but more feasible solutions to be found. Weight convergence curves for $p_m=0.01$ and $p_m=0.001$ are jagged compared with $p_m=0.005$ as shown in Figure 23. Towards the end of the optimization process there is a significant drop in the weight for $p_m=0.01$ but an increase in fitness (Figure 24). This shows the high $p_m$ of 0.01 to be unstable. The lowest $p_m$ (0.001) also shows instability by
increasing in weight from iteration 11 to 12. The $p_m=0.005$ value is chosen based on its stability and smooth convergence.

Figure 23: Weight convergence curves using various probability of mutation values

Figure 23: Weight convergence curves using various probability of mutation values
After appropriate values of all parameters were chosen based on the described experimentation, actual dimension optimization was performed on Station 1 using an initial (phase one) population of 50 design solutions and 50 design iterations. The entire design process took approximately ten hours using a standard workstation machine with a 2.83GHz dual core processor. The bulk of the time is for $50 \times 17 + 30 \times 33 = 1840$ structural analyses required to determine the better fit designs. Phase one design optimization took about four hours and
45 minutes to undergo 17 design iterations. The phase one design weighs 190 kips and has three overstressed members.

Phase two design optimization added 11 regions (nine roof regions and two column regions, Figures 25 and 26) and 15 design variables and used a reduced initial population of 30 design solutions. Once initial feasibility was met, the total weight of the structure increased by 5 kips (2.6%). After five and a half hours and 33 design iterations the weight was reduced by 4% and the design stayed feasible. The final weight of 188 kips is 12% lower than the initial weight. Figure 27 shows the weight convergence over the entire process and indicates when all constraints have been met. The phase one and final solutions are presented in Table 1 along with the initial design.

---

**Figure 25:** Roof design phase two (1-6 initial and added *) regions of Station 1
(9 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 9)

**Figure 26:** Column design phase two (7-11 initial and added *) regions of Station 1
(2 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 2)
Figure 27: Convergence curve for two-phase dimension optimization for Station 1
Table 1: Final optimum design for Station 1

<table>
<thead>
<tr>
<th>Region</th>
<th>Initial Solution</th>
<th>Phase 1 Solution</th>
<th>Phase 2 (Final) Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HSS Shape</td>
<td>No. Overstressed Members</td>
<td>HSS Shape</td>
</tr>
<tr>
<td>HSS Shape</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Original Roof Members</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS12x6x1/4</td>
</tr>
<tr>
<td>2</td>
<td>HSS12x8x3/8</td>
<td>0</td>
<td>HSS12x6x5/16</td>
</tr>
<tr>
<td>3</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS12x6x1/4</td>
</tr>
<tr>
<td>4</td>
<td>HSS12x8x1/2</td>
<td>0</td>
<td>HSS12x6x3/8</td>
</tr>
<tr>
<td>5</td>
<td>HSS12x8x3/8</td>
<td>0</td>
<td>HSS12x6x3/8</td>
</tr>
<tr>
<td>6</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS12x6x1/4</td>
</tr>
<tr>
<td><strong>Original Columns</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>HSS16x20x2</td>
<td>0</td>
<td>HSS16x20x2</td>
</tr>
<tr>
<td>8</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS16x20x1.5</td>
</tr>
<tr>
<td>9</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS16x20x0.75</td>
</tr>
<tr>
<td>10</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS16x20x1.25</td>
</tr>
<tr>
<td>11</td>
<td>HSS16x20x1.5</td>
<td>0</td>
<td>HSS16x20x1.5</td>
</tr>
<tr>
<td><strong>New Roof Members</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>New Cols</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Weight (kips)</strong></td>
<td><strong>214</strong></td>
<td><strong>190</strong></td>
<td><strong>188</strong></td>
</tr>
</tbody>
</table>

2.2.b Example 2

The second roof structure, Station 2, consists of a pair of identical structures situated on both sides of the New Booth Street bridge near Napean Bay in Ottawa, Canada though only one structure is considered. Station 2 is geometrically more complex than Station 1 with an irregular shape but a similar diamond-shaped grid pattern (Figures 28 through 31). This structure has 306
roof members and 34 inclined columns. Figure 28 is a 3D perspective view of the structure while Figure 29 shows a top view, Figure 30 shows a side view along the length, and Figure 31 shows a side view along the width of the structure. The roof structure is 203 feet long, 67 feet wide, and 55 feet tall. The roof is fixed to columns at locations shown in Figure 29. The taller columns, the four sets on the far side and the one set on the near side of Figure 28, are fixed to the ground while the rest are pinned to the ground.

Figure 28: 3D Perspective view of Station 2
Figure 29: Top view of Station 2

Figure 30: Side view, along the length, of Station 2
Station 2 is divided into 12 regions: six roof regions and six column regions. As with Station 1, Station 2 is also highly controlled by major-axis bending, so members with similar major-axis moments are grouped together in the 12 regions presented in Figures 32 and 33. Due to large variances in height and stresses among the 34 columns, each column region is allowed to differ in outer dimensions, d and b, as well as thickness, t. As such the initial (phase one) optimization problem has 19 design variables: thickness, \( t_r \), for each roof region, depth, d, and width, b, for all roof members, thickness, \( t_q \), depth, \( d_q \), and width, \( b_q \), for each column region.
Figure 32: Initial roof regions of Station 2

Figure 33: Initial column regions of Station 2
Wind produces a windward pressure of 2 psf on the roof, a central pressure of 22.3 psf away from the roof, and a leeward pressure of 11.5 psf away from the roof as shown in Figure 31. The fundamental period of the structure is in the range of 0.72 to 0.84 seconds and the base shear is in the range of 35 kips to 36 kips.

Station 2 is in the same class of structures and is similar in size to Station 1. Therefore, the same GA parameters tested in Example 1 (Figures 16 through 24) are used for Station 2. The value of $\beta$ for Station 2, determined from the initial solution and Eqs. (20) through (22), is 6.5663. Dimension optimization was performed on Station 2 using an initial (phase one) population of 50 design solutions and 48 design iterations. The entire design process took approximately eleven and a half hours. Phase one design optimization took about three and a half hours to undergo 14 design iterations. The phase one design weighs 253 kips and has 10 overstressed members.

Phase two design optimization added 24 regions (22 roof regions and two column regions, Figures 34 and 35) and 28 design variables. Feasibility was not achieved by incrementally increasing dimensions at the beginning of phase two, because one of the columns remained overstressed at maximum available values of $d=20$ inches, $b=20$ inches, and $t=2$ inches. When this happens the algorithm allows the member to remain infeasible and continues the optimization process until feasibility is achieved. The weight of the structure at this point is 258 kips, 2% higher than the phase one solution. It took 26 iterations for the phase two optimization to achieve feasibility. This is a good example of the benefits of a GA over classical optimization methods. For a brief period on time the structure was caught in a local, infeasible solution, but by continuing to optimize the structure as a whole, the localized stresses in the column were reduced.
With an initial population of 30 designs, phase two underwent a total of 56 iterations and took about eight hours. The final weight of the structure is 263 kips, 4% higher than the infeasible phase one solution, but 4% lower than the initial solution. Figure 36 presents the weight convergence over the entire process and indicates when all constraints have been met and Table 2 presents a comparison between initial, phase one, and final solutions.

Figure 34: Roof design phase two (1-6 initial and added *) regions of Station 2 (22 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 22)
Figure 35: Column design phase two (7-12 initial and added *) regions of Station 2 (2 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 2)
Figure 36: Convergence curve for two-phase dimension optimization for Station 2
Table 2: Final optimum design for Station 2

<table>
<thead>
<tr>
<th>Region</th>
<th>Initial Solution</th>
<th>Phase 1 Solution</th>
<th>Phase 2 (Final) Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HSS Shape</td>
<td>No. Overstressed Members</td>
<td>HSS Shape</td>
</tr>
<tr>
<td>Original</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roof Members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HSS14x10x1/4</td>
<td>0</td>
<td>HSS14x10x3/16</td>
</tr>
<tr>
<td>2</td>
<td>HSS14x10x5/16</td>
<td>0</td>
<td>HSS14x10x5/16</td>
</tr>
<tr>
<td>3</td>
<td>HSS14x10x3/8</td>
<td>0</td>
<td>HSS14x10x3/8</td>
</tr>
<tr>
<td>4</td>
<td>HSS14x10x1/2</td>
<td>0</td>
<td>HSS14x10x1/2</td>
</tr>
<tr>
<td>5</td>
<td>HSS14x10x3/4</td>
<td>0</td>
<td>HSS14x10x5/8</td>
</tr>
<tr>
<td>6</td>
<td>HSS14x10x1</td>
<td>0</td>
<td>HSS14x10x1.25</td>
</tr>
<tr>
<td>Original</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Columns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>HSS20x20x1</td>
<td>0</td>
<td>HSS20x16x5/8</td>
</tr>
<tr>
<td>8</td>
<td>HSS20x20x1/2</td>
<td>0</td>
<td>HSS12x18x5/8</td>
</tr>
<tr>
<td>9</td>
<td>HSS20x20x1</td>
<td>0</td>
<td>HSS20x20x3/4</td>
</tr>
<tr>
<td>10</td>
<td>HSS8x8x3/8</td>
<td>0</td>
<td>HSS10x6x3/8</td>
</tr>
<tr>
<td>11</td>
<td>HSS20x20x2</td>
<td>0</td>
<td>HSS16x20x1.75</td>
</tr>
<tr>
<td>12</td>
<td>HSS20x20x2</td>
<td>0</td>
<td>HSS20x20x1.75</td>
</tr>
<tr>
<td>New</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roof Members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cols</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weight (kips)</td>
<td>275</td>
<td>253</td>
<td>263</td>
</tr>
</tbody>
</table>
Chapter 3: Topology Optimization

3.1 Problem Definition

The objective of this chapter is to extend the algorithm to topology optimization with the goal of achieving additional structural efficiencies. Topology optimization uses a fixed structural geometry defined by the architect and optimizes both the member cross-sectional dimensions and the topology by adding (or subtracting) members to the structure. The roof structures considered are made up of a diamond-shaped grid pattern where each grid element has four joints and four frame members (Figure 37). In some cases, for example, when high minor-axis bending occurs, structural performance can be improved by using triangular grid elements rather than diamond grid elements. By performing topology optimization, a diamond element is transformed into two triangular elements by adding a cross member (Figure 37). Cross members are hidden by wood panels and will not alter the aesthetics of the structure.

Figure 37: Diamond grid element with frame and cross members
Again, the structures are divided into regions of the same cross-sectional dimensions. As cross members are added they make up their own region separate from the designer’s predefined regions. The optimization problem formulation is similar to that of dimension optimization presented in Chapter 2, however, the total number of members, $N$, is now a variable. The following objective function for topology optimization is similar to Eq. (1) but includes the additional weight of cross-members:

$$\begin{align*}
    f(d, b, d_c, b_c, d_x, b_x, t_i, t_q, t_x) &= \rho \left[ \sum_{i=1}^{n} \left[ bd - (b - 2t_i)(d - 2t_i) \right] L_i + \\
    \sum_{q=1}^{n_c} \left[ b_c d_c - (b_c - 2t_q)(d_c - 2t_q) \right] L_q + \left[ b_x d_x - (b_x - 2t_x)(d_x - 2t_x) \right] L_x \right] \\
    &+ \left[ \frac{1}{\rho} \left[ \sum_{i=1}^{n} \left[ bd - (b - 2t_i)(d - 2t_i) \right] L_i + \\
    \sum_{q=1}^{n_c} \left[ b_c d_c - (b_c - 2t_q)(d_c - 2t_q) \right] L_q + \left[ b_x d_x - (b_x - 2t_x)(d_x - 2t_x) \right] L_x \right] + \\
    \sum_{j=1}^{N} \left[ r_j g_c \left( d, d_c, or \ d_x, b, b_c, or \ b_x, t_i, t_q, or \ t_x \right) - 1 \right]^2 \right]
\end{align*}$$

(24)

where $d$, $b$, $d_c$, and $t$ are the depth, width, and thickness of cross members, respectively, and $L_i$ is the total length of cross members. Constraint equations are the same as defined in Chapter 2, Eqs. (2) through (10).

### 3.2 Optimization Problem Formulation using a Genetic Algorithm

The same two-phase GA developed for dimension optimization in Chapter 2 is used for simultaneous dimension and topology optimization. The following updated fitness function includes the added weight and constraints for the cross-members:

$$\begin{align*}
    \min F(d, b, d_c, b_c, d_x, b_x, t_i, t_q, t_x) &= \frac{1}{\rho} \left[ \sum_{i=1}^{n} \left[ bd - (b - 2t_i)(d - 2t_i) \right] L_i + \\
    \sum_{q=1}^{n_c} \left[ b_c d_c - (b_c - 2t_q)(d_c - 2t_q) \right] L_q + \left[ b_x d_x - (b_x - 2t_x)(d_x - 2t_x) \right] L_x \right] + \\
    \sum_{j=1}^{N} \left[ r_j g_c \left( d, d_c, or \ d_x, b, b_c, or \ b_x, t_i, t_q, or \ t_x \right) - 1 \right]^2
\end{align*}$$

(25)

Cross members support diamond grid elements against buckling inward. Their ends are fixed to provide additional lateral stiffness along the length of the structure, and in the case
when the roof structure has a cantilevered portion such as Example 2, can help support the cantilever more efficiently. Cross members are also rectangular HSS. To improve computational speed by avoiding additional design variables, all cross members are chosen to be of the same size during phase one. This is an effective strategy because the cross members are usually considerably smaller and lighter than the other frame members (up to 77% in cross-sectional areas in the examples presented in this chapter).

Genetic information for cross member cross-sectional dimensions, \( d_x, b_x, \) and \( t_x \), are added to the end of the discrete design variable string for all other members. In order to incorporate topology optimization simultaneously with dimension optimization, a binary string is added to the end of this discrete variable string. In this binary string a “1” represents the existence of a certain cross member and a “0” represents the non-existence of the cross member. The position of the binary bit within the string indicates the location of the cross member within the structure, and the number of bits is the total number of possible cross members, \( N_{ux} \). An example string for a roof structure with four roof regions and one column region consisting of the following HSS shapes: 1) HSS12x8x3/16, 2) HSS12x8x3/8, 3) HSS12x8x1/2, 4) HSS12x8x5/8, HSS12x16x1 columns, and HSS6x4x1/4 cross members is shown in Figure 38. The example roof has two out of five possible cross members.

\[
\begin{array}{cccccccccc}
\text{d} & \text{b} & \text{d}_x & \text{b}_x & \text{t}_1 & \text{t}_2 & \text{t}_3 & \text{t}_4 & \text{t}_5 & \text{binary} \\
\end{array}
\]

Figure 38: Example string representing a solution with four roof regions, one column region, and two out of five possible cross members

To begin GA optimization, a population of \( N_p \) solutions are generated around the initial design for each design variable \( (d, b, d_x, b_x, t_x, \text{or } t_x) \) using the normal probability distribution described in Chapter 2 (Figure 10), however, each cross member size \( (d_x, b_x, \text{and } t_x) \) is generated
randomly. Initial cross member configurations are also generated randomly by creating a binary string with randomly arranged “1”s and “0”s. It is assumed that the cross member can be placed across any diamond element in the roof structure.

The addition of cross members is a design strategy developed for this research. Therefore the initial design does not consider cross members. While the initial population of designs in the GA search is created around the initial design, the initial design itself is deliberately excluded from the population. Due to the random generation of cross member configuration, initial designs are relatively unfit to begin with. Including the designer’s better fit design in the initial population would steer the optimization towards that design and away from a global optimum.

Topology optimization is performed during phase one only. Phase two is used for fine-tuning and moving the solution towards a completely feasible and a globally optimum design. The added cross members redistribute stresses making it difficult to divide roof members into regions based on similar response characteristics. Regions are initially defined per designer’s initial design which does not include cross members. As the topology optimization progresses the preselected regions become less effective and new cross member configurations are formed. To account for this redistribution of stresses, roof regions are re-established periodically throughout the phase one design while column regions remain the same. Based on numerical experimentation, regions are re-assigned every three iterations. This is short enough so that the design is not too dependent on the previous region assignments and will not get stuck in a local optimum, and long enough to lead to a global optimum topology within a reasonable number of iterations. The cross member configuration, column member dimensions, and outer roof member dimensions (d and b) at every third iteration are used to
establish new regions. For this purpose, all roof frame member thicknesses are temporarily set to ½ inch to establish uniformity and allow for a fair comparison of stresses in the roof members. Structural analysis is run and critical constraint values for roof frame members are calculated and normalized with respect to the highest value for all members. Based on the availability of six different thicknesses for roof members, six roof regions or fewer are assigned using the following equations based on extensive numerical experimentation to achieve the most economical designs:

\[
g_{c,j}^{\text{norm}} = \frac{g_{c,j}}{g_{c,j}^{\text{max}}} \forall j, \text{where } 0 \leq g_{c,j}^{\text{norm}} \leq 1.0
\]  

\[
\text{Region} =
\begin{cases}
1 & 0 \leq g_{c,j}^{\text{norm}} \leq 0.1 \\
2 & 0.1 < g_{c,j}^{\text{norm}} \leq 0.3 \\
3 & 0.3 > g_{c,j}^{\text{norm}} \leq 0.5 \\
4 & 0.5 > g_{c,j}^{\text{norm}} \leq 0.7 \\
5 & 0.7 > g_{c,j}^{\text{norm}} \leq 0.9 \\
6 & 0.9 > g_{c,j}^{\text{norm}} \leq 1.0
\end{cases}
\]  

In the second phase, the GA is performed on the expanded optimization problem with additional regions including one for all cross members again until all constraints are satisfied and a global optimum solution is obtained. Phase two for topology optimization is essentially the same as for dimension optimization but with additional members. Figure 39 is a flow chart describing the combined dimension and topology optimization algorithm. Refer to Chapter 1, Figures 14 and 15 for flowcharts describing GA operations and phase two.
Figure 39: Flowchart Topology optimization phase one
3.3 Examples

The same roof structures, loading, and load combinations as in Chapter 1, Station 1 (Figures 3 through 6) and Station 2 (Figures 27 through 30), are considered for topology optimization.

3.3.a Example 1

Station 1 is initially divided into the six roof regions and five column regions shown in Figures 8 and 9 and values for all parameter defined in Chapter 1 (Figures 16 through 24) remain unchanged. The entire optimization took about five hours. Phase one optimization used an initial population of 50 designs and underwent 14 design iterations in about three hours and 40 minutes. The final phase one design weighs 201 kips and has nine overstressed members with the final regions and final cross member configuration shown in Figure 40.

Phase two optimization added 14 regions (seven roof regions and seven column regions, Figures 40 and 41) and 28 design variables and used an initial population of 30 designs. Roof region six and column regions seven through nine were eliminated during phase one. As such,
at the end of phase one there are a total of 12 roof regions, one cross member region, and nine column regions. Phase two took about one hour and 30 minutes and increased the phase one design weight to 220 kips. The convergence curve for the entire optimization process is shown in Figure 42 and the initial, phase one, and final solutions are presented in Table 3. The final solution is almost 3% heavier than the designer’s initial solution though it uses smaller HSS8x8 members than the initial solution’s HSS12x8 members. The structure is not lighter but it is thinner and sleeker looking which can be a desirable trait among modern architecture.

However, this is not the intent of this research and topology optimization will not be considered further for Station 1.

Station 1 behaves much like a typical arched roof structure and its design is based more heavily on major-axis bending than minor-axis bending. There is no variation in height along the length of the roof structure (Figure 5), so vertical loads do not act to pinch the diamond elements together along the shorter span (joints A to B, Figure 37). The small amount of minor-axis bending is caused by axial forces in the diamond frame elements. This example proves that a structure such as this does not benefit from the topology optimization presented in this research. Cross members allow for smaller and lighter diamond frame members though this weight reduction does not outweigh the increase in weight caused by the addition of members. The optimization process converged quickly but if allowed to continue with more strict stopping criteria, the cross-members would likely be eliminated from the structure.
Figure 42: Convergence curve for two-phase topology optimization for Station 1
Table 3: Final optimal solution for Station 1

<table>
<thead>
<tr>
<th>Region</th>
<th>Initial Solution</th>
<th>Phase 1 Solution</th>
<th>Phase 2 (Final) Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HSS Shape</td>
<td>No. Over-stressed Members</td>
<td>HSS Shape</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Original Roof Members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS8x8x3/16</td>
</tr>
<tr>
<td>2</td>
<td>HSS12x8x3/8</td>
<td>0</td>
<td>HSS8x8x3/16</td>
</tr>
<tr>
<td>3</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS8x8x3/8</td>
</tr>
<tr>
<td>4</td>
<td>HSS12x8x1/2</td>
<td>0</td>
<td>HSS8x8x5/8</td>
</tr>
<tr>
<td>5</td>
<td>HSS12x8x3/8</td>
<td>0</td>
<td>HSS8x8x3/8</td>
</tr>
<tr>
<td>6</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS8x8x1/2</td>
</tr>
<tr>
<td>Original Columns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>HSS16x20x2</td>
<td>0</td>
<td>HSS18x20x2</td>
</tr>
<tr>
<td>8</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS18x20x3/4</td>
</tr>
<tr>
<td>9</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS18x20x5/8</td>
</tr>
<tr>
<td>10</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS18x20x1.25</td>
</tr>
<tr>
<td>11</td>
<td>HSS16x20x1.5</td>
<td>0</td>
<td>HSS18x20x1</td>
</tr>
<tr>
<td>Cross</td>
<td>-</td>
<td></td>
<td>HSS4x6x3/16</td>
</tr>
<tr>
<td>New Roof Members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NewCols</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weight (kips)</td>
<td>214</td>
<td>201</td>
<td>220</td>
</tr>
</tbody>
</table>
3.3.b Example 2

Station 2 is initially divided into the six roof regions and six column regions shown in Figures 32 and 33 and values for all parameter defined in Chapter 1 (Figures 16 through 24) remain unchanged. Unlike Station 1, Station 2 varies in height along the length of the structure (Figure 30) and consequently the short span of the diamonds grid elements (joint A to B, Figure 37) is not horizontal. One joint is higher than the other causing vertical loads to pinch the diamonds together along this shorter span. Therefore, in contrast to Station 1, minor-axis bending plays a significant role in the design of Station 2. In this case cross members reduce the overall stresses, mainly minor-axis bending, and allow for smaller and lighter frame members.

A numerical test was performed for region reassignments for Station 2 considering its geometric complexity. When regions are reassigned to the roof structure based on Eqs. (26) and (27), region one consists of the smallest members and region six consists of the largest members, and there are typically no large changes in the region assignments from one set of three iterations to the next. Figure 43 shows the fitness convergence curve for a small optimization problem which uses 30 designs per iterations over 12 iterations for Station 2. This figure shows relatively smooth fitness convergence over the entire 12 iterations with no significant increase in fitness after every third iteration. This led to the decision of the “every third iteration” strategy for topology optimization and proves the described method to be effective for reassigning regions.
The entire optimization took about 14 hours and 20 minutes. Phase one design was performed using an initial population of 50 designs and took about eight hours and 50 minutes to undergo 28 design iterations. The phase one solution weighs 246 kips with eleven overstressed members. Figure 44 presents the final region assignments and cross member configuration. During phase one, roof region was eliminated resulting in a total of 11 regions (five roof and six column regions).
After analyzing results presented in Figure 44 it is apparent that the cross members improve the overall performance for Station 2 in more ways than were originally intended. The following structural benefits are observed: 1) reduction of minor-axis bending or pinching of diamonds as previously described, 2) added strength caused by cross members helping to tie the diagonal arches together, and 3) added stiffness in the portion of the roof which supports the cantilever. When comparing the initial roof regions (Figure 16) with the final regions (Figure 44), a concentration of heavier members is observed in the central portion of the roof which supports the cantilever (indicated by higher region numbers) in the initial design and not the final topology optimization design. The cross members reduce this concentration of high stresses.

Phase two optimization added 22 regions (16 roof regions and six column regions) and 36 design variables (Figures 44 and 45). At the beginning of phase two column region eleven...
was eliminated resulting in a total of 27 roof regions, two cross member regions, and 11 column regions. Phase two design was performed using an initial population of 30 design solutions and took about five hours and 26 iterations. The final weight of the structure is 252 kips which is 2% higher than the infeasible phase one solution but 8% lower than the designer’s initial solution. Figure 46 presents the convergence curve for the entire process and Table 4 presents a comparison of the final solution with the phase one and initial solutions. In this case topology optimization is beneficial and the decrease in weight from the smaller frame members outweighs the increase in weight from the additional cross members. The final design is 4% lighter than the dimension optimization final design from Chapter 1.

Figure 45: Column design final phase one (6-11) and phase two (*) regions of Station 2 (6 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 6)
Figure 46: Convergence curve for two-phase dimension and topology optimization for Station 2
Table 4: Final optimal solution for Station 2

<table>
<thead>
<tr>
<th>Region</th>
<th>Initial Solution</th>
<th>Phase 1 Solution</th>
<th>Phase 2 (Final) Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HSS Shape</td>
<td>No. Overstressed Members</td>
<td>HSS Shape</td>
</tr>
<tr>
<td>Original Roof Members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HSS14x10x1/4</td>
<td>0</td>
<td>HSS14x8x5/16</td>
</tr>
<tr>
<td>2</td>
<td>HSS14x10x5/16</td>
<td>0</td>
<td>HSS14x8x5/16</td>
</tr>
<tr>
<td>3</td>
<td>HSS14x10x3/8</td>
<td>0</td>
<td>HSS14x8x5/8</td>
</tr>
<tr>
<td>4</td>
<td>HSS14x10x1/2</td>
<td>0</td>
<td>HSS14x8x1/2</td>
</tr>
<tr>
<td>5</td>
<td>HSS14x10x3/4</td>
<td>0</td>
<td>HSS14x8x5/8</td>
</tr>
<tr>
<td>6</td>
<td>HSS14x10x1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Cross</td>
<td>-</td>
<td>-</td>
<td>HSS4x6x1/4</td>
</tr>
<tr>
<td>Original Columns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>HSS20x20x1</td>
<td>0</td>
<td>HSS16x20x1.25</td>
</tr>
<tr>
<td>8</td>
<td>HSS20x20x1/2</td>
<td>0</td>
<td>HSS20x18x5/8</td>
</tr>
<tr>
<td>9</td>
<td>HSS20x20x1</td>
<td>0</td>
<td>HSS18x20x1</td>
</tr>
<tr>
<td>10</td>
<td>HSS8x8x3/8</td>
<td>0</td>
<td>HSS8x8x5/16</td>
</tr>
<tr>
<td>11</td>
<td>HSS20x20x2</td>
<td>0</td>
<td>HSS16x18x1.75</td>
</tr>
<tr>
<td>12</td>
<td>HSS20x20x2</td>
<td>0</td>
<td>HSS20x16x1.5</td>
</tr>
<tr>
<td>New Roof Members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>HSS14x10x1/2</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>HSS14x10x3/8</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>HSS14x10x3/4</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>HSS14x10x3/8</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>HSS14x10x3/8</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>HSS14x10x3/8</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td>HSS14x10x3/4</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>HSS14x10x5/8</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td>HSS14x10x1/2</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td>HSS14x10x5/8</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td>HSS14x10x1/2</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td>HSS14x10x3/8</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td>HSS14x10x1/2</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td>HSS14x10x1/2</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td>HSS14x10x3/8</td>
</tr>
<tr>
<td>New Cols</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td>HSS8x6x3/8</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td>HSS10x14x1/2</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td>HSS8x8x3/8</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td>HSS8x8x1/4</td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td>HSS20x16x2</td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td>HSS20x16x2</td>
</tr>
<tr>
<td>New Cross</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>HSS6x6x0.25</td>
</tr>
<tr>
<td>Weight (kips)</td>
<td></td>
<td>275</td>
<td>246</td>
</tr>
</tbody>
</table>
Chapter 4: Shape Optimization

4.1 Problem Definition

The objective of this chapter is to extend the previous algorithms to shape optimization with the goal of achieving additional structural efficiencies. This is done by altering the geometry of the roof structure while simultaneously optimizing the roof member and column cross-sectional dimensions and the roof topology. Aesthetics is a significant consideration in the structures of the type considered in this research. As such, preserving the general form created by the architect is considered in the shape optimization algorithm. To achieve this, heuristic limits are imposed to avoid drastic or undesirable changes in their architectural form.

The structural forms considered in this research consist of a series of diagonally arranged arches in two directions along the length of the structure (Figure 47). The arches along the width are the primary arches, and the arches at an inclination of about 60° relative the longitudinal axis in the top view are the secondary arches. The shape and depth of the primary arches are altered to optimize the load-carrying capacity of the roof structure and minimize its weight.

Figure 47: Top view of Station 1 showing primary and secondary arches
4.2 Optimization Problem Formulation using a Genetic Algorithm

Shape optimization poses a difficult challenge when using an automated process such as a GA. It is relatively uncommon among the four types of structural design optimization problems (dimension, topology, material, and shape) because of the difficulties associated with achieving a smooth final result. This is sometimes done with additional constraints; however, in the methodology presented in this chapter, the optimization problem formulation is the same for shape optimization as for topology optimization defined in Chapter 3. The objective function is defined in Chapter 3, Eq. (24) and constraint equations are defined in Chapter 2, Eqs. (2) through (10).

The chromosome or string representing each design is again lengthened to account for shape optimization design variables. It consists of three portions: 1) discrete dimension variables, 2) binary topology variables, and 3) binary shape variables. The third portion of the string, boldfaced in Figure 48, consists of floating point shape optimization variables transformed to binary strings whose lengths depend on the desired accuracy and the range of possible values. Figure 48 shows a string representing an example solution with four roof regions, one column region, three out of five possible cross members, and two 4-bit shape optimization variables. Two methods of altering the geometry of the structure are presented in this chapter, one is rather simple to be used for roof structures with relatively regular geometries, and the other for more complicated geometries. Design variables are problem specific and their lengths are specific to each method as well as to each problem.
Figure 48: Example string representing a solution with four roof regions, one column regions, three out of five possible cross members, and 2 4-bit shape optimization variables.

To begin GA optimization, \( N_p \) initial design solutions are generated around the initial design as described in Chapter 2. Binary shape variables are generated randomly by creating a binary strings with randomly arranged “1”s and “0”s.

A combination of two different crossover operations in this chapter: uniform and one-point crossover. Uniform crossover is used for actual or binary discrete variables and one-point crossover is used for binary floating point variables. When a binary string is used to represent one floating point design variable, the one-point crossover operation swaps one section of a parent string with the same section in another parent string to create offspring designs. That
section is placed at the end of the string to encompass the lesser significant bits (bits at the beginning of the string are considered to be more significant because they are multiplied by two raised to a higher power when converting to decimal numbers). The length of the section is determined randomly and can be anywhere from one bit to the entire length of the binary string. An example of one-point crossover is presented in Figure 49 for two 4-bit design variables.

![Figure 49: Example of the one-point crossover operation for two 4-bit design variables](image)

The shape optimization algorithm presented in this research changes only the vertical coordinates of roof joints. Two methods are presented.

### 4.2.a Method 1: Implicit Variation of Vertical Coordinates

The first method alters the vertical coordinates of roof joints implicitly. The first step of this method is to consider the primary arches in the two-dimensional (2D) planes where they appear as arches and identify the joints with fixed locations due to boundary conditions, vertical and horizontal clearances, and/or designer preferences. An example arch is shown in Figure 50 where joints 1 and 11 are fixed due to boundary conditions. The next step is to create a polynomial curve which passes through all fixed joints. The order of the polynomial should be equal to or greater than the number of fixed joints as described later. As an example, consider the following the second-order polynomial for the arch curve:

\[
Z = AY^2 + BY + C
\]  

(28)
where \( Z \) is the set of Z-coordinates, \( Y \) is the set of Y-coordinates for all joints on the arch, and coefficients \( A \), \( B \), and \( C \) are either *free* or *fixed* parameters. The number of *free* parameters is equal to the order of the polynomial minus the number of fixed joints plus one. When there are two fixed joints and a second-order polynomial is used, then two of the three parameters \( (A, B, \text{ and } C) \) are *fixed* parameters determined from the fixed joint coordinates. The third parameter is the *free* parameter and a design variable in the GA optimization process. If \( A \) is chosen as the *free* parameter in the example arch of Figure 50, the other two coefficients \( B \) and \( C \) are found by solving the following system of linear equations:

\[
\begin{bmatrix}
Y_1 \\
Y_{11}
\end{bmatrix}
\begin{bmatrix}
B \\
C
\end{bmatrix} =
\begin{bmatrix}
Z_1 - AY_1^2 \\
Z_{11} - AY_{11}^2
\end{bmatrix}
\]

(29)

Where \((Y_1,Z_1)\) and \((Y_{11},Z_{11})\) are the coordinates of the fixed points 1 and 11.

Figure 50: Example arch for shape optimization methods 1 and 2
Once all parameters are defined, the Z-coordinates for joints 1 through 11 are determined from Eq. (28) or a similar polynomial equation depending on the desired order. An example polynomial curve is shown with a dotted curve in Figure 50 using an arbitrary A value. Additional parameters can be used as design variables by adding a ΔB, for example, to B obtained from Eq. (29). Once B changes, the remaining fixed parameters must be found using a similar system of linear equations but with one fewer equation and one fewer unknown. This concept will be clarified when example 1 is presented. This method is effective for roof structures with more regularity and repetition in its form.

4.2.b Method 2: Explicit Inclusion of Vertical Coordinates

In this method the vertical coordinates of roof joints are altered explicitly using a Z-coordinate multiplier, H, for one or a few preselected joints on the primary arch (there can be multiple multipliers). Multiple multipliers can be used (H₁, H₂, etc.) each a design a variable of the GA optimization. These preselected joints are referred to as design joints. The order of the polynomial curve is determined so that there are no free parameters. In this method, the multipliers are the design variables, while in method 1 the free parameters are the design variables. In the example arch of Figure 50 there are two fixed joints, 1 and 11, and one design joint, 7, and therefore a second-order polynomial (Eq. 28) is used where the values of all three fixed parameters are determined using the following system of linear equations:

\[
\begin{bmatrix}
Y₁² & Y₁ & 1 \\
Y₇² & Y₇ & 1 \\
Y_{11}² & Y_{11} & 1
\end{bmatrix}
\begin{bmatrix}
A \\
B \\
C
\end{bmatrix}
= 
\begin{bmatrix}
Z₁ \\
Z₇ \\
Z_{11}
\end{bmatrix}
\]

(30)

where Z₇ is the initial Z-coordinate for design joint 7 multiplied by design variable H. An example polynomial curve is shown in Figure 50 with an arbitrary value of H=1.1. This method is
developed for roof structures with little regularity and repetition in its form where the first method becomes complicated.

Depending on the complexity and irregularity of the structural form one polynomial curve can be used for all arches or a different curve can be used for each individual arch. Numerical experimentation is required to determine a reasonable range of design variable values depending on architectural forms and limitations.

Shape optimization is performed during phase one only and regions are again re-established every three iterations. Phase two remains unchanged from Chapters 2 and 3. Figure 51 is a flow chart describing the combined dimension, topology, and shape optimization algorithm. Flowcharts describing GA operations and phase two are presented in Chapter 2, Figures 14 and 15. The algorithm described in Figures 14, 15, and 51 was developed using MATLAB. MATLAB coding for combined dimension, topology, and shape optimization is presented in the Appendix.
Figure 51: Flowchart for Shape optimization phase one
4.3 Examples

The same roof structures, loading, and load combinations as in Chapter 1, Station 1 (Figures 3 through 6) and Station 2 (Figures 28 through 31), are considered for shape optimization. Method one (Section 4.2.a) is used for Station 1 because of its regularity and Method two (Section 4.2.b) is used for Station 2 because of its complexity.

4.3.a Example 1

Shape optimization method one is used for Station 1. Due to its geometric regularity, one third-order polynomial is used for all 16 primary arches along the length of the structure (Figure 47). The global coordinate system for the structure is shown in the upper left corner of Figure 47 with X and Y axes along the length and width of the roof respectively, and the Z-axis in the vertical direction. The initial form of the arch is shown with solid lines in the YZ-plane in Figure 52. Boundary conditions (fixed-connections) require that joints 1 and 8 remain fixed in their locations, and vertical and horizontal train clearances require that joint 2 remain fixed as well. Since joints 1 and 2 are both fixed, the line between joints 1 and 2 remains unchanged and joint 1 is not considered in the polynomial approximation. A set of $Z$-coordinates is obtained for all arches between joints 2 and 9 using a third-order polynomial equation of the following form:

$$Z = AY^3 + BY^2 + CY + D$$

(31)

where $Y$ is the set of constant Y-coordinates for all arches. In this example, coefficients $A$, $B$, $C$, and $D$ and the two $Y$ and $Z$ sets are the same for all arches.
Since the third-order polynomial must pass through fixed joints 2 and 8, two out of the four parameters (A, B, C, and D) are fixed and two are free. Initially, the edge point 9 is assumed to be fixed reducing the number of free parameter to one. The coefficient A is chosen to be the free parameter or shape design variable and is varied to alter the depth of the arch. Coefficients B, C, and D are found by solving the following system of linear equations:

$$\begin{bmatrix}
Y_2^2 & Y_2 & 1 \\
Y_8^2 & Y_8 & 1 \\
Y_9^2 & Y_9 & 1
\end{bmatrix} \begin{bmatrix}
B \\
C \\
D
\end{bmatrix} = \begin{bmatrix}
Z_2 - AY_2^3 \\
Z_8 - AY_8^3 \\
Z_9 - AY_9^3
\end{bmatrix}$$

(32)

The coefficients B, C, and D are determined using Eq. (32) for any given A obtained via the GA optimization. An example third-order polynomial is shown in Figure 52.

Next, in order to make small adjustments to the shape of the arch, point 9 is allowed to move in the vertical direction. This is done by using another design variable, ΔC, which is added to C obtained from Eq. (32). The purpose of ΔC is to change the shape of the arch by dropping the cantilever down (point 9) and consequently rounding out the arch. To avoid undesirable
and aesthetically unacceptable double-curvature in the roof, upward movement of point 9 is not allowed. Figure 53 portrays examples of different shapes using the following maximum and minimum design variable values:

\[-11 \times 10^{-7} \leq A \leq -4 \times 10^{-7}\]  \hspace{1cm} (33)

\[-0.3 \leq \Delta C \leq 0\]  \hspace{1cm} (34)

where the shallowest possible arch coincides with the initial form. The roof cannot decrease in height to ensure that vertical train clearances already established remain met and because deeper arches will carry the load more efficiently than shallower arches. The limits in Eqs. (33) and (34) were obtained by numerical experimentation with the goal of preserving the overall architectural form.
Figure 53: Upper and lower limits for the primary arch using minimum and maximum values for design variables A and ΔC. Example 1 uses $A_{\text{max}}$ and $\Delta C_{\text{min}}$, Example 2 uses $A_{\text{min}}$ and $\Delta C_{\text{min}}$, Example 3 uses $A_{\text{max}}$ and $\Delta C_{\text{max}}$, and Example 4 uses $A_{\text{min}}$ and $\Delta C_{\text{max}}$.

The length of the binary strings used to represent design variables A and ΔC depends on each variable’s range of possible values as well as the desired accuracy. The string must be long enough to represent enough values but short enough to keep a relatively small search space so that a global solution can be found in a reasonable amount of time. Based on numerical experimentation, an 8-bit string deemed to be suitable to represent the two design variables, A and ΔC. An 8-bit string allows for $2^8=256$ variations of each design variable and $256 \times 256=65,536$ arch depth and shape variations within the eight foot height difference from the shortest possible arch to the tallest possible arch (Figure 53).

Since topology optimization proved to be ineffective for Station 1 in Chapter 3, it is not included in this example. Station 1 is initially divided into the six roof regions and five column
regions shown in Figures 8 and 9 and values for all parameter defined in Chapter 1 (Figures 16 through 24) remain unchanged.

The entire optimization process took about 13 hours. This process is especially time consuming for shape optimization. Redrawing the model to account for changed coordinates of each of the 113 free joints takes about 24 seconds for one design and adds 24x50=1200 seconds or 20 minutes to each iteration of 50 designs.

Phase one optimization took about seven hours and 20 minutes for each population of 50 designs to undergo 16 iterations. The phase one solution weighs 166 kips with five overstressed members. Final roof and column regions are shown in Figures 54 and 55, respectively. A 3D view of the final shape is shown in Figure 56 and a side view is shown in Figure 57 with the initial form. The design evolution is presented in Figure 58 which shows the roof shape from every fourth iteration.

![Figure 54: Roof design final phase one (1-6) and phase two (*) regions of Station 1 (10 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 10)
Figure 55: Column design final phase one (7-11) and phase two (*) regions of Station 1
(2 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 2)

Figure 56: 3D perspective view of the final optimal shape for Station 1

Figure 57: Side view of the final optimal shape (shown in black) with the initial shape (shown in blue) for Station 1
Figure 58: Shape design evolution for Station 1

The initial arch design is shallow causing vertical loads to impose a high thrust on the columns. The result is a deeper arch which improves its load-carrying capacity and reduces this thrust. Design variable $\Delta C$ is very close to zero, meaning that the height of the end of the cantilever did not change. This reduces the tension in cantilevered members which acts to increase the outward pull on the top of the columns in addition to the outward thrust imposed by the arches. The roof increased in height by about 6.2 feet.
Phase two optimization added 12 regions (10 roof regions and two column regions, Figures 54 and 55). It took about five and a half hours for each population of 30 designs to undergo 27 iterations. Feasibility was not met until the sixth iteration of phase two. The final solution weighs 166 kips which is 22% lighter than the designer’s initial solution and 10% lighter than the final design achieved from dimension optimization in Chapter 2. The convergence curve for the entire process is presented in Figure 59 and the final solution is presented in Table 5.
Figure 59: Convergence curve for two-phase shape optimization for Station 1
<table>
<thead>
<tr>
<th>Region</th>
<th>Original Roof Members</th>
<th>Original Columns</th>
<th>New Roof Members</th>
<th>New Cols</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS8x6x1/4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>HSS12x8x3/8</td>
<td>0</td>
<td>HSS8x6x1/4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS8x6x5/16</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>HSS12x8x1/2</td>
<td>0</td>
<td>HSS8x6x1/2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>HSS12x8x3/8</td>
<td>0</td>
<td>HSS8x6x5/8</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>HSS12x8x1/4</td>
<td>0</td>
<td>HSS8x6x3/4</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>HSS16x20x2</td>
<td>0</td>
<td>HSS16x20x2</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS16x20x1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>1HSS16x20x5/8</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS16x20x5/8</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>HSS16x20x1.25</td>
<td>0</td>
<td>HSS16x20x1.25</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>HSS8x6x3/8</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td>HSS8x6x3/8</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td>HSS8x6x1/2</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>HSS8x6x1/2</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>HSS8x6x3/4</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>HSS8x6x5/16</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>HSS8x6x5/8</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td>HSS8x6x5/16</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>HSS8x6x3/4</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td>HSS8x6x1</td>
<td>0</td>
</tr>
<tr>
<td>Weight (kips)</td>
<td>214</td>
<td>166</td>
<td>166</td>
<td></td>
</tr>
</tbody>
</table>
Station 2 consists of 15 primary arches, all different shapes, and consequently each must be considered separately (Figure 60). If shape optimization for this example were performed using method 1, constraints would need to be added to Eq. (15) to control the depth and shape of each arch to preserve the overall architectural form. Instead, the second shape optimization method is used. One design variable is the Z-coordinate multiplier, \( H \), for the two preselected rows of design joints shown in Figure 60. Another design variable, \( c \), is multiplied by \( H \) for the second row of primary arches only. The purpose of \( c \) is to allow each row of design joints to use a different multiplier but to ensure that they are not drastically different. An example of the undesirable effects of too drastic a difference between row 1 and row 2 multipliers is shown as Example 1 of Figure 61. Requirements are set so that \( c \leq 1.0 \) and \( cH \geq 1.0 \) to keep the roof above its initial height and to avoid increasing the potential for arch buckling. Allowing \( cH < 1.0 \) introduces an undesirable dip or double curvature shown as Example 2 in Figure 61. \( H \) and \( cH \) are multiplied by the Z-coordinate of all design joints in rows one and two, respectively, ensuring that the profile does not deviate from the original form shown in Figure 31. The following equations are used to obtain new Z-coordinates for the design joints:

\[
Z'_f = \begin{cases} 
HZ_{f,o} & \text{for joints in row 1} \\
cHZ_{f,o} & \text{for joints in row 2}
\end{cases}
\]

(35)

where \( Z'_f \) is the new Z-coordinate for design joint \( f \) and \( Z_{f,o} \) is the initial Z-coordinate for joint \( f \).
Z-coordinates for all other joints are determined from polynomial equations which pass through design joints and fixed joints. Arches 1 through 9 (Figure 60) have two fixed ends. An example of such an arch, arch 7, is presented in Figure 62 where joints 1 and 11 are fixed due to boundary conditions and joints 2 and 10 are fixed due to vertical and horizontal train clearances.
As such, the lines between joints 1 and 2 and joints 10 and 11 remain unchanged and joints 1 and 11 are not considered in the polynomial approximation. A third-order polynomial equation of the following form is used to define Z-coordinates for arches 1 through 10 with two fixed joints (2 and 10) and two design joints (4 and 8):

\[
Z'_m = AY_m^3 + B Y_m^2 + CY_m + D
\]  

(36)

where \(Z'_m\) is the set of new Z-coordinates for arch m, \(Y_m\) is the set of constant Y-coordinates for arch m, and A, B, C, and D coefficients are determined from the four points with known coordinates (two fixed joints and two design joints) using the following system of linear equations:

\[
\begin{bmatrix}
Y_2^3 & Y_2^2 & Y_2 & 1 \\
Y_4^3 & Y_4^2 & Y_4 & 1 \\
Y_8^3 & Y_8^2 & Y_8 & 1 \\
Y_{10}^3 & Y_{10}^2 & Y_{10} & 1
\end{bmatrix}
\begin{bmatrix}
A \\
B \\
C \\
D
\end{bmatrix}
=
\begin{bmatrix}
Z_2 \\
Z_4' \\
Z_8' \\
Z_{10}'
\end{bmatrix}
\]  

(37)
In the second method used for this example, a different polynomial equation is defined for each arch with its own set of coefficients A, B, C, and D.

Arches 10 through 15 (Figure 60) have one fixed end and one free end. An example of such an arch, arch 13, is presented in Figure 63 where joint 2 is fixed due to train clearances and joints 3 and 7 are design joints. In this case a second-order polynomial of the following form is used to define all other Z-coordinates for arches 11 through 15:

\[
Z'_m = EY^2_m + FY_m + G
\]

(38)

where E, F, and G coefficients are determined from the three joints with known coordinates (one fixed joint and two design joints) using the following system of equations:

\[
\begin{bmatrix}
Y_2^2 & Y_2 & 1 \\
Y_3^2 & Y_3 & 1 \\
Y_7^2 & Y_7 & 1
\end{bmatrix}
\begin{bmatrix}
E \\
F \\
G
\end{bmatrix}
= 
\begin{bmatrix}
Z_2' \\
Z_3' \\
Z_7'
\end{bmatrix}
\]

(39)

Again, a different polynomial equation is defined for each arch with its own set of coefficients E, F, and G.
Figure 63: An example primary arch, arch 13, showing a third-order polynomial curve passing through fixed joints and design joints.

The following minimum and maximum limits for design variables $H$ and $c$ were determined with the goal of preserving the architectural form:

$$1.0 \leq H \leq 1.15 \quad (40)$$

$$0.95 \leq c \leq 1.0 \quad (41)$$

Example shapes using the aforementioned maxima and minima are shown for arch 7 with two fixed ends in Figure 64. Example shapes for arch 13 with one fixed end and one free end are shown in Figure 65. The effects of design variable $c$ are better represented in Figure 66 which displays three 3D models of the structure: A) $H=1.0$ and $c=1.0$ (the initial form), B) $H=1.15$ and $c=0.95$, and C) $H=1.15$, $c=1.0$. The top of roof C leans to the left and the top of roof B leans to the right. A $c$-value resulting in a centered arch similar to initial roof A is desired both structurally and aesthetically.
Figure 64: Upper and lower limits for an example primary arch, arch 7, using minimum and maximum values for design variables $H$ and $c$. Example 1 uses $H_{\text{max}}$ and $c_{\text{min}}$, Example 2 uses $H_{\text{max}}$ and $c_{\text{max}}$, and Example 3 uses $H_{\text{min}}$ and $c_{\text{max}}$.

Figure 65: Upper and lower limits for an example primary arch, arch 13, using minimum and maximum values for design variables $H$ and $c$. Example 1 uses $H_{\text{max}}$ and $c_{\text{min}}$, Example 2 uses $H_{\text{max}}$ and $c_{\text{max}}$, and Example 3 uses $H_{\text{min}}$ and $c_{\text{max}}$. 
A 6-bit binary string was determined to adequately represent design variables $H$ and $c$ based on numerical experimentation. This allows for $2^6 = 64$ variations of each design variable and $64 \times 64 = 4,096$ roof design variations within the 7-foot vertical span between upper and lower limits.

Station 2 is initially divided into the six roof regions and six column regions shown in Figures 33 and 34 and values for all parameters defined in Chapter 1 (Figures 16 through 25) remain unchanged. Topology optimization is included in this example.

The entire process of simultaneous dimension, topology, and shape optimization took about 17 hours and 40 minutes. Phase one used a population of 50 designs and underwent 27 iterations in about 13 hours and resulted in a design which weighs 201 kips and has 14 overstressed members. The final design regions and cross member configuration are displayed in Figures 67 and 68. Figure 69 shows a 3D perspective view of the final shape and cross...
member configuration for Station 2, and Figures 70 and 71 show side views along the length and width, respectively, of the final shape with the initial shape. Figures 69 though 71 include another view of the roof structure showing the wood panels to better portray the smoothness and architectural form. Figure 72 presents the design evolution showing the roof shape from iterations 3, 9, 15, 21, and 27. The result is deeper arches which increase the load-carrying capacity, decrease thrust on the columns, and provide stronger arches to support for the cantilever. Changes in arch depths are subtle in Figure 72. Arches are relatively shallow in iterations 3, 9, and 15 and begin to get deeper in iteration 21.

Figure 67: Roof design final phase one (1-6) and phase two (*) regions of Station 2 (24 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 24)
Figure 68: Column design final phase one (7-12) and phase two (*) regions of Station 2
(3 asterisks indicate members assigned additional regions in phase two increasing the number of roof regions by 3)
Figure 69: 3D perspective view of the final optimal shape showing A) the steel frame only and B) the steel frame with wood panels for Station 2
Figure 70: Side view along the length of Station 2 of the final optimal shape (shown in black) showing A) the steel frame with the initial shape (shown in blue) and B) the steel frame with wood panels.
Figure 71: Side view along the width of Station 2 of the final optimal shape (shown in black) showing A) the steel frame with the initial shape (shown in blue) and B) the steel frame with wood panels.
Figure 72: Shape design evolution for Station 2
Phase two took about four hours and 40 minutes to undergo 23 iterations each with populations of 30 designs. A total of 27 regions (24 roof regions and three column regions, Figures 66 and 67) and 33 design variables were added during phase two optimization. The final design weighs 209 kips which is 24% lighter than the designer’s initial design and 16% lighter than the final topology optimization design from Chapter 3. The convergence curve for the entire process is shown in Figure 73 and the final solution is presented in Table 6.
Figure 73: Convergence curve for two-phase shape optimization for Station 2
Table 6: Final optimal solution for Station 2

<table>
<thead>
<tr>
<th>Region</th>
<th>Initial Solution</th>
<th>Phase 1 Solution</th>
<th>Phase 2 (Final) Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HSS Shape</td>
<td>No. Over-stressed Members</td>
<td>HSS Shape</td>
</tr>
<tr>
<td>Original Roof Members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>HSS14x10x1/4</td>
<td>0</td>
<td>HSS10x12x5/16</td>
</tr>
<tr>
<td>2</td>
<td>HSS14x10x5/16</td>
<td>0</td>
<td>HSS10x12x3/16</td>
</tr>
<tr>
<td>3</td>
<td>HSS14x10x3/8</td>
<td>0</td>
<td>HSS10x12x5/16</td>
</tr>
<tr>
<td>4</td>
<td>HSS14x10x1/2</td>
<td>0</td>
<td>HSS10x12x5/8</td>
</tr>
<tr>
<td>5</td>
<td>HSS14x10x3/4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Cross</td>
<td></td>
<td>-</td>
<td>HSS4x8x3/16</td>
</tr>
<tr>
<td>Original Columns</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>HSS14x10x1/2</td>
<td>0</td>
<td>HSS14x18x1/2</td>
</tr>
<tr>
<td>7</td>
<td>HSS20x20x1</td>
<td>0</td>
<td>HSS16x20x1</td>
</tr>
<tr>
<td>8</td>
<td>HSS20x20x1/2</td>
<td>0</td>
<td>HSS14x18x5/8</td>
</tr>
<tr>
<td>9</td>
<td>HSS20x20x1</td>
<td>0</td>
<td>HSS12x20x3/4</td>
</tr>
<tr>
<td>10</td>
<td>HSS8x8x3/8</td>
<td>0</td>
<td>HSS6x8x1/2</td>
</tr>
<tr>
<td>11</td>
<td>HSS20x20x2</td>
<td>0</td>
<td>HSS20x16x1.25</td>
</tr>
<tr>
<td>12</td>
<td>HSS20x20x2</td>
<td>0</td>
<td>HSS14x12x2</td>
</tr>
<tr>
<td>New Roof Members</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Region</td>
<td>Initial Solution</td>
<td>Phase 1 Solution</td>
<td>Phase 2 (Final) Solution</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td></td>
<td>HSS Shape</td>
<td>No. Overstressed Members</td>
<td>HSS Shape</td>
</tr>
<tr>
<td>New Roof</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>New Cols</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weight (kips)</td>
<td>275</td>
<td>201</td>
<td>209</td>
</tr>
</tbody>
</table>
Chapter 5: Conclusion and Future Work

The two-phase genetic algorithm developed for this research optimized two complex one-of-a-kind structures and achieved considerably lighter and more structurally efficient designs. The algorithm was first developed for dimension optimization and resulted in 12% and 4% weight savings for Stations 1 and 2, respectively. The algorithm was then extended to topology optimization and achieved additional weight savings of 4% for Station 2. The algorithm was extended further to perform shape optimization which performed simultaneous dimension, topology, and shape optimization and achieved additional weight savings of 10% and 16% for Stations 1 and 2, respectively. The final algorithm resulted in optimum designs which are 22% and 24% lighter than the initial designs created in a structural design office for the two roof structures.

The algorithm yielded a global optimum solution in less than 19 hours on a standard workstation machine with a 2.83 GHZ dual core processor, a relatively short amount of time considering the complexity of both the structures and the optimization problem. The following strategies allowed the problem to be solved in a reasonable amount of time without the help of supercomputers, a common aid in large structural design problems:

1. Implementing a two-phase method which first finds a slightly under-designed and lightweight solution which is close to but not entirely feasible, then performing a more fine-tuned search with more strict criteria to achieve a global optimal and fully feasible minimum weight design,
2. Using a more intuitive starting point by creating the initial population around an initial design using a normal probability distribution rather than at random, and
3. Strategically reducing the design space by ruling out impractical designs.
The initial design of the structure was developed iteratively over a period of days and allowed the GA to benefit from a relatively lightweight starting point. Had a rough initial design or no initial design been used, the GA might have taken longer from starting further away from an optimum design but would have achieved similar results. An advantage to GA’s is that they are non-deterministic, so that each run takes a different route depending on the size and design variable values of the initial population. Therefore, an initial design is a benefit but not a requirement for a GA.

The behavior of the complex free-form structures considered is difficult to predict and the designer may not have the time or means to create a good initial design. Any local changes affect the entire structure, so one must look at the structure as a whole. For this reason an optimization method which performs a global search, much like the GA developed, is necessary for large structural design problems.

Future work includes the following:

1. Analyzing the effects of different starting points (a rough initial design, no initial design, etc.),
2. Enhancing the topology optimization algorithm to study the effects of the arrangement of members or the angle created by the diamond-shape grid elements (adjusting the angles of the diagonal arches),
3. Performing cost optimization and comparing the results to minimum weight design, and
4. Solving the same problem using a different heuristic optimization method such as particle swarm optimization and comparing results.
References


ASCE (2010), Minimum Design Loads for Buildings and Other Structures, ASCE7-10, American Society of Civil Engineers.


26:3, pp. 207-224.


Main Program – Phase One

% Genetic Algorithm Dimension Optimization of Steel Space-Frame Roof Structures

clear;
cclc;
tic

% Obtain user input

% nReg = no. roof member regions + no. column pairs = 6 + 5 = 11
nReg_r = input('Number of roof regions (NOTE: must be labelled "1", "2", etc.): ');
nReg_c = input('Number of column regions (NOTE: must be labelled "1", "2", etc.): ');
nArc = input('Number of arches: ');
nReg = nReg_r + nReg_c;
nPop = input('Number of design solutions in population for phase 1: ');
nIter = input('Maximum number of generations (design iterations) for phase 1: ');
run1 = input('Is this the first run for this problem (1 = yes, 0 = no)?: ');
nDB = nReg_c + 1;
nDv = nReg + 2*nDB; % nDv = nReg s, roof d & b, column d & b
nBin = 12;
population = 1;
conv = 1;
phase = 1;
re_assign = 1;
SOLN2 = zeros(0,0);

% initialize SAP2000

feature('COM_SafeArraySingleDim', 1);
feature('COM_PassSafeArrayByRef', 1);
SapObject = actxserver('Sap2000v15.SapObject');
SapObject.ApplicationStart;
SapModel = SapObject.SapModel;

% open existing SAP2000 model and unlock

FileName = strcat(cd, '\Bare Model.sdb');
SapModel.File.OpenFile(FileName);
SapModel.SetModelIsLocked(0);

% get number of frames and frame lengths from SAP2000 model

nMem = SapModel.FrameObj.Count;
x1 = zeros(1,nMem);
y1 = zeros(1,nMem);
z1 = zeros(1,nMem);
x2 = zeros(1,nMem);
y2 = zeros(1,nMem);
z2 = zeros(1,nMem);
L = zeros(1,nMem);

for  i = 1:nMem

  % Obtain frame element lengths from SAP2000
  % Obtain I-end and J-end points associated with current frame
  % element
  FrameID = num2str(i);
  Point1 = ' ';
  Point2 = ' ';

  [ret Point1 Point2] = SapModel.FrameObj.GetPoints(char(FrameID),
Point1, Point2);

  % Obtain Point1 coordinates
  x = zeros(1,1,'double');
y = zeros(1,1,'double');
z = zeros(1,1,'double');

  [ret x1(i) y1(i) z1(i)] = SapModel.PointObj.GetCoordCartesian(char(Point1), x, y, z);

  % Obtain Point2 coordinates
  x = zeros(1,1,'double');
y = zeros(1,1,'double');
z = zeros(1,1,'double');

  [ret x2(i) y2(i) z2(i)] = SapModel.PointObj.GetCoordCartesian(char(Point2), x, y, z);

  % Calculate frame length
  L(i) = sqrt((x1(i) - x2(i))^2 + (y1(i) - y2(i))^2 + (z1(i) - z2(i))^2);

end

% Get group assignments from SAP2000 (NOTE: Groups never change throughout optimization process)
if run1 == 1
RegID = zeros(nMem,2); % matrix, (member no.s in order)x(region no.s)
r2 = 0; % index counter

for i = 1:nReg

    RegRef = num2str(i); % name of group
    NumberItems = 0; % number of members in specified group
    ObjectType = cellstr(' '); % array, object type of each member in the group, 2=frame object
    ObjectName = cellstr(' '); % array, name of each member in specified group
    [ret NumberItems ObjectType ObjectName] = SapModel.GroupDef.GetAssignments(char(RegRef), NumberItems, ObjectType, ObjectName);

    r1 = r2 + 1; r2 = r2 + NumberItems;
    RegID(r1:r2,1) = str2double(ObjectName); % Frame numbers
    RegID(r1:r2,2) = i; % Group number

end % for i = 1:nReg

RegID = sortrows(RegID,1); % Sort ascending by frame number

else

    Tbl = strcat('A1:B',num2str(nMem)); % for nPop = 50 ONLY
    [NUM,TXT,RAW] = xlsread('Solution.xls','Sheet2',char(Tbl));
    RegID = cell2mat(RAW);

end

% Obtain arch coordinates

JntsPerArc = zeros(nArc,1);

for i = 1:nArc

    ArchID = strcat('A',num2str(i));
    NumberItems = 0; % number of members in specified group
    ObjectType = cellstr(' '); % array, object type of each member in the group, 2=frame object
    ObjectName = cellstr(' '); % array, name of each member in specified group
    [ret JntsPerArc(i) ObjectType ObjectName] = SapModel.GroupDef.GetAssignments(char(ArchID), NumberItems, ObjectType, ObjectName);

end

arc_coords = zeros(max(JntsPerArc),3,nArc);
profile_1 = zeros(nArc,2); profile_2 = profile_1;
profile_1(:,1) = [4*ones(1,12) 3 2 1]';
profile_2(:,1) = [8*ones(1,12) 7 6 5]';

jnt = 1;

for i = 1:nArc
    % Create 3d matrix (1 page for each arch)
    for j = 1:JntsPerArc(i)
        JointID = num2str(jnt);
        x = zeros(1,1,'double');
        y = zeros(1,1,'double');
        z = zeros(1,1,'double');

        [ret arc_coords(j,1,i) arc_coords(j,2,i) arc_coords(j,3,i)] =
        SapObject.SapModel.PointObj.GetCoordCartesian(char(JointID), x, y, z);
        jnt = jnt + 1;
    end

    profile_1(i,2) = arc_coords(profile_1(i,1),3,i);
    profile_2(i,2) = arc_coords(profile_2(i,1),3,i);
end

H_min = 1; H_max = 1.3;
c_min = 0.95; c_max = 1.0;

% Close bare model
SapObject.ApplicationExit(false());
SapModel = 0;
SapObject = 0;

% Open Ground Model and obtain connectivity data
[connectivity Lx nXm Xms dlx blx tlx] = ground_model;

nDv_DIM = nDv; nDv_TOPO = 3 + nXm;

% Get user defined solution from current SAP2000 model
[d1 b1 t1 binary1 k] = initial_soln(RegID, nDB, nReg, nReg_r, nReg_c, nMem, connectivity, nXm);

% For 1st iteration, obtain user defined design solution and create random design population
% Otherwise, use GA offspring as design solution
\begin{verbatim}
des_soln = zeros(nPop,nDv,nIter);
fitness = zeros(nIter,nPop);
f_objective = zeros(nIter,nPop);
f_constraint = zeros(nIter,nPop);
design = zeros(nDv,nIter);
average_weight = zeros(1,nIter);
average_fitness = zeros(1,nIter);
best_weight = zeros(1,nIter);
overstressed = zeros(1,nIter);
zs = zeros(max(JntsPerArc),nArc,nPop);
design_z = zeros(max(JntsPerArc),nArc,nIter);

while population <= nIter
    if population == 1 && run1 == 1
        \%
        \% Convert dimension data in inches to binary data and input as
        \% first design solution
        init_soln = [d1 b1 t1 dx b1x t1x binary1 zeros(1,nBin)];
        \%
        \% Create random binary strings and input as remaining design
        \% solutions
        des_soln(:,:,population) = random_solution(nPop, nReg, nDv,
        nDv_DIM, nDB, nXm, nBin, k, init_soln, des_soln(:,:,population),
        phase);
        \% set size of population to variable nPop for future
        \% calculations
    elseif population == 1 && run1 == 0
        Tbl = strcat('A1:AX',num2str(nDv)); \% for nPop = 50 ONLY
        [NUM,TXT,RAW] = xlsread('Solution.xls','Sheet1',char(Tbl));
        des_soln(:,:,population) = cell2mat(RAW)';
    else \% population > 1
        des_soln(:,:,population) = offspring;
    end \% end if population == 1

    \% Open New Model
    feature('COM_SafeArraySingleDim', 1);
    feature('COM_PassSafeArrayByRef', 1);
    SapObject = actxserver('Sap2000v15.SapObject');
    SapObject.ApplicationStart;
    SapModel = SapObject.SapModel;
    FileName = strcat(cd,'\New Model.sdb');
    SapModel.File.OpenFile(FileName);
\end{verbatim}
SapModel.SetModelIsLocked(0);

% Assign HSS dimensions to SAP2000 model, run analysis, and determine fitness for each solutions

nOS_mem = zeros(nMem + nXm, nPop);

for soln = 1:nPop
    % Skip analysis/calculation for repeated solutions
    if soln > 1
        while soln <= nPop
            if any(des_soln(soln,:,population) - des_soln(soln - 1,:,population)) == 0 % any returns 1 if any element is nonzero
                fitness(population,soln) = fitness(population,soln - 1);
                f_objective(population,soln) = f_objective(population,soln - 1);
                f_constraint(population,soln) = f_constraint(population,soln - 1);
                soln = soln + 1;
            else
                break;
            end
        end
    end
    if soln > nPop
        break;
    end
end
SapModel.SetModelIsLocked(0);
SapObject.SapModel.FrameObj.Delete('CROSS',1);
% establish all frame sections

d = des_soln(soln,1:nDB,population);
b = des_soln(soln,nDB + 1:2*nDB,population);
t = des_soln(soln, 2*nDB + 1:nDv_DIM, population);
members = zeros(nMem, 3);

for i = 1:nMem
    members(i,:) = [d(k(RegID(i,2))) b(k(RegID(i,2))) t(RegID(i,2))];
end

mem_assign = unique(members, 'rows');

for i = 1:size(mem_assign, 1)
    TubeID = strcat('HSS', num2str(mem_assign(i,1)),'x', num2str(mem_assign(i,2)),'x', num2str(mem_assign(i,3)));
    SapModel.PropFrame.SetTube(char(TubeID), 'A500GrB46', mem_assign(i,1), mem_assign(i,2), mem_assign(i,3), mem_assign(i,3));
end

% assign frame sections

for j = 1:nMem
    FrameID = num2str(j);
    TubeID = strcat('HSS', num2str(members(j,1)),'x', num2str(members(j,2)),'x', num2str(members(j,3)));
    SapModel.FrameObj.SetSection(char(FrameID), char(TubeID), 0, 0, 0);
end

% establish all cross member sections

dx = des_soln(soln, nDv_DIM + 1, population);
bx = des_soln(soln, nDv_DIM + 2, population);
tx = des_soln(soln, nDv_DIM + 3, population);

xTubeID = strcat('HSS', num2str(dx),'x', num2str(bx),'x', num2str(tx));
SapObject.SapModel.PropFrame.SetTube(char(xTubeID), 'A500GrB46', dx, bx, tx, tx);

% Add cross members

nx = 1;

for i = 1:nXm
if des_soln(soln, nDv_DIM + 3 + i, population) == 1

    Point1 = num2str(connectivity(2, i));
    Point2 = num2str(connectivity(6, i));
    Name = '';
    xID = num2str(nMem + nx);

    SapObject.SapModel.FrameObj.AddByPoint(char(Point1), char(Point2), Name, char(xTubeID), char(xID));
    SapModel.FrameObj.SetGroupAssign(char(xID), 'CROSS');

    nx = nx + 1;
end

end

nx = nx - 1;

% Determine H multipliers

len = nBin/2;
string_H = des_soln(soln, nDv_DIM + nDv_TOPO + 1:nDv_DIM + nDv_TOPO + len, population); decimal_H = 0;
string_c = des_soln(soln, nDv_DIM + nDv_TOPO + len + 1:nDv, population); decimal_c = 0;

for i = 1:len
    decimal_H = decimal_H + string_H(i)*2^(len-i);
    decimal_c = decimal_c + string_c(i)*2^(len-i);
end

% Set H multipliers and redraw model

H = H_min + decimal_H/(2^len)*(H_max - H_min); p1 = H*profile_1(:,2);
c = c_min + decimal_c/(2^len)*(c_max - c_min); p2 = c*H*profile_2(:,2);

for i = 1:nArc
    ya = arc_coords(:,2,i); ya = ya(1:JntsPerArc(i));% 2D x-coords
    za = arc_coords(:,3,i); za = za(1:JntsPerArc(i));% 2D y-coords
    if i < 10

end
A_mat = [ya(2)^3 ya(2)^2 ya(2) 1; ya(profile_1(i,1))^3 ya(profile_1(i,1))^2 ya(profile_1(i,1)) 1; ya(profile_2(i,1))^3 ya(profile_2(i,1))^2 ya(profile_2(i,1)) 1; ya(JntsPerArc(i) - 1)^3 ya(JntsPerArc(i) - 1)^2 ya(JntsPerArc(i) - 1) 1];

b_mat = [za(2); p1(i); p2(i); za(JntsPerArc(i) - 1)];

P = (inv(A_mat)*b_mat)'; A = P(1); B = P(2); C = P(3); D = P(4);

za = [za(1); A*ya(2:JntsPerArc(i) - 1).^3 + B*ya(2:JntsPerArc(i) - 1).^2 + C*ya(2:JntsPerArc(i) - 1) + D; za(JntsPerArc(i))];  zs(1:JntsPerArc(i),i,soln) = za;

elseif i > 10 && i < 15

A_mat = [ya(JntsPerArc(i) - 1)^2 ya(JntsPerArc(i) - 1) 1; ya(profile_1(i,1))^2 ya(profile_1(i,1)) 1; ya(profile_2(i,1))^2 ya(profile_2(i,1)) 1; ya(JntsPerArc(i - 1) - 1)^2 ya(JntsPerArc(i - 1) - 1) 1];

b_mat = [za(JntsPerArc(i) - 1); p1(i); p2(i)];

P = (inv(A_mat)*b_mat)'; A = P(1); B = P(2); C = P(3);

za = [A*ya(1:JntsPerArc(i) - 1).^2 + B*ya(1:JntsPerArc(i) - 1) + C; za(JntsPerArc(i))];  zs(1:JntsPerArc(i),i,soln) = za;

if i == 11

ya = arc_coords(:,2,i - 1); ya = ya(1:JntsPerArc(i - 1));% 2D x-coords
za = arc_coords(:,3,i - 1); za = za(1:JntsPerArc(i - 1));% 2D y-coords

A_mat = [ya(1)^3 ya(1)^2 ya(1) 1; ya(profile_1(i - 1,1))^3 ya(profile_1(i - 1,1))^2 ya(profile_1(i - 1,1)) 1; ya(profile_2(i - 1,1))^3 ya(profile_2(i - 1,1))^2 ya(profile_2(i - 1,1)) 1; ya(JntsPerArc(i - 1) - 1)^3 ya(JntsPerArc(i - 1) - 1)^2 ya(JntsPerArc(i - 1) - 1) 1];

b_mat = [(arc_coords(1,3,i - 2) + arc_coords(1,3,i))/2; H*profile_1(i - 1,2); c*H*profile_2(i - 1,2); za(JntsPerArc(i - 1) - 1) 1];

P = (inv(A_mat)*b_mat)'; A = P(1); B = P(2); C = P(3); D = P(4);

za = [A*ya(1:JntsPerArc(i - 1) - 1).^3 + B*ya(1:JntsPerArc(i - 1) - 1).^2 + C*ya(1:JntsPerArc(i - 1) - 1) + D; za(JntsPerArc(i - 1))];  zs(1:JntsPerArc(i - 1),i - 1,soln) = za;

end

elseif i == 15
\[
A_{\text{mat}} = \begin{bmatrix}
(ya(JntsPerArc(i))^2 & ya(JntsPerArc(i)) & 1 \\
y_a(profile_1(i,1))^2 & ya(profile_1(i,1)) & 1 \\
y_a(profile_2(i,1))^2 & ya(profile_2(i,1)) & 1
\end{bmatrix};
\]

\[
b_{\text{mat}} = \begin{bmatrix}
ya(JntsPerArc(i)) \\
p1(i) \\
p2(i)
\end{bmatrix};
\]

\[
\begin{align*}
P &= (\text{inv}(A_{\text{mat}}) * b_{\text{mat}})'; \\
A &= P(1); \\
B &= P(2); \\
C &= P(3);
\end{align*}
\]

\[
za = [A \cdot ya.^2 + B \cdot ya + C];
\]

\[
zs(1:JntsPerArc(i),i,soln) = za;
\]

\[
\text{end}
\]

\[
\text{end}
\]

\[
jnt = 1;
\]

\[
\text{for } i = 1:n\text{Arc}
\]

\[
\text{for } j = 1:JntsPerArc(i)
\]

\[
\text{JointID} = \text{num2str}(jnt);
\]

\[
\text{SapObject.SapModel.EditPoint.ChangeCoordinates}_1(\text{char(JointID)},
\text{arc_coords}(j,1,i), \text{arc_coords}(j,2,i), \text{zs}(j,i,soln), 1);
\]

\[
jnt = jnt + 1;
\]

\[
\text{end}
\]

\[
\text{end}
\]

\[
\% \text{ set steel design overwrites for cross members}
\]

\[
\text{SapModel.DesignSteel.SetCode('AISC360-05/IBC2006');}
\]

\[
\text{SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetOverwrite('CROSS',}
\text{19,0.1,1,1});
\]

\[
\text{SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetOverwrite('CROSS',}
\text{20,0.1,1,1});
\]

\[
\text{SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetOverwrite('CROSS',}
\text{29,1.0,1,1});
\]

\[
\text{SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetOverwrite('CROSS',}
\text{30,1.0,1,1});
\]

\[
\text{SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetPreference(1,3);}
\]

\[
\text{SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetPreference(2,3);}
\]

\[
\text{SapModel.View.RefreshWindow;}
\]
% Save SAP2000 model and run analysis
SapModel.Analyze.RunAnalysis();

% Start steel design check
SapModel.DesignSteel.StartDesign;
pause(2);

% Obtain steel design summary information from SAP2000
gov_ratio = zeros(1,nMem + nx);
for i = 1:nMem + nx
    FrameID = num2str(i);
    NumberItems = 0;
    FrameName = cellstr(' ');
    Ratio = zeros(1,1,'double');
    RatioType = 0;
    Location = zeros(1,1,'double');
    ComboName = cellstr(' ');
    ErrorSummary = cellstr(' ');
    WarningSummary = cellstr(' ');
    [ret NumberItems FrameName Ratio RatioType Location ComboName ErrorSummary WarningSummary] =
    SapModel.DesignSteel.GetSummaryResults(char(FrameID), NumberItems, FrameName, Ratio, RatioType, Location, ComboName, ErrorSummary, WarningSummary, 0);
    gov_ratio(i) = Ratio;
    if gov_ratio(i) > 1
        nOS_mem(i,soln) = 1;
    end
end

% Calculate fitness of current solution
[f_objective(population,soln) f_constraint(population,soln)] =
calculate_fitness(nMem, nXm, nDV_DIM, RegID, k, L, d, b, t, Lx, dx, bx, tx, gov_ratio, des_soln(soln,:,population), phase);

% if soln == 1 && population == 1 % user defined solution
% if run1 == 1
%     F = f_constraint(population,soln)/0.60;
beta = f_objective(population,soln)/(0.40*F);

else

    Tbl = strcat('A',num2str(2*nDv + 5));
    [NUM,TXT,RAW] = xlsread('Solution.xls','Sheet1',char(Tbl));
    beta = RAW;

end

beta = 6.56627868380308;

fitness(population,soln) = 1/beta*f_objective(population,soln) + f_constraint(population,soln);

end % end for soln = 1:N

% Find average weight and best solution from current generation

average_weight(population) = mean(f_objective(population,:));
average_fitness(population) = mean(fitness(population,:));
min_fitness = min(fitness(population,:));

for i = 1:nPop

    if fitness(population,i) == min_fitness

        index = i;
        break;

    end

end

design(:,population) = des_soln(index,:,population)'; % best
design per iteration
best_weight(population) = f_objective(population,index); % best
weight per iteration
percent_OS = sum(nOS_mem(:,index))/nMem; % number of overstressed members in best solution
members_OS = nOS_mem(:,index);
overstressed(population) = sum(members_OS);
design_z(:,:,population) = zs(:,:,index);

% Write solution to Excel

SOLN = zeros(2*nDv + 5,max(nPop,nIter));
SOLN(1:nDv,1:nPop) = des_soln(:,population)';
SOLN(nDv + 1:2*nDv,1:nIter) = design;
SOLN(2*nDv + 1,1:nIter) = average_weight;
SOLN(2*nDv + 2,1:nIter) = average_fitness;
SOLN(2*nDv + 3,1:nIter) = best_weight;
SOLN(2*nDv + 4,1:nIter) = overstressed;
SOLN(2*nDv + 5,1) = beta;

SOLN2(:,2*re_assign - 1:2*re_assign) = RegID;

% Check stopping criteria
if population > 1
    conv = (average_weight(population - 1) - average_weight(population))/average_weight(population - 1);
end

if conv <= 0.001 && conv >= 0
    if percent_OS <= 0.05
        xlswrite('Solution.xls',SOLN,'Sheet1','A1');
        xlswrite('Solution.xls',SOLN2,'Sheet2','A1');

        phase = phase + 1;
        SapObject.ApplicationExit(false());
        SapModel = 0;
        SapObject = 0;

        break;  % break out of "while population <= nIter"
    end
end
end

if population < nIter
    xlswrite('Solution.xls',SOLN,'Sheet1','A1');
    xlswrite('Solution.xls',SOLN2,'Sheet2','A1');
end

% Perform Genetic Algorithm operations
if any(any(diff(des_soln(:,:,population),1,1))) ~= 0

    % Perform reproduction operation
    mating_pool = zeros(nPop,nDv);
    mating_pool = reproduction(population, nPop, fitness, mating_pool, des_soln);
end

% Perform Genetic Algorithm operations
if any(any(diff(des_soln(:,:,population),1,1))) ~= 0

    % Perform reproduction operation
    mating_pool = zeros(nPop,nDv);
    mating_pool = reproduction(population, nPop, fitness, mating_pool, des_soln);
end
% Perform crossover operation

offspring = uniform_crossover(nPop, nDv, nDv_DIM, nDv_TOPO, nReg, nDB, nBin, mating_pool);

else
    offspring = des_soln(:,:,population);
end

% Perform mutation operation

offspring = mutation(nPop, nDv, nDv_DIM, nReg, nDB, k, offspring);

if population == nIter
    SOLN(1:nDv,1:nPop) = offspring';
    xlswrite('Solution.xls',SOLN,'Sheet1','A1');
    xlswrite('Solution.xls',SOLN2,'Sheet2','A1');
    SapObject.ApplicationExit(false());
    SapModel = 0;
    SapObject = 0;

    break;
end

if rem(population,3) == 0
    RegID = re_assign_regions(RegID, design(:,population), k, nDB, nDv_DIM, nMem, nArc, nXm, nReg_r, connectivity, arc_coords, design_z(:,:,population), JntsPerArc);
    re_assign = re_assign + 1;
end

population = population + 1

%% Close model

SapObject.ApplicationExit(false());
SapModel = 0;
SapObject = 0;

end % end while population <= nIter

%% Open Final Model
feature('COM_SafeArraySingleDim', 1);
feature('COM_PassSafeArrayByRef', 1);
SapObject = actxserver('Sap2000v15.SapObject');
SapObject.ApplicationStart;
SapModel = SapObject.SapModel;
FileName = strcat(cd, '\Final Model.sdb');
SapModel.File.OpenFile(FileName);
SapModel.SetModelIsLocked(0);

% establish all frame sections

df = design(1:nDB,population);
bf = design(nDB + 1:2*nDB,population);
tf = design(2*nDB + 1:nDv,population);
members = zeros(nMem,3);

for i = 1:nMem
    members(i,:) = [df(k(RegID(i,2))) bf(k(RegID(i,2)))
                   tf(RegID(i,2))];
end

mem_assign = unique(members,'rows');

for i = 1:size(mem_assign,1)
    TubeID = strcat('HSS',num2str(mem_assign(i,1)),'x',num2str(mem_assign(i,2)),'x',
                     num2str(mem_assign(i,3)));
    SapModel.PropFrame.SetTube(char(TubeID), 'A500GrB46',
                     mem_assign(i,1), mem_assign(i,2), mem_assign(i,3), mem_assign(i,3));
end

% assign frame sections

for j = 1:nMem
    FrameID = num2str(j);
    TubeID = strcat('HSS',num2str(members(j,1)),'x',num2str(members(j,2)),'x',num2str(members(j,3)));
    SapModel.FrameObj.SetSection(char(FrameID), char(TubeID), 0, 0, 0);
end

% establish all cross member sections

dxf = design(nDv_DIM + 1,population);
bxf = design(nDv_DIM + 2,population);
txf = design(nDv_DIM + 3,population);
xTubeID = strcat('HSS', num2str(dxf), 'x', num2str(bxf), 'x', num2str(txf));
SapObject.SapModel.PropFrame.SetTube(char(xTubeID), 'A500GrB46', dxf, bxf, txf, txf);

% Add cross members
nx = 1;
for i = 1:nXm
    if design(nDv_DIM + 3 + i, population) == 1
        Point1 = num2str(connectivity(2, i));
        Point2 = num2str(connectivity(6, i));
        Name = '';
        [ret, xID] = SapObject.SapModel.FrameObj.AddByPoint(char(Point1), char(Point2), Name, char(xTubeID));
        SapModel.FrameObj.SetGroupAssign(char(xID), 'CROSS');
        nx = nx + 1;
    end
end

% Redraw Model
jnt = 1;
for i = 1:nArc
    for j = 1:JntsPerArc(i)
        JointID = num2str(jnt);
        SapObject.SapModel.EditPoint.ChangeCoordinates_1(char(JointID), arc_coords(j, 1, i), arc_coords(j, 2, i), design_z(j, i, population), 1);
        jnt = jnt + 1;
    end
end

% set steel design overwrites for cross members
SapModel.DesignSteel.SetCode('AISC360-05/IBC2006');
SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetOverwrite('CROSS', 19, 0.1, 1);
SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetOverwrite('CROSS', 20, 0.1, 1);
SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetOverwrite('CROSS', 29, 1.0, 1);
SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetOverwrite('CROSS', 30, 1.0, 1);
SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetPreference(1, 3);
SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetPreference(2, 3);
phase
time = toc
SOLN(2*nDv + 5, 3) = time;
xlswrite('Solution.xls', SOLN, 'Sheet1', 'A1');

Function – calculate_fitness

function [f_objective, f_constraint] = calculate_fitness(nMem, nXm, nDv_DIM, RegID, k, L, d, b, t, Lx, dx, bx, tx, gov_ratio, des_soln, phase)

f_objective = 0;
f_constraint = 0;

rho = 0.0002836; % weight per unit volume (kip/in^3)

if phase == 1
    penalty_coef = 2.0; % penalizes ratios which are greater than 1 (failing)
else
    penalty_coef = 2.5;
end

m = nMem + 1;
for i = 1:nMem + nXm
    if i <= nMem
        f_objective = f_objective + rho*L(i)*(b(k(RegID(i,2)))*d(k(RegID(i,2))) - (b(k(RegID(i,2))) - 2*t(RegID(i,2)))*(d(k(RegID(i,2))) - 2*t(RegID(i,2))));
if gov_ratio(i) >= 1
    gov_ratio(i) = gov_ratio(i)*penalty_coef;
end
f_constraint = f_constraint + (gov_ratio(i) - 1)^2;
else
    if des_soln(nDv_DIM + 3 + (i - nMem)) == 1
        f_objective = f_objective + rho*Lx(i - nMem)*(dx*bx - (dx - tx)*(bx - tx));
        if gov_ratio(m) >= 1
            gov_ratio(m) = gov_ratio(m)*penalty_coef;
        end
        f_constraint = f_constraint + (gov_ratio(m) - 1)^2;
        m = m + 1;
    end
end
end
end % end fitness function

Function – ground_model

function [connectivity Lx nXm Xms dlx blx tlx] = ground_model()

feature('COM_SafeArraySingleDim', 1);
feature('COM_PassSafeArrayByRef', 1);
SapObject = actxserver('Sap2000v15.SapObject');
SapObject.ApplicationStart;
SapModel = SapObject.SapModel;
FileName = strcat(cd, '\Ground Model.sdb');
SapModel.File.OpenFile(FileName);
SapModel.SetModelIsLocked(0);

NumberItems = 0; % number of members in specified group
ObjectType = cellstr(' '); % array, object type of each member in the group, 2=frame object
ObjectName = cellstr(' ');
% array, name of each member in specified group
[ret nXm ObjectType Xms] = SapModel.GroupDef.GetAssignments('CROSS',
NumberItems, ObjectType, ObjectName);

Xms = str2double(Xms);

x1 = zeros(1,nXm);
y1 = zeros(1,nXm);
z1 = zeros(1,nXm);
x2 = zeros(1,nXm);
y2 = zeros(1,nXm);
z2 = zeros(1,nXm);
Lx = zeros(1,nXm);

connectivity = zeros(9,nXm);

for i = 1:nXm
    % Obtain frame element lengths from SAP2000
    % Obtain I-end and J-end points associated with current frame
element
    FrameID = num2str(Xms(i));
    Point1 = ' '
    Point2 = ' '

    [ret Point1 Point2] = SapModel.FrameObj.GetPoints(char(FrameID),
    Point1, Point2);

    % Obtain Point1 coordinates
    x = zeros(1,1,'double');
y = zeros(1,1,'double');
z = zeros(1,1,'double');

    [ret x1(i) y1(i) z1(i)] =
    SapModel.PointObj.GetCoordCartesian(char(Point1), x, y, z);

    % Obtain Point2 coordinates
    x = zeros(1,1,'double');
y = zeros(1,1,'double');
z = zeros(1,1,'double');

    [ret x2(i) y2(i) z2(i)] =
    SapModel.PointObj.GetCoordCartesian(char(Point2), x, y, z);

    % Calculate frame length
    Lx(i) = sqrt((x1(i) - x2(i))^2 + (y1(i) - y2(i))^2 + (z1(i) -
z2(i))^2);
connectivity(:,i) = [Xms(i)  str2double(Point1) x1(i) y1(i) z1(i)
str2double(Point2) x2(i) y2(i) z2(i)]';

if i == 1
    PropName = ''; 
    SAuto = ''; 
    [ret Prop SAuto] = SapModel.FrameObj.GetSection(char(FrameID), PropName, SAuto);
    FileName = ''; 
    MatProp = ''; 
    t3 = zeros(1,1,'double'); 
    t2 = zeros(1,1,'double'); 
    tf = zeros(1,1,'double'); 
    tw = zeros(1,1,'double'); 
    Color = 0; 
    Notes = ''; 
    GUID = '';

[ret FileName MatProp d1x b1x t1x  t1x Color Notes GUID] = 
SapModel.PropFrame.GetTube(char(Prop), FileName, MatProp, t3, t2, tf, tw, Color, Notes, GUID);

end
end

SapObject.ApplicationExit(false());
SapModel = 0;
SapObject = 0;

Function – initial_soln

function [d1 b1 t1 binary1 k] = initial_soln(RegID, nDB, nReg, nReg_r, nReg_c, nMem, connectivity, nXm)

feature('COM_SafeArraySingleDim', 1); 
feature('COM_PassSafeArrayByRef', 1); 
SapObject = actxserver('Sap2000v15.SapObject'); 
SapObject.ApplicationStart;
SapModel = SapObject.SapModel;
FileName = strcat(cd, '\Initial Model.sdb'); 
SapModel.File.OpenFile(FileName);
SapModel.SetModelIsLocked(0);
RegID_group = sortrows(RegID,2); 
% RegID matrix sorted by group numbers

d1 = zeros(1,nDB); 
b1 = zeros(1,nDB); 
t1 = zeros(1,nReg); 

\[ k = [\text{ones}(1,nReg_r) \text{ 2:nReg_c + 1}]; \% \text{identifies which } d \text{ and } b \text{ to use} \]

\[ m = 0; \]

\[ \text{for } i = 1:nMem \]

\[ \text{if } \text{RegID\_group}(i,2) > m \]

\[ \text{FrameID} = \text{num2str}(...) \; \text{PropName} = ''; \]
\[ \text{SAuto} = ''; \]
\[ [\text{ret Prop SAuto}] = \text{SapModel.FrameObj.GetSection(char(FrameID), PropName, SAuto)}; \]

\[ \text{FileName} = ''; \]
\[ \text{MatProp} = ''; \]
\[ t3 = \text{zeros}(1,1, 'double'); \]
\[ t2 = \text{zeros}(1,1, 'double'); \]
\[ tf = \text{zeros}(1,1, 'double'); \]
\[ tw = \text{zeros}(1,1, 'double'); \]
\[ \text{Color} = 0; \]
\[ \text{Notes} = ''; \]
\[ \text{GUID} = ''; \]

\[ [\text{ret FileName MatProp d1(k(RegID\_group(i,2))) b1(k(RegID\_group(i,2))) t1(RegID\_group(i,2)) Color Notes GUID}] = \]
\[ \text{SapModel.PropFrame.GetTube(char(Prop), FileName, MatProp, t3, t2, tf, tw, Color, Notes, GUID)}; \]

\[ m = m + 1; \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{for } i = 1:length(t1) \]

\[ \text{if } t1(i) == 0 \]

\[ \text{if } i == 1 \]

\[ t1(i) = t1(i + 1); \]

\[ \text{else} \]

\[ t1(i) = t1(i - 1); \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{end} \]

\[ \text{binary1} = \text{zeros}(1,nXm); \]

139
NumberItems = 0; % number of members in specified group
ObjectType = cellstr(’’); % array, object type of each member in
the group, 2=frame object
ObjectName = cellstr(’’); % array, name of each member in
specified group
[ret nXim ObjectType Xims] = SapModel.GroupDef.GetAssignments(’CROSS’,
NumberItems, ObjectType, ObjectName);

for i = 1:nXim
    FrameID = Xims(i);
    Point1 = ’ ’;
    Point2 = ’ ’;
    [ret Point1 Point2] = SapModel.FrameObj.GetPoints(char(FrameID),
    Point1, Point2);
    for j = 1:size(connectivity,2)
        if str2double(Point1) == connectivity(2,j) &&
        str2double(Point2) == connectivity(6,j)
            binary1(j) = 1;
            break;
        end
    end
end

% Close initial model
SapObject.ApplicationExit(false());
SapModel = 0;
SapObject = 0;

Function – mutation

function [offspring] = mutation(nPop, nDv, nDv_DIM, nReg, nDB, k,
    offspring)

db_array = [4 6 8 10 12 14 16 18 20];
t_array = [0.1875 0.2500 0.3125 0.3750 0.5000 0.6250 0.7500 1.0000
1.2500 1.5000 1.7500 2.0000];

for soln = 1:nPop
    random = randi([1,200],1,nDv);
for j = 1:nDv

    if random(j) == 1

        if j <= 2*nDB

            if j == 1 || j == nDB + 1

                START = 3; END = length(db_array);

            else

                START = 2; END = length(db_array);

            end

            index = randi([START,END]);
            offspring(soln,j) = db_array(index);

        elseif j <= nDv_DIM

            if k(j - 2*nDB) == 1 || k(j - 2*nDB) == 5

                START = 1; END = 9;

            else

                START = 5; END = length(t_array);

            end

            index = randi([START,END]);
            offspring(soln,j) = t_array(index);

        elseif j <= nDv_DIM + 2

            index = randi([1,4]);
            offspring(soln,j) = db_array(index);

        elseif j == nDv_DIM + 3

            index = randi([1,6]);
            offspring(soln,j) = t_array(index);

        else

            if offspring(soln,j) == 1

                offspring(soln,j) = 0;

            else

            end

    end
offspring(soln, j) = 1;
end
end
end
end

% Test whether any solutions are HSS6/8x6x0.1875 (this soln won't run in SAP)
offspring(soln, :) = test_db(offspring(soln, :), nReg, nDB);
end
end

Function – random_solution

function [des_soln] = random_solution(nPop, nReg, nDv, nDv_DIM, nDB, nXm, nBin, k, init_soln, des_soln, phase)

db_array = [4 6 8 10 12 14 16 18 20];
t_array= [0.1875 0.2500 0.3125 0.3750 0.5000 0.6250 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000];

for i = 1:nPop
    for j = 1:nDv_DIM + 4
        if j <= 2*nDB
            D = zeros(1, length(db_array));
            X = db_array; SD = 4; M = 100; dD = 1;

            if j == 1 || j == nDB + 1 % Roof members
                START = 3; END = length(db_array);
            else % Columns
                START = 2; END = length(db_array);
            end

            elseif j <= nDv_DIM

that the page is not complete.
D = zeros(1,length(t_array));
X = t_array; SD = 0.5; M = 10; dD = 3;

% fix: should represent roof members AND short columns

if k(j - 2*nDB) == 1 || k(j - 2*nDB) == 5 % Roof members and short columns
START = 1; END = 9;
else % Tall columns
START = 5; END = length(t_array);
end

elseif j <= nDv_DIM + 2
D = zeros(1,length(db_array));
X = db_array; SD = 3; M = 100; dD = 1;
START = 1;
for f = 1:4
if des_soln(i,1) == X(f)
END = f;
else
END = 4;
end
end

elseif j == nDv_DIM + 3
D = zeros(1,length(t_array));
X = t_array; SD = 0.5; M = 10; dD = 1;
START = 1; END = 6;
else
des_soln(i,j:nDv) = randi([0,1],1,nXm + nBin);
break;
end
if phase == 2
    for g = 1:length(X)
        if des_soln(1,j) == X(g)
            START = g - dD; END = g + dD;
            if START <= 0
                START = 1;
            elseif END > length(X)
                END = length(X);
            end
        end
    end
end

P = round(M*(pdf('Normal',X,init_soln(j),SD))); D(START:END) = P(START:END);
pool = zeros(sum(D),2); pool(:,1) = randperm(size(pool,1));
m = 1; n = 1;
    for q = 1:length(D)
        for l = 1:D(m)
            pool(n,2) = m;
            n = n + 1;
        end
        m = m + 1;
    end
    pool = sortrows(pool); pool = pool(:,2);
    rand = randi([1,length(pool)]);
    index = pool(rand);
    des_soln(i,j) = X(index);
end
% Test whether any solutions are HSS6/8x6x0.1875 (this soln wont run in SAP)

des_soln(i,:) = test_db(des_soln(i,:), nReg, nDB);
end

Function – reassign_regions

function [RegID] = re_assign_regions(RegID, design, k, nDB, nDv_DIM, nMem, nArc, nXm, nReg_r, connectivity, arc_coords, za, JntsPerArc)

feature('COM_SafeArraySingleDim', 1);
feature('COM_PassSafeArrayByRef', 1);
SapObject = actxserver('Sap2000v15.SapObject');
SapObject.ApplicationStart;
SapModel = SapObject.SapModel;
FileName = strcat(cd, '\New Model.sdb');
SapModel.File.OpenFile(FileName);
SapModel.SetModelIsLocked(0);

SapObject.SapModel.FrameObj.Delete('CROSS',1);

% establish all frame sections

df = design(1:nDB);
bf = design(nDB + 1:2*nDB);
wf = design(2*nDB + 1:nDv_DIM);
members = zeros(nMem,3);
for i = 1:nMem
    members(i,:) = [df(k(RegID(i,2))) bf(k(RegID(i,2)))
    tf(RegID(i,2))];
end
mem_assign = unique(members,'rows');
for i = 1:size(mem_assign,1)

    TubeID =
    strcat('HSS',num2str(mem_assign(i,1)),'x',num2str(mem_assign(i,2)),'x',
    num2str(mem_assign(i,3)));
    SapModel.PropFrame.SetTube(char(TubeID), 'A500GrB46',
    mem_assign(i,1), mem_assign(i,2), mem_assign(i,3), mem_assign(i,3));
end

% assign frame sections

for j = 1:nMem
    FrameID = num2str(j);
    TubeID = strcat('HSS',num2str(members(j,1)),'x',num2str(members(j,2)),'x',num2str(members(j,3)));
    SapModel.FrameObj.SetSection(char(FrameID), char(TubeID), 0, 0, 0);
end

% establish all cross member sections

dx = design(nDv_DIM + 1);
bx = design(nDv_DIM + 2);
tx = design(nDv_DIM + 3);

xTubeID = strcat('HSS',num2str(dx),'x',num2str(bx),'x',num2str(tx));
SapObject.SapModel.PropFrame.SetTube(char(xTubeID), 'A500GrB46', dx, bx, tx, tx);

% Add cross members

nx = 1;
for i = 1:nXm
    if design(nDv_DIM + 3 + i) == 1
        Point1 = num2str(connectivity(2,i));
        Point2 = num2str(connectivity(6,i));
        Name = ' ';
        xID = num2str(nMem + nx);
        SapObject.SapModel.FrameObj.AddByPoint(char(Point1),char(Point2),Name,char(xTubeID),char(xID));
    end
    nx = nx + 1;
end
end

nx = nx - 1;

% Reshape Model

jnt = 1;
for i = 1:nArc
for j = 1:JntsPerArc(i)
    JointID = num2str(jnt);
    SapObject.SapModel.EditPoint.ChangeCoordinates_1(char(JointID),
    arc_coords(j,1,i), arc_coords(j,2,i), za(j,i), 1);

    jnt = jnt + 1;
end

% set steel design overwrites for cross members
SapModel.DesignSteel.SetCode(’AISC360-05/IBC2006’);
SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetPreference(1,3);
SapObject.SapModel.DesignSteel.AISC360_05_IBC2006.SetPreference(2,3);

NumberItems = 0; % number of members in specified group
ObjectType = cellstr(’ ’); % array, object type of each member in the group, 2=frame object
ObjectName = cellstr(’ ’); % array, name of each member in specified group
[ret nRm ObjectType Rms] = SapModel.GroupDef.GetAssignments(’Roof Frames’, NumberItems, ObjectType, ObjectName);
Rms = str2double(Rms);

TubeID = strcat(’HSS’,num2str(df(1)),’x’,num2str(bf(1)),’x0.5’);
SapModel.PropFrame.SetTube(char(TubeID), ’A500GrB46’, df(1), bf(1),
0.5, 0.5);
SapObject.SapModel.FrameObj.SetSection(’Roof Frames’,char(TubeID),1);

% Save SAP2000 model and run analysis
SapModel.Analyze.RunAnalysis();

% Start steel design check
SapModel.DesignSteel.StartDesign;
pause(2);

% Obtain steel design summary information from SAP2000
gov_ratio = zeros(1,nRm);

for i = 1:nRm
    FrameID = num2str(Rms(i));
    NumberItems = 0;
FrameName = cellstr(' ');
Ratio = zeros(1,1,'double');
RatioType = 0;
Location = zeros(1,1,'double');
ComboName = cellstr(' ');
ErrorSummary = cellstr(' ');
WarningSummary = cellstr(' ');

[ret NumberItems FrameName Ratio RatioType Location ComboName ErrorSummary WarningSummary] =
SapModel.DesignSteel.GetSummaryResults(char(FrameID), NumberItems, FrameName, Ratio, RatioType, Location, ComboName, ErrorSummary, WarningSummary, 0);

gov_ratio(i) = Ratio;
end

G = max(gov_ratio);
gov_ratio = gov_ratio/G;

Group = zeros(nRm,2);
Group(:,1) = Rm;
for i = 1:length(gov_ratio)
    if gov_ratio(i) <= 0.1
        Group(i,2) = 1;
    elseif gov_ratio(i) > 0.1 && gov_ratio(i) <= 0.3
        Group(i,2) = 2;
    elseif gov_ratio(i) > 0.3 && gov_ratio(i) <= 0.6
        Group(i,2) = 3;
    elseif gov_ratio(i) > 0.6 && gov_ratio(i) <= 0.8
        Group(i,2) = 4;
    elseif gov_ratio(i) > 0.8 && gov_ratio(i) <= 0.9
        Group(i,2) = 5;
    else
        Group(i,2) = 6;
    end
end
SapModel.SetModelIsLocked(0);

for i = 1:nReg_r
    GroupID = num2str(i);
    SapModel.GroupDef.Clear(char(GroupID));
end

for i = 1:nRm
    GroupID = num2str(Group(i,2));
    FrameID = num2str(Group(i,1));
    SapModel.GroupDef.SetGroup(char(GroupID));
    SapModel.FrameObj.SetGroupAssign(char(FrameID), char(GroupID));
end

% Update RegID
RegID(1:nRm,:) = sortrows(Group,1);

%assign_solution(RegID, df, bf, nRm, Rms)

FileName = strcat(cd,'\New Model.sdb');
SapModel.File.Save(FileName);

SapObject.ApplicationExit(false());
SapModel = 0;
SapObject = 0;

Function - reproduction

function [mating_pool] = reproduction(population, nPop, fitness, mating_pool, design_solution)

average_fitness = mean(fitness(population,:));
diff_fitness = zeros(1,nPop);

for i = 1:nPop
    % Eliminate unfit design solutions
    if fitness(population,i) < average_fitness
diff_fitness(i) = average_fitness - fitness(population,i); \% new fitness, higher = better

else

diff_fitness(i) = 0;

end
end

num_copies = round(diff_fitness./sum(diff_fitness).*nPop);

A = 1:size(num_copies,2);
nc_temp = [A; num_copies]';
nc_temp = sortrows(nc_temp,2);

x = sum(num_copies) - nPop;

if x > 0
    j = 1;
else
    j = nPop;
end

while sum(nc_temp(:,2)) ~= nPop
    while nc_temp(j,2) == 0
        if x > 0
            j = j + x/abs(x);
        else
            j = nPop;
        end
    end
nc_temp(j,2) = nc_temp(j,2) - x/abs(x);

j = j + x/abs(x);

if x > 0 && j > size(nc_temp,1)
    j = 1;
end

end

nc_temp = sortrows(nc_temp,1)';
num_copies = nc_temp(2,:);

m = 1;

for i = 1:nPop
    if num_copies(i) > 0
        for j = 1:num_copies(i)
            mating_pool(m,:) = design_solution(i,:,population);
            m = m + 1;
        end
    end
end

Function – test_db

function [des_soln] = test_db(des_soln, nReg, nDB)

if des_soln(1) <= 6
    des_soln(1) = 8;
end

if des_soln(nDB + 1) <= 6
    des_soln(1) = 8;
end

C = abs(8 - [des_soln(1) des_soln(nDB + 1)]);
if C(1) + C(2) == 0
    for i = 1:nReg
        if des_soln(2*nDB + i) < 0.3125
            des_soln(2*nDB + i) = 0.3125;
        end
    end
end

Function – uniform_crossover

function [offspring] = uniform_crossover(nPop, nDv, nDv_DIM, nDv_TOPO, nReg, nDB, nBin, mating_pool)

percent_zeros = 0.70;
num_zeros = round(percent_zeros*(nDv_DIM + nDv_TOPO));

% Jumble the order of solutions in mating_pool
I = randperm(nPop)';
temp_pool = [I mating_pool];
temp_pool = sortrows(temp_pool,1);
mating_pool = temp_pool(:,2:(nDv + 1));

offspring = mating_pool;
for soln = 1:2:nPop - 1
    mask = ones(1,nDv);
    index = randperm(length(mask) - 16);
    index = index(1:num_zeros);
    index = sort(index);

    for j = 1:num_zeros
        mask(index(j)) = 0;
    end

    for k = 1:2
        L = randi([1,nBin/2]);
mask(nDv_DIM + nDv_TOPO + 1 + (nBin/2 - L) + nBin/2*(k-1):nDv_DIM + nDv_TOPO + nBin/2*k) = 0;

end

for i = 1:nDv
    if mask(i) == 0
        A = mating_pool(soln,i);
        B = mating_pool(soln + 1, i);

        offspring(soln,i) = B;
        offspring(soln + 1, i) = A;
    end
end

% Test whether any solutions are HSS6/8x6x0.1875 (this soln won't run in SAP)
offspring(soln,:) = test_db(offspring(soln,:), nReg, nDB);

end

end

Program – Phase Two

clear;
clc;
tic;

% nReg = no. roof member regions + no. column pairs = 6 + 5 = 11
nReg_r = input('Number of roof regions (NOTE: must be labelled "1", "2", etc.): '); nReg_r_old = nReg_r;
nReg_c = input('Number of column regions (NOTE: must be labelled "1", "2", etc.): '); nReg_c_old = nReg_c;
nReg = nReg_r + nReg_c; nReg_old = nReg;
nPop = input('Number of design solutions in population for phase 2: '); nPop_old = nPop;
nIter = input('Maximum number of generations (design iterations) for phase 2: ');
run1 = input('Is this the first run for this problem (1 = yes, 0 = no)?: ');
nDB = nReg_c + 1; nDB_old = nDB;
nDv = nReg + 2*nDB; % nDv = nReg t's, roof d & b, column d & b
nDv_old = nDv;
population = 1;
conv = 1;
interrupt_GA = 0;
phase = 2;
no_solution = 0;

% initialize SAP2000

feature('COM_SafeArraySingleDim', 1);
feature('COM_PassSafeArrayByRef', 1);
SapObject = actxserver('Sap2000v15.SapObject');
SapObject.ApplicationStart;
SapModel = SapObject.SapModel;

% open existing SAP2000 model and unlock

FileName = strcat(cd, '\Final Model.sdb');
SapModel.File.OpenFile(FileName);
SapModel.SetModelIsLocked(0);

% get number of frames and frame lengths from SAP2000 model

nMem = SapModel.FrameObj.Count;

x1 = zeros(1,nMem);
y1 = zeros(1,nMem);
z1 = zeros(1,nMem);
x2 = zeros(1,nMem);
y2 = zeros(1,nMem);
z2 = zeros(1,nMem);
L = zeros(1,nMem);

for i = 1:nMem
    % Obtain frame element lengths from SAP2000
    % Obtain I-end and J-end points associated with current frame
    % element
    FrameID = num2str(i);
    Point1 = ' ';
    Point2 = ' ';
    [ret Point1 Point2] = SapModel.FrameObj.GetPoints(char(FrameID), Point1, Point2);
    % Obtain Point1 coordinates
    x = zeros(1,1,'double');
    y = zeros(1,1,'double');
    z = zeros(1,1,'double');
    [ret x1(i) y1(i) z1(i)] = SapModel.PointObj.GetCoordCartesian(char(Point1), x, y, z);
    % Obtain Point2 coordinates
x = zeros(1,1,'double');
y = zeros(1,1,'double');
z = zeros(1,1,'double');

[ret x2(i) y2(i) z2(i)] = SapModel.PointObj.GetCoordCartesian(char(Point2), x, y, z);

% Calculate frame length
L(i) = sqrt((x1(i) - x2(i))^2 + (y1(i) - y2(i))^2 + (z1(i) - z2(i))^2);

end

% Get group assignments from SAP2000 (NOTE: Groups never change throughout optimization process)
RegID = zeros(nMem,2); % matrix, (member no.s in order)x(region no.s)
regions = zeros(1,nReg);
r2 = 0; % index counter
for i = 1:nReg
    RegRef = num2str(i); % name of group
    NumberItems = 0; % number of members in specified group
    ObjectType = cellstr(' '); % array, object type of each member in the group, 2=frame object
    ObjectName = cellstr(' '); % array, name of each member in specified group
    [ret NumberItems ObjectType ObjectName] = SapModel.GroupDef.GetAssignments(char(RegRef), NumberItems, ObjectType, ObjectName);
    r1 = r2 + 1; r2 = r2 + NumberItems;
    RegID(r1:r2,1) = str2double(ObjectName); % Frame numbers
    RegID(r1:r2,2) = i; % Group number
    regions(i) = NumberItems;
end % for i = 1:nReg

RegID = sortrows(RegID,1); % Sort ascending by frame number

% Get user defined solution from current SAP2000 model
RegID_group = sortrows(RegID,2); % RegID matrix sorted by group numbers

d1 = zeros(1,nDB);
b1 = zeros(1,nDB);
t1 = zeros(1,nReg);
\[ k = \text{ones}(1, n_{\text{Reg}_r}) 2:n_{\text{Reg}_c} + 1 \]; \text{ identifies which } d \text{ and } b \text{ to use }

\begin{align*}
\text{m} &= 0; \\
\text{for } i &= 1:n_{\text{Mem}} \\
\text{if } \text{RegID}_{\text{group}}(i, 2) &> m \\
\text{FrameID} &= \text{num2str}(' \text{RegID}_{\text{group}}(i, 1)'); \\
\text{PropName} &= ' ' ; \\
\text{SAuto} &= ' ' ; \\
[\text{ret } \text{Prop} \text{ SAuto}] &= \text{SapModel.FrameObj.GetSection('char(FrameID), PropName, SAuto)}; \\
\text{FileName} &= ' ' ; \\
\text{MatProp} &= ' ' ; \\
\text{t3} &= \text{zeros}(1,1,'\text{double'}); \\
\text{t2} &= \text{zeros}(1,1,'\text{double'}); \\
\text{tf} &= \text{zeros}(1,1,'\text{double'}); \\
\text{tw} &= \text{zeros}(1,1,'\text{double'}); \\
\text{Color} &= 0; \\
\text{Notes} &= ' ' ; \\
\text{GUID} &= ' ' ; \\
[\text{ret FileName MatProp d1(k(RegID_{\text{group}}(i,2)))} \\
b1(k(\text{RegID}_{\text{group}}(i,2))) \text{ t1(RegID}_{\text{group}}(i,2)) \text{ Color Notes GUID}] &= \text{SapModel.PropFrame.GetTube('char(Prop), FileName, MatProp, t3, t2, tf, tw, Color, Notes, GUID)}; \\
\text{m} &= \text{m} + 1; \\
\end{align*}

\begin{align*}
\text{end} \\
\text{end} \\
\text{end} \\
\text{% Save SAP2000 model and run analysis} \\
\text{FileName} &= \text{strcat('cd, 'Phase 2 Model.sdb');} \\
\text{SapModel.File.Save(FileName);} \\
\text{SapModel.Analyze.RunAnalysis();} \\
\text{% Start steel design check} \\
\text{SapModel.DesignSteel.StartDesign;} \\
\text{pause(2);} \\
\text{gov_ratio} &= \text{zeros}(1,n_{\text{Mem}}); \\
\text{members}_{\text{OS}_1} &= \text{zeros}(1,n_{\text{Mem}}); \ % gc > 0.95 \\
\text{members}_{\text{OS}_2} &= \text{zeros}(1,n_{\text{Mem}}); \ % gc > 1.0 \\
\text{for } i &= 1:n_{\text{Mem}} \\
\text{FrameID} &= \text{num2str}(i); \\
\text{NumberItems} &= 0; \\
\end{align*}
FrameName = cellstr(' ');  
Ratio = zeros(1,1,'double');  
RatioType = 0;  
Location = zeros(1,1,'double');  
ComboName = cellstr(' ');  
ErrorSummary = cellstr(' ');  
WarningSummary = cellstr(' ');  

[ret NumberItems FrameName Ratio RatioType Location ComboName  
ErrorSummary WarningSummary] =  
SapModel.DesignSteel.GetSummaryResults(char(FrameID), NumberItems,  
FrameName, Ratio, RatioType, Location, ComboName, ErrorSummary,  
WarningSummary, 0);  
gov_ratio(i) = Ratio;  
if gov_ratio(i) > 0.95  
    members_OS_1(i) = 1;  
end  
if gov_ratio(i) > 1  
    members_OS_2(i) = 1;  
end  
end  
SapModel.SetModelIsLocked(0);  

% Assign overstressed members to their own region  
new_regions = zeros(1,sum(members_OS_1));  
need_beef = zeros(1,sum(members_OS_2));  
s = 1; r = 1;  
db_array = [6 8 10 12 14 16 18 20];  
t_array= [0.1875 0.2500 0.3125 0.3750 0.5000 0.6250 0.7500 1.0000  
1.2500 1.5000 1.7500 2.0000];  
for i = 1:nMem  
    if regions(RegID(i,2)) > 1  
        if members_OS_1(i) == 1  
            nReg = nReg + 1;  
            new_regions(s) = i; s = s + 1;  
            regions(nReg) = 1;  
        end  
        % Add design variable  
end  
end
if RegID(i,2) > nReg_r_old

    nReg_c = nReg_c + 1;
    nDB = nDB + 1;
    d1(length(d1) + 1) = d1(k(RegID(i,2)));  
    b1(length(b1) + 1) = b1(k(RegID(i,2)));  
    k(length(k) + 1) = length(d1);

else

    nReg_r = nReg_r + 1;
    k(length(k) + 1) = 1;

end

t1(length(t1) + 1) = t1(RegID(i,2));
RegID(i,2) = nReg;

end

end

if members_OS_2(i) == 1

    need_beef(r) = i; r = r + 1;

end

end

new_regions = new_regions(new_regions ~= 0);
nDv = length(d1) + length(b1) + length(t1);

% Clear all groups
for i = 1:nReg_old

    GroupID = num2str(i);
    SapModel.GroupDef.Clear(char(GroupID));

end

% Reassign groups
for i = 1:nMem

    GroupID = num2str(RegID(i,2));
    FrameID = num2str(RegID(i,1));
    SapModel.GroupDef.SetGroup(char(GroupID));
    SapModel.FrameObj.SetGroupAssign(char(FrameID), char(GroupID));

end
% Save phase 2 model

FileName = strcat(cd,'\Phase 2 Model.sdb');
SapModel.File.Save(FileName);

% SapObject.ApplicationExit(false());
SapModel = 0;
SapObject = 0;

% Increase dimensions to make initial solution feasible

if run1 == 1
    [d1 b1 t1 iteration] = make_feasible(d1, b1, t1, k, nMem, nDB_old, RegID, need_beef, new_regions, regions);
end

%% begin phase 2 optimization

feature('COM_SafeArraySingleDim', 1);
feature('COM_PassSafeArrayByRef', 1);
SapObject = actxserver('Sap2000v15.SapObject');
SapObject.ApplicationStart;
SapModel = SapObject.SapModel;
FileName = strcat(cd,'\Phase 2 Model.sdb');
SapModel.File.OpenFile(FileName);
SapModel.SetModelIsLocked(0);

design_solution = zeros(nPop,nDv,nIter);

fitness = zeros(nIter,nPop);
f_objective = zeros(nIter,nPop);
f_constraint = zeros(nIter,nPop);

design = zeros(nDv,nIter);
average_weight = zeros(1,nIter);
average_fitness = zeros(1,nIter);
best_weight = zeros(1,nIter);
overstressed = zeros(1,nIter);

while population <= nIter && no_solution == 0;
    if population == 1 && run1 == 1
        % Convert dimension data in inches to binary data and input as first design solution
        design_solution(1,:,population) = [d1 b1 t1];
    elseif population > 1
        % Create random binary strings and input as remaining design solutions
    end

end
design_solution = random_solution_dim(nPop, nReg, nDv, nDB,
    design_solution(:,:,population));

% set size of population to variable nPop for future
calculations

elseif population == 1 && run1 == 0

    Tbl = strcat('A1:AX', num2str(nDv)); % for nPop = 50 ONLY
    [NUM,TXT,RAW] = xlsread('Solution.xls','Sheet1',char(Tbl));
    design_solution(:,:,population) = cell2mat(RAW)';

else  % population > 1

    design_solution(:,:,population) = offspring;

end % end if population == 1

nOS_mem = zeros(nMem,nPop);

for soln = 1:nPop

    % Skip analysis/calculations for repeated solutions
    if soln > 1

        while soln <= nPop

            if any(design_solution(soln,:,population) -
                design_solution(soln - 1,:,population)) == 0 %any returns 1 if any
                element is nonzero

                fitness(population,soln) = fitness(population,soln
                - 1);
                f_objective(population,soln) =
                f_objective(population,soln - 1);
                f_constraint(population,soln) =
                f_constraint(population,soln - 1);
                soln = soln + 1;
            
            else

                break;
            
        end

    end

    if soln > nPop

        break;

    end
SapModel.SetModelIsLocked(0);

% establish all frame sections

d = design_solution(soln,1:nDB,population);
b = design_solution(soln,nDB + 1:2*nDB,population);
t = design_solution(soln,2*nDB + 1:nDv,population);
members = zeros(nMem,3);

for i = 1:nMem

    members(i,:) = [d(k(RegID(i,2))) b(k(RegID(i,2)))
    t(RegID(i,2))];

end

mem_assign = unique(members,'rows');

for i = 1:size(mem_assign,1)

    TubeID = strcat('HSS',num2str(mem_assign(i,1)),'x',num2str(mem_assign(i,2)),'x',
    num2str(mem_assign(i,3)));
    SapModel.PropFrame.SetTube(char(TubeID), 'A500GrB46',
    mem_assign(i,1), mem_assign(i,2), mem_assign(i,3), mem_assign(i,3));

end

% assign frame sections

for j = 1:nMem

    FrameID = num2str(j);
    TubeID =
    strcat('HSS',num2str(members(j,1)),'x',num2str(members(j,2)),'x',num2str(members(j,3)));
    SapModel.FrameObj.SetSection(char(FrameID), char(TubeID),
    0, 0, 0);

end

SapModel.View.RefreshWindow;

% Save SAP2000 model and run analysis
SapModel.Analyze.RunAnalysis();
% Start steel design check
SapModel.DesignSteel.StartDesign;
pause(2);

% Obtain steel design summary information from SAP2000
gov_ratio = zeros(1,nMem);
for i = 1:nMem
    FrameID = num2str(i);
    NumberItems = 0;
    FrameName = cellstr(' ');
    Ratio = zeros(1,1,'double');
    RatioType = 0;
    Location = zeros(1,1,'double');
    ComboName = cellstr(' ');
    ErrorSummary = cellstr(' ');
    WarningSummary = cellstr(' ');

    [ret NumberItems FrameName Ratio RatioType Location ComboName ErrorSummary WarningSummary] =
    SapModel.DesignSteel.GetSummaryResults(char(FrameID), NumberItems, FrameName, Ratio, RatioType, Location, ComboName, ErrorSummary, WarningSummary, 0);

    gov_ratio(i) = Ratio;
    if gov_ratio(i) > 1
        nOS_mem(i,soln) = 1;
    end
end

% Calculate fitness of current solution
[f_objective(population,soln) f_constraint(population,soln)] =
    calculate_fitness_dim(nMem, RegID, k, L, d, b, t, gov_ratio, phase);

if soln == 1 && population == 1 % user defined solution
    Tbl = strcat('A',num2str(2*nDv_old + 5));
    [NUM,TXT,RAW] =
    xlsread('Solution.xls','Sheet1',char(Tbl));
    beta = RAW;
end

% CHANGE IF INITIAL SOLUTION CHANGES
%beta = 5.79797682048017;
beta = 6.56627868380308;

fitness(population, soln) = 1/beta*f_objective(population, soln) + f_constraint(population, soln);

end % end for soln = 1:N

% Find average weight and best solution from current generation

average_weight(population) = mean(f_objective(population,:));
average_fitness(population) = mean(fitness(population,:));
min_fitness = min(fitness(population,:));

for i = 1:nPop

    if fitness(population,i) == min_fitness

        index = i;
        break;

    end

end

design(:,population) = design_solution(index,:,population)';
% best design per iteration
best_weight(population) = f_objective(population,index);
% best weight per iteration
percent_OS = sum(nOS_mem(:,index))/nMem;
% number of overstressed members in best solution
members_OS = nOS_mem(:,index);
overstressed(population) = sum(members_OS);

% Write solution to Excel

SOLN = zeros(2*nDv + 4,max(nPop,nIter));
SOLN(1:nDv,1:nPop) = design_solution(:, :, population)';
SOLN(nDv + 1:2*nDv,1:nIter) = design;
SOLN(2*nDv + 1,1:nIter) = average_weight;
SOLN(2*nDv + 2,1:nIter) = average_fitness;
SOLN(2*nDv + 3,1:nIter) = best_weight;
SOLN(2*nDv + 4,1:nIter) = overstressed;
SOLN(2*nDv + 5,1) = beta;

xlswrite('Solution.xls',SOLN,'Sheet1','A1');

% Check stopping criteria

if population > 1
conv = (average_weight(population - 1) -
average_weight(population))/average_weight(population - 1);

end

if conv <= 0.001 && conv >= 0

if percent_OS == 0
    break;  % break out of "while population <= nIter"
elseif population == nIter
    break;
end

elseif population == nIter
    break;
end

% Perform Genetic Algorithm operations
if any(any(diff(design_solution(:,:,population),1,1))) ~= 0

% Perform reproduction operation
mating_pool = zeros(nPop,nDv);

mating_pool = reproduction(population, nPop, fitness, 
mating_pool, design_solution);

% Perform crossover operation
offspring = uniform_crossover_2(nPop, nDv, nReg, nDB, 
mating_pool);
else
    offspring = design_solution(:,:,population);
end

% Perform mutation operation
offspring = mutation_dim(nPop, nDv, nReg, nDB, k, offspring);

population = population + 1
end % end while population <= nIter

% establish all frame sections

df = design(1:nDB,population);
bf = design(nDB + 1:2*nDB,population);
tf = design(2*nDB + 1:nDv,population);
members = zeros(nMem,3);

for i = 1:nMem
    members(i,:) = [df(k(RegID(i,2))) bf(k(RegID(i,2))) tf(RegID(i,2))];
end

mem_assign = unique(members,'rows');

for i = 1:size(mem_assign,1)
    TubeID = strcat('HSS',num2str(mem_assign(i,1)),'x',num2str(mem_assign(i,2)),'x',num2str(mem_assign(i,3)));
    SapModel.PropFrame.SetTube(char(TubeID), 'A500GrB46', mem_assign(i,1), mem_assign(i,2), mem_assign(i,3), mem_assign(i,3));
end

% assign frame sections

for j = 1:nMem
    FrameID = num2str(j);
    TubeID = strcat('HSS',num2str(members(j,1)),'x',num2str(members(j,2)),'x',num2str(members(j,3)));
    SapModel.FrameObj.SetSection(char(FrameID), char(TubeID), 0, 0, 0);
end

time = toc

Function – calculate_fitness_dim

function [f_objective, f_constraint] = calculate_fitness_dim(nMem, RegID, k, L, d, b, t, gov_ratio, phase)

    f_objective = 0;
f_constraint = 0;

    rho = 0.0002836;   % weight per unit volume (kip/in^3)
if phase == 1
    penalty_coef = 2.0; % penalizes ratios which are greater than 1 (failing)
else
    penalty_coef = 2.5;
end
for i = 1:nMem
    f_objective = f_objective +
    rho*L(i)*(b(k(RegID(i,2)))*d(k(RegID(i,2))) - (b(k(RegID(i,2))) -
    2*t(RegID(i,2)))*(d(k(RegID(i,2))) - 2*t(RegID(i,2))));
    if gov_ratio(i) >= 1
        gov_ratio(i) = gov_ratio(i)*penalty_coef;
    end
    f_constraint = f_constraint + (gov_ratio(i) - 1)^2;
end
end % end fitness function

Function – increase_db

function [D B x] = increase_db(D, B)

db_array = [6 8 10 12 14 16 18 20];
x = 0;
if D(2,1) == 0
    for i = 1:length(db_array) - 1
        if D(1,1) == db_array(i)
            D(1,1) = db_array(i + 1); D(2,1) = 0; x = 0;
            break;
        else
            D(2,1) = 1; x = 1;
        end
    end
end
if B(2,1) == 0
    for i = 1:length(db_array) - 1
        if B(1,1) == db_array(i)
            B(1,1) = db_array(i + 1); B(2,1) = 0; x = 0;
            break;
        else
            B(2,1) = 1; x = 1;
        end
    end
end

function [d1 b1 t1 iteration] = make_feasible(d1, b1, t1, k, nMem, nDB_old, RegID, need_beef, new_regions, regions)

%%

feature('COM.SafeArraySingleDim', 1);
feature('COM.PassSafeArrayByRef', 1);
SapObject = actxserver('Sap2000v15.SapObject');
SapObject.ApplicationStart;
SapModel = SapObject.SapModel;
FileName = strcat(cd, '\Phase 2 Model.sdb');
SapModel.File.OpenFile(FileName);
SapModel.SetModelIsLocked(0);

iteration = 1;

db_array = [6 8 10 12 14 16 18 20];
t_array= [0.1875 0.2500 0.3125 0.3750 0.5000 0.6250 0.7500 1.0000
1.2500 1.5000 1.7500 2.0000];

D = d1; B = b1; T = t1;
D(2,:) = ones(1,length(D)); B(2,:) = ones(1,length(B)); T(2,:) =
ones(1,length(T));

%
for i = 1:nMem
    I = sort(abs(i - need_beef));
    if I(1) == 0
        T(2,RegID(i,2)) = 0;
        if k(RegID(i,2)) > nDB_old || (regions(RegID(i,2)) == 1 && k(RegID(i,2)) > 1)
            D(2,k(RegID(i,2))) = 0; B(2,k(RegID(i,2))) = 0;
        end
    end
end

nOS = length(need_beef);
while nOS(iteration) > 0
    X = 0;
    for i = 1:length(T)
        if T(2,i) == 0
            for j = 1:length(t_array) - 1
                if T(1,i) == t_array(j)
                    T(1,i) = t_array(j + 1); T(2,i) = 0;
                    break;
                else % t maxed out
                    if j == length(t_array) - 1
                        T(2,i) = 1;
                        if k(i) > nDB_old || (regions(i) == 1 && k(i) > 1)
                            [D(:,k(i)) B(:,k(i))] = increase_db(D(:,k(i)), B(:,k(i)));
                        end
                    end
                end
            end
        end
    end
end
end
else
    if D(2,k(i)) == 0 || B(2,k(i)) == 0
        [D(:,k(i)) B(:,k(i))] = increase_db(D(:,k(i)), B(:,k(i)));
    end
end
end

d1_old = d1; b1_old = b1; t1_old = t1;
d1 = D(1,:);
b1 = B(1,:);
t1 = T(1,:);
if sum(d1 - d1_old) + sum(b1 - b1_old) + sum(t1 - t1_old) == 0
    break;
end
members = zeros(nMem,3);
for i = 1:nMem
    members(i,:) = [d1(k(RegID(i,2))) b1(k(RegID(i,2))) t1(RegID(i,2))];
end
mem_assign = unique(members,'rows');
SapModel.SetModelIsLocked(0);
for i = 1:size(mem_assign,1)
    TubeID = strcat('HSS',num2str(mem_assign(i,1)),'x',num2str(mem_assign(i,2)),'x', num2str(mem_assign(i,3)));
    SapModel.PropFrame.SetTube(char(TubeID), 'A500GrB46', mem_assign(i,1), mem_assign(i,2), mem_assign(i,3), mem_assign(i,3));
end
% assign frame sections
for j = 1:nMem

    FrameID = num2str(j);
    TubeID = strcat('HSS',num2str(members(j,1)),'x',num2str(members(j,2)),'x',num2str(members(j,3)));
    SapModel.FrameObj.SetSection(char(FrameID), char(TubeID), 0, 0, 0);
end

SapModel.View.RefreshWindow;

% Save SAP2000 model and run analysis
FileName = strcat(cd,'\Phase 2 Model.sdb');
SapModel.File.Save(FileName);
SapModel.Analyze.RunAnalysis();

% Start steel design check
SapModel.DesignSteel.StartDesign;
pause(2);

% Obtain steel design summary information from SAP2000

gov_ratio = zeros(1,nMem);
nOS_mem = zeros(1,nMem);

for i = 1:nMem

    FrameID = num2str(i);
    NumberItems = 0;
    FrameName = cellstr(' ');
    Ratio = zeros(1,1,'double');
    RatioType = 0;
    Location = zeros(1,1,'double');
    ComboName = cellstr(' ');
    ErrorSummary = cellstr(' ');
    WarningSummary = cellstr(' ');

    [ret NumberItems FrameName Ratio RatioType Location ComboName ErrorSummary WarningSummary] =
    SapModel.DesignSteel.GetSummaryResults(char(FrameID), NumberItems, FrameName, Ratio, RatioType, Location, ComboName, ErrorSummary,
    WarningSummary, 0);

    gov_ratio(i) = Ratio;
    A = sort(abs(i - new_regions));
if A(1) == 0

    if gov_ratio(i) > 1

        nOS_mem(i) = 1;
        T(2,RegID(i,2)) = 0;

        if k(RegID(i,2)) > nDB_old || (regions(RegID(i,2)) == 1
        && k(RegID(i,2)) > 1)

            D(2,k(RegID(i,2))) = 0; B(2,k(RegID(i,2))) = 0;

        end

    else % remove from need_beef list so it is no longer
    updated

        for j = 1:length(new_regions)

            if i == new_regions(j)

                T(2,RegID(i,2)) = 1; D(2,k(RegID(i,2))) = 1;
                B(2,k(RegID(i,2))) = 1;

            end

        end

    end

end

end

nOS = [nOS sum(nOS_mem)];
iteration = iteration + 1;
end

iteration = iteration - 1

SapObject.ApplicationExit(false());
SapModel = 0;
SapObject = 0;
Function – mutation_dim

```matlab
function [offspring] = mutation_dim(nPop, nDv, nReg, nDB, k, offspring)

db_array = [6 8 10 12 14 16 18 20];
t_array = [0.1875 0.2500 0.3125 0.3750 0.5000 0.6250 0.7500 1.0000
1.2500 1.5000 1.7500 2.0000];

for soln = 1:nPop
    random = randi([1,200],1,nDv);
    for j = 1:nDv
        if random(j) == 1
            if j <= 2*nDB
                if j == 1 || j == nDB + 1
                    START = 1; END = 6;
                else
                    START = 1; END = length(db_array);
                end
                index = randi([START,END]);
                offspring(soln,j) = db_array(index);
            else
                if k(j - 2*nDB) == 1
                    START = 1; END = 9;
                else
                    START = 1; END = length(t_array);
                end
                index = randi([START,END]);
                offspring(soln,j) = t_array(index);
            end
        end
    end
end
```

% Test whether any solutions are HSS6/8x6x0.1875 (this soln wont run in SAP)

offspring(soln,:) = test_db(offspring(soln,:), nReg, nDB);
end
end

random_solution_dim

function [design_solution] = random_solution_dim(nPop, nReg, nDv, nDB, design_solution)

db_array = [4 6 8 10 12 14 16 18 20];
t_array= [0.1875 0.2500 0.3125 0.3750 0.5000 0.6250 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000];

for i = 2:nPop
    for j = 1:nDv
        if j <= 2*nDB
            D = zeros(1,length(db_array));
            X = db_array; SD = 3; M = 100; dD = 1;
        else
            D = zeros(1,length(t_array));
            X = t_array; SD = 0.5; M = 10; dD = 3;
        end
        for g = 1:length(X)
            if design_solution(1,j) == X(g)
                START = g - dD; END = g + dD;
                if START <= 0
                    START = 1;
                elseif END > length(X)
                    END = length(X);
                end
            end
        end
    end
end
end

end

P = round(M*(pdf('Normal',X,design_solution(1,j),SD)));
D(START:END) = P(START:END);
pool = zeros(sum(D),2); pool(:,1) = randperm(size(pool,1));
m = 1; n = 1;
for q = 1:length(D)
    for l = 1:D(m)
        pool(n,2) = m;
        n = n + 1;
    end
    m = m + 1;
end
pool = sortrows(pool); pool = pool(:,2);
rand = randi([1,length(pool)]);
index = pool(rand);
design_solution(i,j) = X(index);
end

% Test whether any solutions are HSS6/8x6x0.1875 (this soln wont run in SAP)
design_solution(i,:) = test_db(design_solution(i,:), nReg, nDB);
end
end

Function – uniform_crossover_2

function [offspring] = uniform_crossover_2(nPop, nDv, nReg, nDB, mating_pool)

percent_zeros = 0.70;
um_zeros = round(percent_zeros*nDv);

% Jumble the order of solutions in mating_pool
I = randperm(nPop)';

temp_pool = [I mating_pool];
temp_pool = sortrows(temp_pool,1);
mating_pool = temp_pool(:,2:(nDv + 1));

offspring = mating_pool;

for soln = 1:2:nPop - 1

    mask = ones(1,nDv);
    index = randperm(length(mask));
    index = index(1:num_zeros);
    index = sort(index);

    for j = 1:num_zeros

        mask(index(j)) = 0;

    end

    for i = 1:nDv

        if mask(i) == 0

            A = mating_pool(soln,i);
            B = mating_pool(soln + 1, i);

            offspring(soln,i) = B;
            offspring(soln + 1, i) = A;

        end

    end

end

% Test whether any solutions are HSS6/8x6x0.1875 (this soln wont run in SAP)

offspring(soln,:) = test_db(offspring(soln,:), nReg, nDB);

end

end